Gould MPX-32™

Release 3.3

Technical Manual

Volume I

December 1986

Publication Order Number: 322-001551-200

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., 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.

PROPRIETARY INFORMATION

The information contained herein is proprietary to Gould CSD and/or its vendors, and its use, disclosure or duplication is subject to the restrictions stated in the Gould CSD license agreement Form No. 620-06 or the applicable third-party sublicense agreement.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the
Government is subject to restrictions
as set forth in subdivision (b) (3) (ii) of
the Rights in Technical Data and Computer
Software clause at 52.227.7013

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

HISTORY

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

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

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

The Gould MPX-32 Release 3.3 Technical Manual Volume I, Publication Order Number **322-001551-200** (Revision 2, Release 3.3), was printed December 1986.

This manual contains the following pages:

Title page
Copyright page
iii/iv through xvii/xviii
1-1 through 1-62
2-1 through 2-164
3-1 through 3-37/38
4-1 through 4-20
5-1 through 5-35/36
6-1 through 6-25/6-26
7-1 through 7-16
8-1 through 8-12
A-1 through A-3/A-4

CONTENTS

CHAPTER 2 - SYSTEM TABLES AND VARIABLES

CHAPTER 3 - SYSTEM TASK DESCRIPTIONS

CHAPTER 4 - SYSTEM GENERATION TASK DESCRIPTION

CHAPTER 5 - BATCH TASK DESCRIPTIONS

CHAPTER 6 - SYSTEM TRACE

CHAPTER 7 - SYSTEM INITIALIZERS AND BUILDERS

CHAPTER 8 - INTERNAL PROCESSING UNIT (IPU)

FIGURES

TABLES

Documentation Conventions

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

lowercase letters

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

```
!ACTIVATE taskname
```

means replace taskname with the name of a task. For example,

```
!ACTIVATE DOCCONV
```

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

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

means a break occurred on the specified task.

UPPERCASE LETTERS

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

```
SAVE filename
```

means enter SAVE followed by a filename. For example,

```
SAVE DOCCONV
```

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

```
taskname,taskno ABORTED

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

Braces { }

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

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

means enter the value for either counter or startbyte.

Brackets [ ]

An element inside brackets is optional. For example,

[CURR]

means the term CURR is optional.

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

[base name]
[progname]

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

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

TRACE [lower address [upper address]]

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

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

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

could be coded as

M.DFCB FCB12,IN

or

M.DFCB FCB12,IN,,ERRAD

or

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

Horizontal Ellipsis . . .

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

name [,name]...

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

Vertical Ellipsis .
.
.

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

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

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

Numbers and Special Characters

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

(value)

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

Underscore

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

ACTIVATE taskname

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

RESET

means type either RESET or RST.

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

TSM > EDIT

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

Subscript Delta $_{\Delta}$

A subscript delta specifies a required space. For example,

EDT > STO$_{\Delta}$TSSPGM

means a space is required between O and T.

CHAPTER 1

SYSTEM DESCRIPTION

1.1 Naming Conventions

To assist in the identification of system components, the following naming conventions are used in MPX-32 software and documentation.

1.1.1 Communications Region

Names of variables within the MPX-32 communications region are prefixed by the characters "C.". The general form is C.x where x is a string of one to six characters.

1.1.2 Task Service Area (TSA)

Names of variables within the TSA associated with each task are prefixed by the characters "T.". The general form is T.x where x is a string of one to six characters.

1.1.3 Entry Variables

Names of variables within table and file entries consist of characters which identify the table or file and the variable. The general form is x.y where x consists of two to four characters which identify the table and y consists of three to six characters which identify the variable. Table or file name prefixes (x) are as follows:

ART	Allocated Resource Table
CDT	Controller Definition Table
CHT	IOP Channel Definition Table
DAT	Dispatch Queue Address Table
DCA	Device Context Area
DFT	Disc File Assignment Table
DQE	Dispatch Queue Entry Table
DTT	Device Type Table
FCB	File Control Block
FPT	File Pointer Table
ICB	Interrupt Control Block
IOQ	I/O Queue Entry
JOB	Job Table
MEM	Memory Allocation Table
MEML	Memory Attribute List
MIDL	Map Image Descriptor List
MQ	Message or Run Request Queue Entry
MVT	Mounted Volume Table
PRB	Parameter Receive Block
PSB	Parameter Send Block

RCB	Resource Create Block
RD	Resource Descriptor
RLB	Resource Logging Block
RRS	Resource Requirement Summary Entry
RXB	Receiver Exit Block
SMD	System Master Directory Entry
SMT	Shared Memory Table
TCA	Terminal Context Area
TCP	Type Control Parameter Block
UDT	Unit Definition Table
VAT	Volume Assignment Table

1.1.4 System Modules and Interrupt Handlers

Names of system modules and interrupt handlers are prefixed by the characters "H.". The general form is H.x where x is a string of one to six characters. Entry points in system modules are identified by the module name, followed by the entry point's numeric identifier. Entry point names are of the general form H.x,n, where n is the numeric entry point identifier.

1.1.5 Common System Subroutines

Common system subroutines are subroutines contained within modules intended for use by other modules. Their names are prefixed by the characters "S.". The general form is S.xn, where x is the one to four character module identifier and n is the subroutine numeric identifier. For example, S.EXEC1 is the first subroutine in the H.EXEC module.

1.1.6 System Macros

Names of nonbase mode system macros are prefixed by the characters "M.". Names of base mode system macros are prefixed by "M_". The general form is M.x or M_x, where x is a string of one to six characters for nonbase mode or one to fourteen characters for base mode.

1.1.7 System Task Load Module Files

Names of system task load module files are prefixed by the characters "J.". The general form is J.x, where x is a string of one to six characters.

1.1.8 Batch Task Load Module and Executable Image Files

Names of system batch task load module files are identical to the names of the tasks contained on the files.

1.1.9 System Permanent Files

Names of system permanent files not containing load modules are prefixed by the characters "M.". The general form is M.x, where x is a string of one to six characters. M.ERR, M.CNTRL, and M.KEY are examples of system permanent files.

1.2 Scheduler - IOCS Interface

I/O Initiation

A task issues an SVC to enter IOCS. I/O services for pretransfer processing are then executed at the software priority level of the requesting task. Once the I/O request is initiated (or queued for initiation), an H.EXEC entry point is called to report the event to the CPU and swapping scheduler:

Entry Point	Event
H.EXEC,1	Interactive input starting
H.EXEC,2	Terminal output starting
H.EXEC,3	Wait I/O starting
H.EXEC,4	No-wait I/O starting

Wait I/O Postprocessing

A return will be made to IOCS from H.EXEC,1, 2, or 3 only upon completion of the I/O request. Post transfer processing may then occur at the software priority level of the requesting task.

No-wait I/O Postprocessing

A return from H.EXEC,4 will be made immediately after recording the no-wait I/O event. Since IOCS will also make an immediate return to the user task, no-wait I/O post transfer processing will occur as a task interrupt service.

No-wait I/O Completion Task Interrupt Service

When the I/O handler interrupt service routine fields a completion interrupt for a no-wait I/O request, it calls the executive subroutine S.EXEC4 to report the event. The I/O queue entry associated with the call is then linked to the task interrupt list in the DQE of the task that made the I/O request. When the scheduler attempts to dispatch control to the task, it finds that a task interrupt is outstanding. Task interrupts are inhibited during execution of any system service for a task. No task interrupt is honored while a higher priority task interrupt is active. When the task interrupt is honored, control is transferred to the IOCS routine specified in the Pre-emptive System Service Header of the I/O queue entry. Posttransfer processing then occurs at the software priority level of the requesting task. When postprocessing of the no-wait I/O request is complete, the task interrupt service is exited by a call to S.EXEC6 or H.EXEC,12.

No-wait I/O Restrictions for System Services

Posttransfer processing for a no-wait I/O request is processed as a task interrupt. Task interrupts are not honored while the task is executing in a system service (PC .LE. TSA address). An exception is made for a task that is in a wait for any no-wait I/O completion state. A task interrupt generated by the completion of no-wait I/O is honored if the task is in the wait for any no-wait I/O completion state. A system service that wants no-wait I/O can issue a series of no-wait calls followed by a wait-for-any call. Be careful that all outstanding calls are completed appropriately.

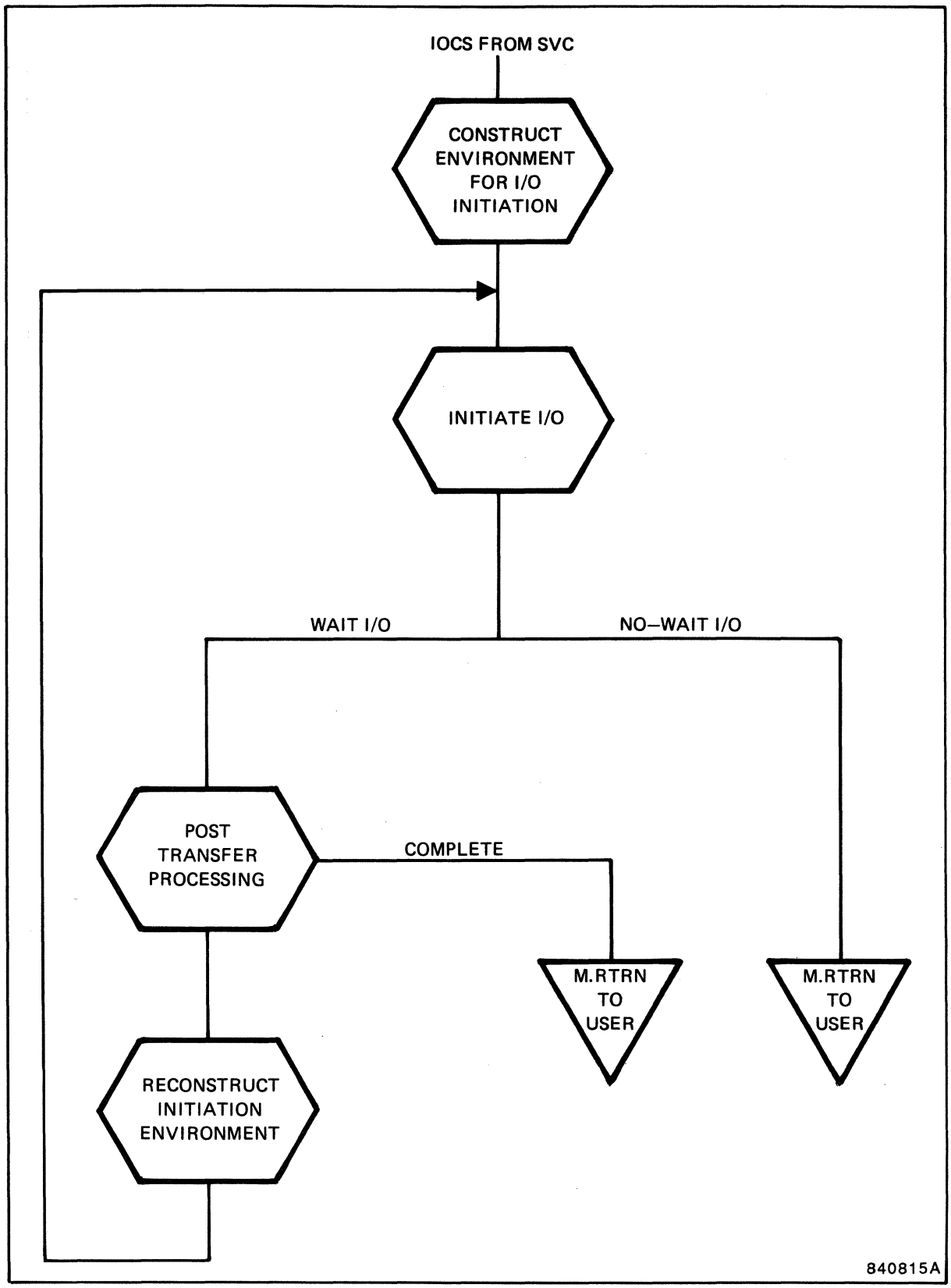


Figure 1-1. Scheduler - IOCS Interface - IOCS I/O SVC Processing Overview

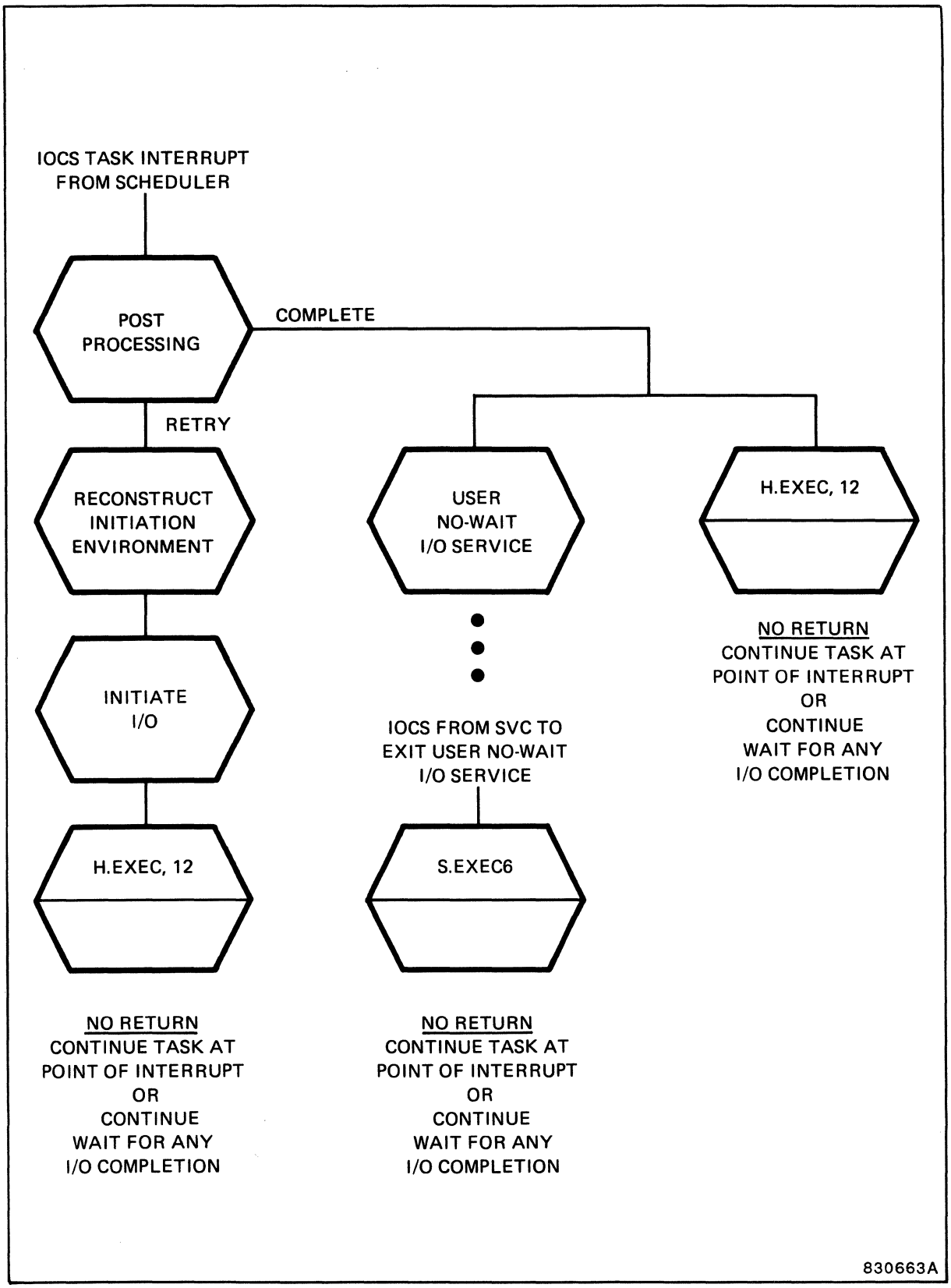


Figure 1-2. Scheduler - IOCS Interface - IOCS No-wait I/O Postprocessing Overview

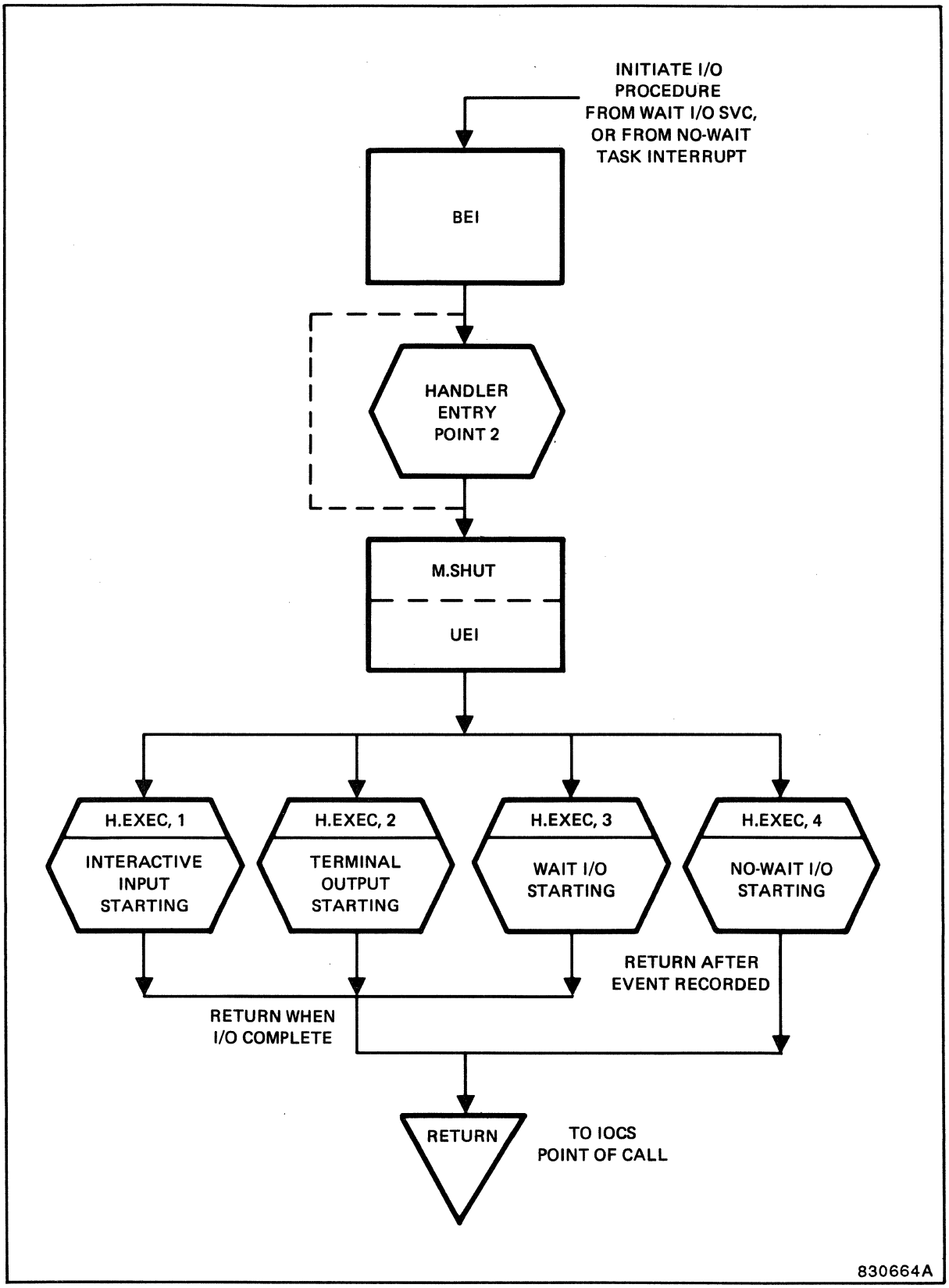


Figure 1-3. Scheduler - IOCS Interface - IOCS Initiate I/O Procedure

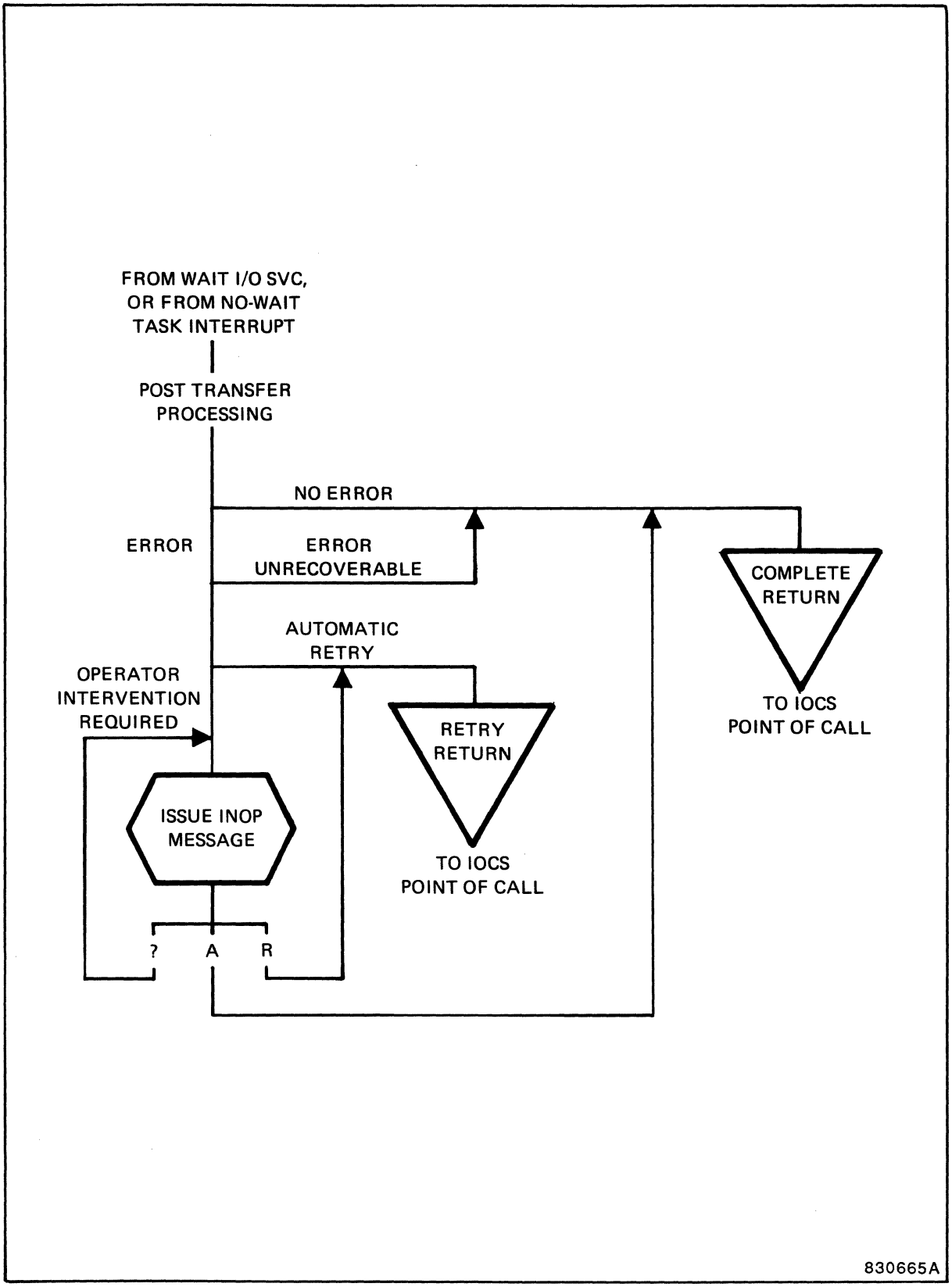


Figure 1-4. Scheduler - IOCS Interface - IOCS Postprocessing Procedure

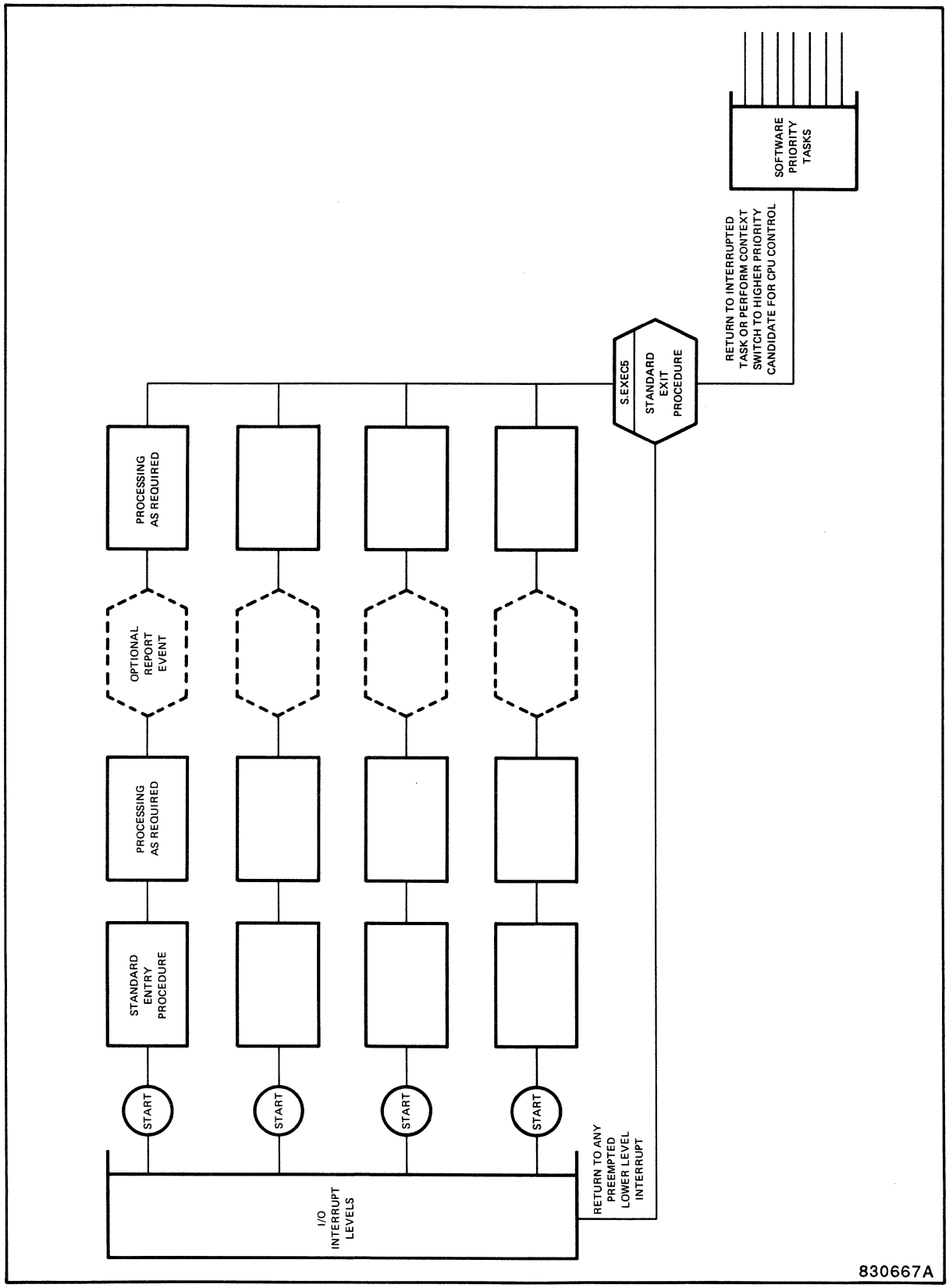


Figure 1-5. Scheduler - I/O Interrupt Interface Overview

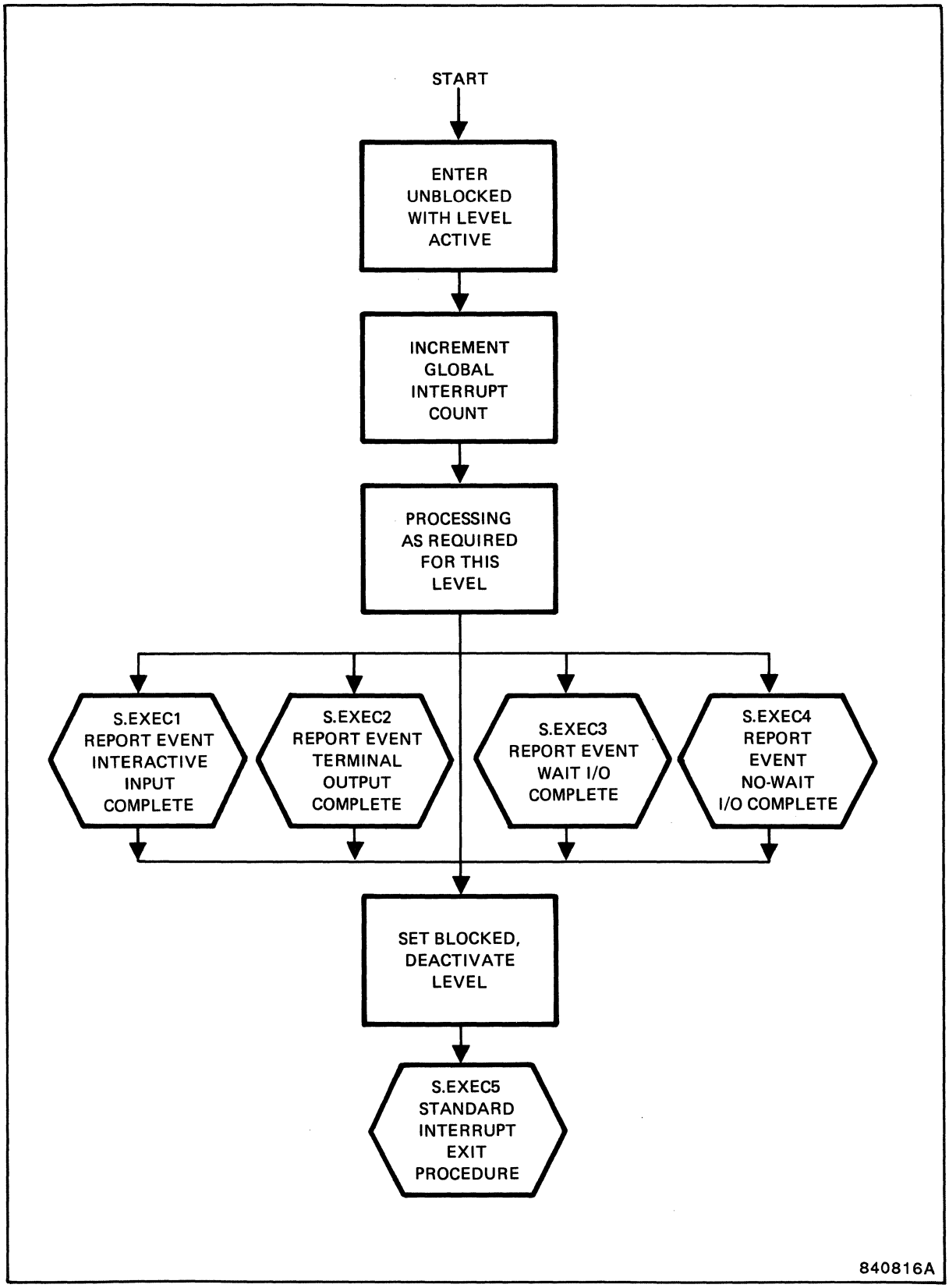


Figure 1-6. Scheduler - I/O Interrupt - Interface, Procedures

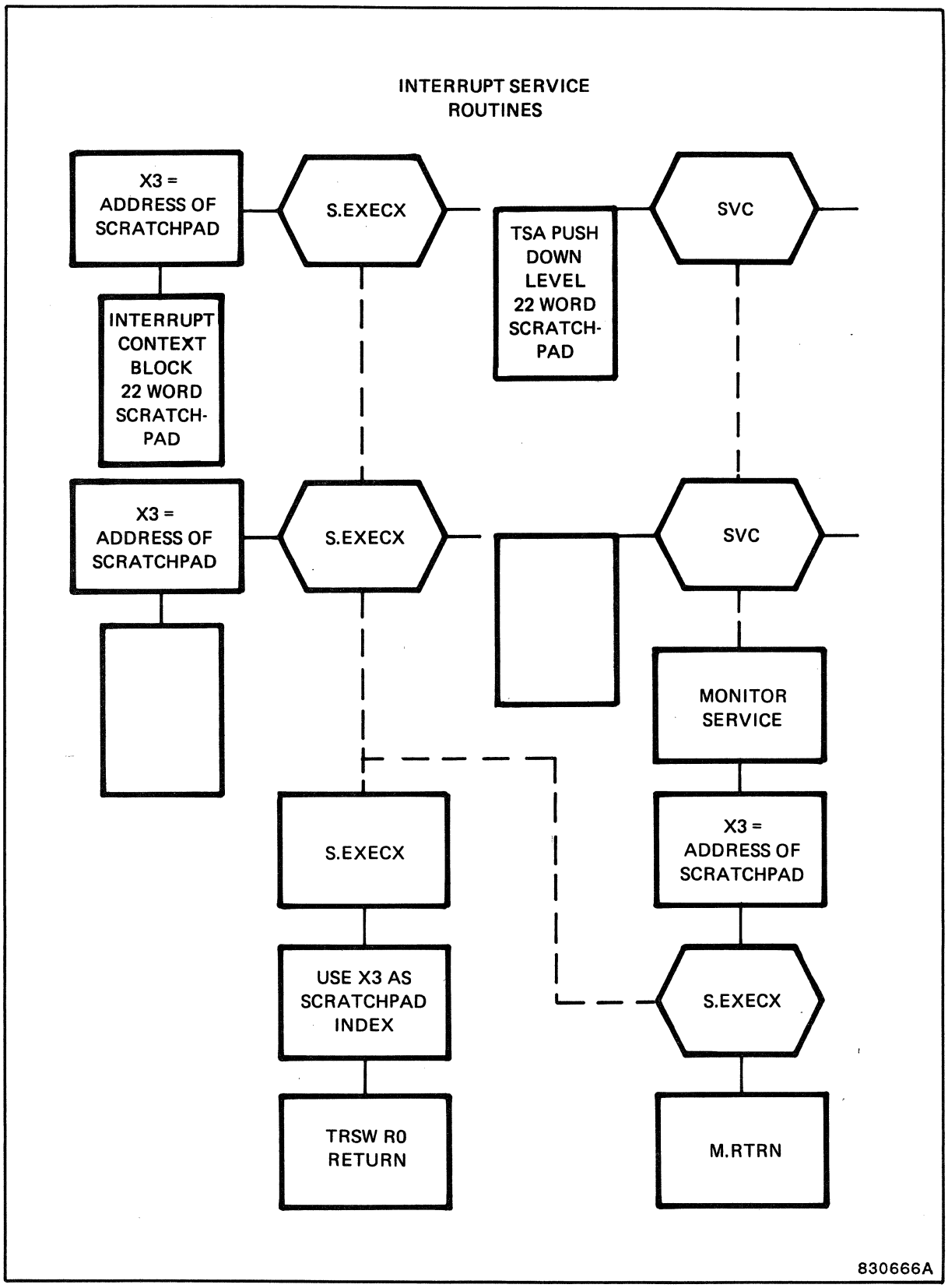


Figure 1-7. Scheduler - I/O Interrupt Interface, Re-entrant Subroutines

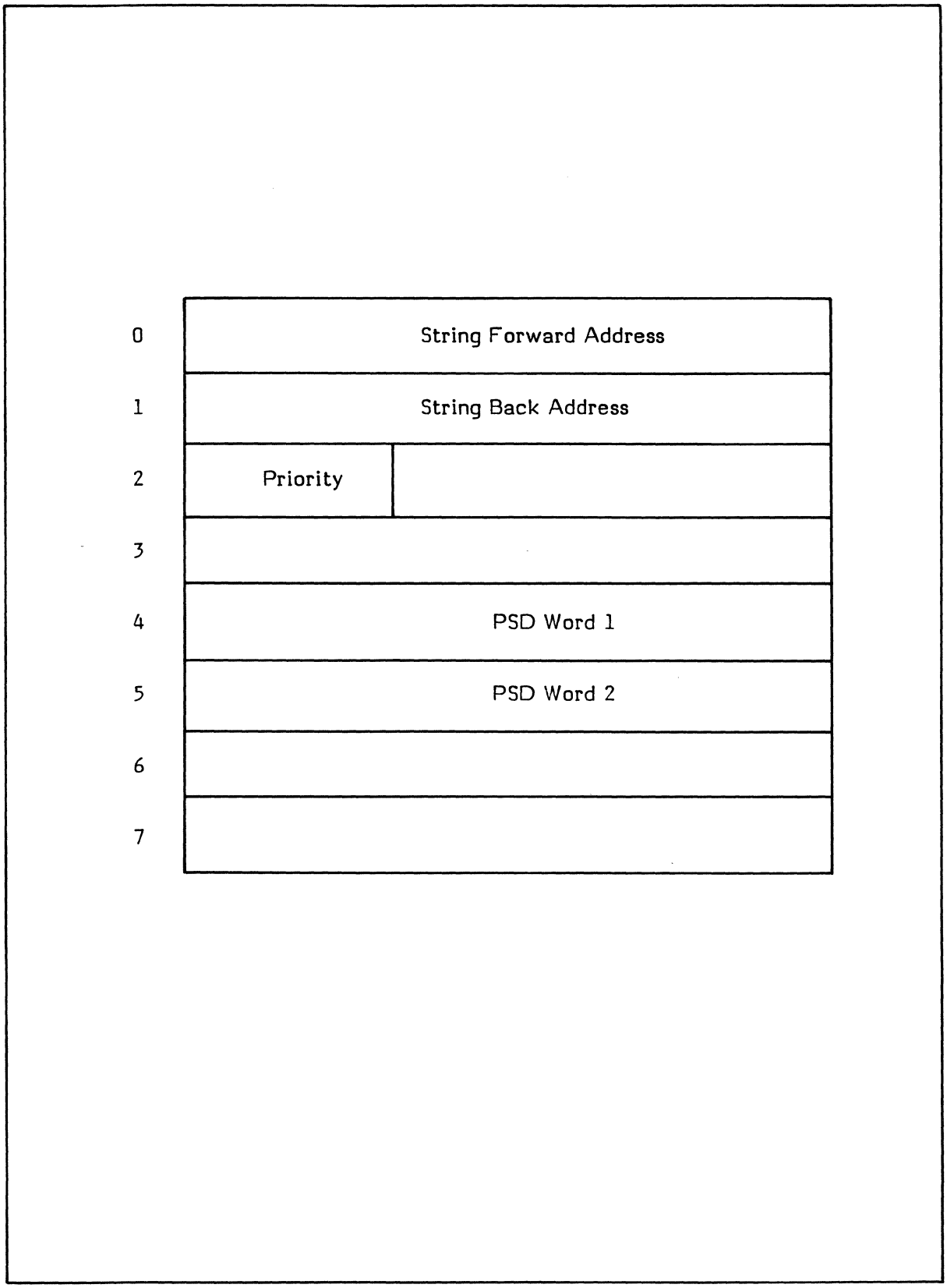


Figure 1-8. Pre-emptive System Service List Entry Header Format

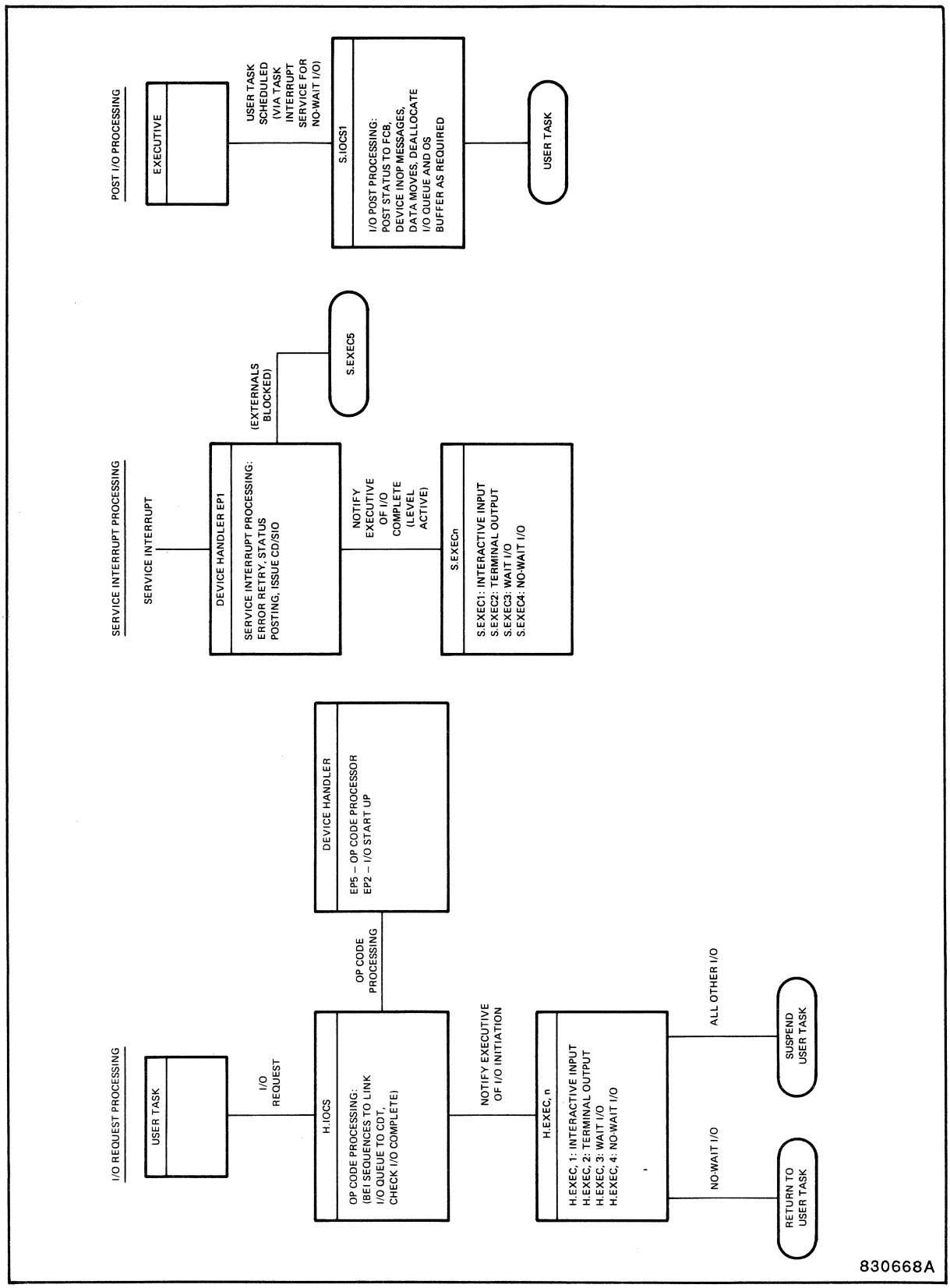


Figure 1-9. I/O Overview from User Request to I/O Complete

1.3 Scheduler - Task Termination Interface

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

1.3.1 Exit Task

The exit task service is called by a task that needs to terminate its execution in a normal fashion.

Outstanding I/O (Exit)

If an exiting task has outstanding I/O, further exit processing is deferred until all outstanding I/O is complete. Any user end action routines associated with no-wait I/O which completes while a task is exiting result in a task abort.

Messages in Receiver Queue (Exit)

All outstanding messages sent to an exiting task are unlinked from the message receiver queue and treated as complete with abnormal status.

Outstanding Run Requests (Exit)

A task attempting to exit with outstanding no-wait run requests (with call back) for other tasks is aborted.

Run Requests in Receiver Queue (Exit)

If an exiting task has requests in its run receiver queue, the current run request is terminated and the appropriate status is posted in the run request parameter block. If any additional run requests are queued, a new copy of the task is activated.

Task Abort Receiver (Exit)

A task abort receiver is not processed on task exit.

Files (Exit)

All open files associated with a task are automatically closed during task exit processing.

Resources (Exit)

All resources associated with a task are automatically deallocated during task exit processing.

1.3.2 Abort Task

The abort task service is called by a task that needs to terminate its execution in an abnormal fashion. It is also initiated by the system when a task encounters a system trap condition, such as undefined instruction, privilege violation, or nonpresent memory, or by a system service because of a parameter validation error. This service is asynchronously initiated by another task or by operator communications. If the OWNERNAME restriction is set in T.ACCESS, only a task of the same owner name can initiate the abort.

Asynchronous Abort

When a task needs to abort another task it calls the asynchronous abort service. If the OWNERNAME restriction is set in T.ACCESS, only a task of the same owner name can initiate the abort. The task to be aborted is in a ready-to-run state or one of the three following wait states:

1. Waiting for execution signal:

    - timed suspend
    - message receive
    - run-request receive
    - interrupt receive

2. Waiting for resource:

    - device
    - disc space
    - memory
    - memory pool

3. Waiting for operation complete:

    - interactive input
    - low speed output
    - any no-wait I/O
    - wait I/O
    - any no-wait message
    - wait message
    - any no-wait run request
    - wait run request

If the specified task to be aborted is waiting for an execution signal, an abort request bit is set in the DQE. The DQE is unlinked from its current state queue and linked to the ready-to-run list at its current priority. When it is selected for execution by the CPU scheduler, the abort request processing then proceeds for the aborting task.

If the specified task is waiting for a resource or operation complete, the abort requested bit is set in its DQE. The task remains linked to its current list, and abort processing does not proceed until outstanding operations are complete and the task is ready to run.

Synchronous Aborts

When the currently executing task encounters an abort condition, the abort bit is set in the DQE. The CPU scheduler then processes the abort request. The following is an outline of synchronous abort processing.

Outstanding I/O

If the aborting task has outstanding I/O, further abort processing is deferred until all outstanding I/O is complete. End-action routine execution is inhibited, and task abort status is reflected in the FCB.

Messages in Receiver Queue

All outstanding messages sent to an aborting task are unlinked from the message receiver queue and treated as complete with abnormal status.

Outstanding Run Requests

If the aborting task has outstanding run requests (with call back) for other tasks, further abort processing is deferred until completion of all such requests. End-action routine execution is inhibited, and task abort status is reflected in the run-request block.

Run Requests in Receiver Queue

If the aborting task has requests in its run receiver queue, the current run request is terminated and the appropriate status is posted in the run-request parameter block. If any additional run requests are queued, a new copy of the task is activated.

Abort Receiver

If the aborting task has an abort receiver, control is transferred to it. All outstanding operation or resource waits have been completed, and all no-wait I/O or no-wait run requests (with call back) have been completed when the abort receiver is entered. End-action routines associated with no-wait operations that completed while the abort request was outstanding have not been executed. Status bits reflecting this are posted in the appropriate FCBs and PSBs. Any files open when the abort request was received remain open on an abort receiver entry. Any resources allocated when the abort request was received remain allocated when the abort receiver is executed.

Open Files

If the aborting task has no intercepting abort receiver, all files open when the abort request was encountered are automatically closed.

Resources

If an aborting task has no intercepting abort receiver, all previously allocated resources are deallocated and the task is no longer active in the system.

1.3.3 Delete Task

The delete task service is called by the system for a task that encounters a second abort condition during processing of an initial abort request. This service is asynchronously initiated by another task or by operator communications. If the OWNERNAME restriction is set in T.ACCESS, only a task of the same owner name can initiate the task delete request.

Asynchronous Delete

When a task needs to delete another task of the same owner name, it calls the asynchronous delete service. The task to be deleted can be in a ready-to-run state or a wait state, such as wait for execution signal, wait for resource, or wait for operation complete. In any case, the delete task bit is set in the DQE, and the task is linked to the ready-to-run list or to the memory request queue for inswap. An exception is made for a task already in the memory request queue. In this case, the task is not linked into the ready-to-run queue until memory scheduler processing is complete.

Synchronous Deletes

When the currently executing task encounters a delete condition, the delete task bit is set in the DQE. The CPU scheduler then processes the delete request. The following is an outline of synchronous delete processing.

Outstanding I/O

Delete processing causes all outstanding I/O to be terminated (killed).

Messages in Receiver Queue

All outstanding messages sent to a task being deleted are unlinked from the message receiver queue and treated as complete with abnormal status.

Outstanding Run Requests

If the task being deleted has outstanding run requests for other tasks, any call back is ignored.

Run Requests in Receiver Queue

If the task being deleted has requests in its run-receiver queue, the current run request is terminated and the appropriate status is posted in the run-request parameter block. If any additional run requests are queued, a new copy of the task is activated.

Abort Receiver

Abort receivers are not processed for tasks being deleted.

Open Files

Files associated with a task being deleted are not automatically closed.

Resources

All resources associated with a task being deleted are deallocated, and the task is no longer active in the system.

1.4 Scheduler-Debug Interface

Design Goals

The structure of the scheduler-debug interface is dictated by the following major design goals:

- MPXDB can be associated with a task at task activation time, or subsequently associated with a terminal task when the break key is struck. MPXDB can also be associated with a task dynamically through a system service call.
- When a task that has MPXDB associated with it is executing, two methods of entering MPXDB are provided: the executing task encounters a previously set MPXDB trap instruction, or the terminal operator presses the break key.
- Entering MPXDB mode by a trap or break is allowed during execution of software (task) interrupt receivers like message, end action, and break.
- MPXDB intercepts any task aborts, automatically enters the MPXDB mode, and informs the operator of the abort reason.
- System entry into the abort receiver is "soft" in that outstanding I/O requests are completed, and files remain open and allocated. This allows the operator the ability to correct and proceed from the environment that caused the abort condition.

Debug Entry Points

To accommodate the scheduler interface and achieve the MPXDB design goals, MPXDB is organized into five entry points. These entry points are reflected by an address table (HAT) structure at the beginning of the MPXDB program. When MPXDB is loaded, the address of the MPXDB HAT is stored in T.DBHAT in the TSA. The first word of the HAT contains the number of MPXDB entry points. Subsequent words contain the address of the individual MPXDB entry points. The entry points provided are:

Entry Point	Description
1	Debug start-up
2	Debug restart
3	Trap/break
4	User break exit
5	Abort

Task Interrupt Status

MPXDB examines a byte (DQE.ATI) in the dispatch queue entry and determines the status of task interrupts. When MPXDB is entered, DQE.ATI contains the definition of all active task interrupts.

Bit	Meaning
0	Reserved
1	Active end action interrupt 1 (DQE.AEA1)
2	Active debug mode interrupt (DQE.ADM)
3	Active user break interrupt (DQE.AUB)
4	Active end action interrupt 2 (DQE.AEA)
5	Active message interrupt (DQE.AMI)
6-7	Reserved

TSA Stack Pushdown Level Interpretation

For all MPXDB entry points except restart, the context associated with the most recently interrupted task level is contained in T.CONTXT. Nested levels of task interrupt are contained in the TSA stack. Unless one of the task interrupt levels (other than DQE.ADM) is active, the TSA stack is clean (empty) on entry to MPXDB. If task interrupts are active, the context storage in the TSA is in reverse order of priority. For example, highest priority is the most recent. In the active task interrupt bit assignments, bit zero is the lowest priority.

Exit from MPXDB Mode

When MPXDB is executing (regardless of entry point) the task is in the MPXDB mode. The MPXDB mode is exited by calling one of the following H.EXEC entry points:

Entry point	Description
H.EXEC,22	Go to specified task context
H.EXEC,23	Run user break receiver

1.4.1 Entry Point 1 - Start-up

This entry point is accomplished in one of two methods: MPXDB is activated with the user task, or the user task issues an SVC call to load and execute MPXDB.

MPXDB Activated with User Task

The program activation service that runs for the task being activated detects that MPXDB is to be activated with the task. After the task is loaded, a special service is called to load MPXDB. Once MPXDB is loaded, the service stores the normal start-up registers and PSD in an MPXDB context block in the TSA (T.CONTXT). The service then adjusts the stack in the TSA to enter MPXDB at the MPXDB start-up entry point. When MPXDB is entered the stack is clean, MPXDB mode is set, and T.CONTXT contains the user task start-up registers and PSD.

MPXDB Activated by Load and Execute (M.DEBUG) SVC

When the user task issues a load and execute MPXDB SVC, the system service loads MPXDB, stores the user's registers and PSD in T.CONTXT, sets MPXDB mode, and adjusts the TSA stack for entry at MPXDB's start-up entry point.

1.4.2 Entry Point 2 - Restart

This entry point is executed when MPXDB needs to terminate any outstanding I/O, discard any outstanding messages, and clear the TSA stack. An MPXDB restart is invoked by an MPXDB call to H.EXEC,24.

1.4.3 Entry Point 3 - Trap/Break

This entry point is executed when a hardware break or M.INT is received by the user task being debugged. It is also entered when a trap SVC is executed. On entry, T.CONTXT contains the interrupted context, and the MPXDB mode task interrupt flag is set.

1.4.4 Entry Point 4 - User Break Exit

This entry point is executed when the user task being debugged executes a break exit. A user task being debugged can only execute its break receiver by giving a break command to MPXDB. MPXDB in turn calls H.EXEC,23. Normal break receiver entry is reserved for MPXDB use when MPXDB is associated with a task. When MPXDB's user break exit entry point is entered, T.CONTXT contains the most recent level of pushdown from the TSA stack. The number of pushdowns in the TSA stack varies based on the number of active task interrupts like message and end action.

1.4.5 Entry Point 5 - Abort

This entry point is executed when an abort request is received for the user task and no user abort receiver has been specified. When the abort is received, the user task context is in T.CONTXT of the TSA. If a task interrupt like message or break receiver was in effect when the abort request was received, the TSA stack is at the associated level of pushdown. Otherwise, the TSA stack is clean.

Wait I/O Operation Status on Abort Receiver Entry

When the abort receiver is entered, any wait I/O operation is completed first. If an abort request is received for a task with wait I/O outstanding, abort processing is deferred until the wait I/O is complete. A service is provided by operator communications to terminate (kill) outstanding I/O requests associated with the specified task. When an I/O request is terminated, appropriate status is posted in the FCB.

No-wait I/O Operation Status on Abort Receiver Entry

When the abort receiver is entered, all no-wait I/O operations is complete. If an abort request is received for a task with no-wait I/O outstanding, abort processing is deferred until all no-wait I/O requests are complete. User end-action routine processing is

inhibited for no-wait I/O completions when the task is aborting. Task abort status is posted in the FCB.

File Status on Abort Receiver Entry

All user files remain open on entry to the task abort receiver.

Inhibit of Abort Receiver Entry

If an abort condition is detected during abort processing for a previously detected abort condition, all outstanding I/O is terminated (killed), no status is posted, abort receiver entry is inhibited, resources are deallocated, and the task is removed from the system.

Reuse of Abort Receiver

Privileged tasks can re-establish an abort receiver from within an abort receiver, allowing privileged tasks to enter their abort receiver more than once. Unprivileged tasks can establish a one-shot abort receiver, but are aborted if an attempt is made to re-establish this receiver.

1.5 Task Interrupts

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

Task Interrupt Priorities

Individual tasks operating in the MPX-32 environment can be organized to take advantage of the task unique software interrupt levels. Each task in the MPX-32 system has six levels of software interrupt:

Level Priority	Description
0	Reserved for operating system use
1	MPXDB
2	Break
3	End action
4	Message
5	Normal execution (run request)

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 MPXDB interrupt level is reserved for system use by tasks running in MPXDB mode. The end-action interrupt level is used for system postprocessing of no-wait I/O, message, or run requests. It also executes user-task specified end-action routines. The normal execution level is used for run-request processing and general base level task execution.

Task Interrupt Scheduling

Task interrupt processing is gated by the MPX-32 scheduler 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 a return is made to the user task execution area.

System Service Calls from Task Interrupt Levels

A task can utilize the complete set of system services from any task interrupt level. It is prohibited, however, from making a wait for any no-wait completion call (M.ANYW) from an end-action routine. It is illegal to issue an I/O request on any FCB that is busy or has postprocessing outstanding.

Task Interrupt Context Storage

When a task interrupt occurs, the 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.

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 level exit system service call. When the level active status is reset, any queued request is processed.

In addition, the following services can inhibit higher priority task interrupts:

M.ASYNCH	Resets the asynchronous task interrupt mode back to the default environment.
M.DSMI	Disables the task interrupts for messages sent to the calling task.
M.DSUB	Deactivates the user break interrupt and allows user breaks by the terminal break key to be acknowledged.
M.ENMI	Enables task interrupts for messages sent to the calling task.
M.ENUB	Activates the user break interrupt and causes further user breaks by the terminal break key to be ignored.
M.SYNCH	Causes message and task interrupts to be deferred until the user makes a call to M.ANYW, M.ASYNCH, M.EAWAIT, or M.WAIT. Any deferred task interrupts are processed when a lower level task interrupt calls the M.ANYW, M.EAWAIT, or M.WAIT services.

User Break Interrupt Receivers

A task can enable the break interrupt level by calling the M.BRK monitor 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 an M.INT service call that 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 monitor service to exit from the pseudointerrupt (break) level.

User End-Action Receivers

When a task issues a no-wait I/O, send message, or send run request, a user-task end-action routine address can be specified. If specified, the routine is entered at the end-action priority level from the appropriate system postprocessing 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	H.IOCS,34
Send message	M.XMEA
Send run request	M.XREA

All types of user end-action exits provide a return or a continue-wait for any option. An interrupt exit normally returns to the interrupted context. A task can issue a series of no-wait request calls followed by a wait for any completion service call from the base level. This wait service (M.ANYW) places the task in an interruptive wait state, allowing the execution of postprocessing and end-action routines associated with the no-wait call. The return or continue wait end-action exit options allow the exiting end-action routine to return to the point following the wait for any call or to continue the wait for any state.

Note: A task is prohibited from making a wait for any service call from an end-action routine.

User Message Receivers

A task can 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. On entry to the message interrupt receiver, register one contains the address of the queue entry (MRRQ) in memory pool. The receiver can call a service M.GMSGP to store the message in a user receiver buffer. No-wait I/O is permitted with the M.WAIT service. After appropriate processing, the message interrupt level can be reset by calling the M.XMSGR system service to exit from the message interrupt receiver.

User Run Receivers

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, the task load module name is used to identify the task to be executed. If a task of that load module name is currently active, the run request is queued from the DQE of the specified task. If the specified task is not active, it is first activated. When a task begins execution as the result of a run request, register one contains the address of the run-request queue entry. The receiver can call a service M.GRUNP to store the run parameters in a user-receiver buffer. After appropriate processing, the run-receiver task exits by calling the M.XRUNR system service. Any queued run requests are then processed.

User Abort Receivers

User abort receivers execute at the normal task execution (base) level. The user task establishes an abort receiver by calling the M.SUAR monitor service. If an abort condition is encountered during task operation, control is transferred to it. On entry, any active software interrupt level is reset, all outstanding operations or resource waits are complete, and all no-wait requests were processed. End-action routines associated with no-wait requests that completed while the abort was outstanding were not executed. Status bits reflecting this are posted in the appropriate FCBs and PSBs. Any files opened or resources allocated when the abort condition was encountered remain opened and/or allocated when the abort receiver is executed. The TSA stack is clean, and the context when the abort condition was encountered is stored in T.CONTXT. When the abort receiver is entered, register six contains a status byte reflecting task interrupt status when the abort condition was encountered.

Bit	Meaning if Set
24	N/A
25	N/A
26	User break interrupt active
27	End action interrupt active
28	Message interrupt active

The standard exit service is used to exit from an abort receiver. If another abort condition is encountered while a task is in an abort receiver, the task is deleted.

1.6 Send/Receive Facilities

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 that issue the requests are called sending task services.

1.6.1 Receiving Task Services

Establishing Message and Run Receiver Capability

Establishing Message Receivers -- 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.

Establishing Run Receivers -- Any valid task can be a run receiver. Although a set of special run receiver services are provided, in the most simple case they are not needed. 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. If a task of that load module name is currently active and is a single-copied task, the run request is queued until the task exits. If a task of that load module name is currently active but is not a single-copied task, the load module is activated (multicopied) to process the request. If a multicopied task is waiting for a run request, the task number is used to activate the load module to process the 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.

Execution of Message and Run Receiver Programs

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.

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.

Obtaining the Passed Parameters

Obtaining Message Parameters -- When the message receiver is entered, register one contains the address of the message queue entry in memory pool. The task can retrieve the message directly from memory pool or 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.

Obtaining the Run-request Parameters -- When the run receiver is entered, register one contains the address of the run request queue entry in memory pool. The task can retrieve the run request parameters directly from memory pool or 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 five word Parameter Receive Block (PRB) as an argument with the call.

Exiting the Receiver Program

Exiting the Message Receiver -- 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 also contains 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 exist, a return to the base level interrupted context is performed.

Exiting the Run Receiver Task -- When run-request processing is complete, the task uses 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.

Waiting for the Next Request

In addition to the wait options described under "Exiting the Receiver Program", a special message-wait call is provided. When operating at the base execution level, a task that has established a message receiver can invoke a service call (M.SUSP) to enter a wait state until the next message is received.

A task can also make use of the M.ANYW service from the base software level. The M.ANYW service is similar to M.SUSP. However, the M.SUSP wait state is ended only on 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.

1.6.2 Sending Task Services

Sending the Request

Message Send Service -- A task can send a message to another active task that has a message receiver established. The sending task must identify the destination task by task activation sequence 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 format allows for the specification of the message to be sent, any parameters to be returned, scheduler and user status, and the address of a user end-action routine. No-wait and no call back mode control options are also provided.

Send Run Request Service -- A task can send a run request to any active or inactive task, identifying the task by load module name or task number if the task is multicopied and waiting for a run request. When the run-request service (M.SRUNR) is called, the doubleword-bounded 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, and the address of a user end-action routine. No-wait and no call back mode control options are also provided.

Waiting for Request Completion

Waiting for Message Completion -- A message can be sent in 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 immediately after the request is queued. The operation in progress bit in the scheduler status field of the PSB is examined to determine completion. A sending task issues a series of no-wait mode messages followed by a call to the M.ANYW system wait service. This allows a task to wait for the completion of any no-wait mode messages previously sent. The completion of such a message causes resumption at the point after the M.ANYW call.

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.

End-action Processing

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

Run-request End-action Processing -- 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.

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.

Word	0 7	8 15	16 23	24 31
0	Load module or executable image name (PSB.LMN) or zero if activation (or task number (PSB.TSKN) if message or run request to multicopied task)			
1	Load module or executable image name, pathname vector, or RID vector if activation (or zero if message or run request to multicopied task)			
2	Priority (PSB.PRI)	Reserved	Number of bytes to be sent (PSB.SQUA)	
3	Reserved	Send buffer address (PSB.SBA)		
3	Return parameter buffer length in bytes (PSB.RPBL)		Number of bytes actually returned (PSB.ACRP)	
5	Reserved	Return parameter buffer address (PSB.RBA)		
6	Reserved	No-wait request end action address (PSB.EAA)		
7	Completion status (PSB.CST)	Processing start status (PSB.IST)	User status (PSB.UST)	Options (PSB.OPT)

Word 0

Bits 0-31 Load module or executable image name - contains characters one through four of the name of the load module or executable image to receive the run request or

Task number - contains the task number of the task to receive the message or the task number of the multicopied load module or executable image to receive the run request.

Word 1

Bits 0-31 Load module or executable image name - contains characters five through eight of the name of the load module or executable image to receive the run request, or zero if the message or run request is sent to multicopied load module or executable image.

Word 2

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

Bits 8-15 — Reserved

Bits 16-31 — Number of bytes to be sent - specifies the number of bytes to be passed (0 to 768) with the message or run request.

Word 3

Bits 0-7 — Reserved

Bits 8-31 — Send buffer address - contains the word address of the buffer containing the parameters to be sent.

Word 4

Bits 0-15 — Return parameter buffer length - contains the maximum number of bytes (0 to 768) that may be accepted as returned parameters.

Bits 16-31 — Number of bytes actually returned - set by the send message or run request service upon completion of the request.

Word 5

Bits 0-7 — Reserved

Bits 8-31 — Return parameter buffer address - contains the word address of the buffer where any returned parameters are stored.

Word 6

Bits 0-7 — Reserved

Bits 8-31 — No-wait request end-action address - contains the address of a user routine to be executed at a software interrupt level upon completion of the request.

Word 7

Bits 0-7 — Completion status - contains completion status information posted by the operating system as follows:

Bit	Meaning if Set
0	Operation in progress (PSB.OIP)
1	Destination task was aborted before completion of processing for this request (PSB.DTA)
2	Destination task was deleted before completion of processing for this task (PSB.DTD)

3	Return parameters truncated -- attempted return exceeds return parameter buffer length (PSB.RPT)
4	Send parameters truncated -- attempted send exceeds destination task receiver buffer length (PSB.SPT)
5	User end-action routine not executed because of task abort outstanding for this task (can be examined in abort receiver to determine incomplete operation) (PSB.EANP)
6-7	Reserved

Bits 8-15 Processing start (initial) status - contains initial status information posted by the operating system as follows:

Bit	Meaning if Set
0	Normal initial status (PSB.IST)
1	Message request task number invalid (PSB.TSKE)
2	Run request load module or executable image name not found (PSB.LMNE)
3	Reserved
4	File associated with run request load module or executable image name does not have a valid load module or executable image format (PSB.LMFE)
5	Dispatch Queue Entry (DQE) space is unavailable for activation of the load module or executable image specified by a run request (PSB.DQEE)
6	An I/O error was encountered while reading the directory to obtain the file definition of the load module or executable image specified in a run request (PSB.SMIO)
7	An I/O error was encountered while reading the file containing the load module or executable image specified in a run request (PSB.LMIO)
8	Memory unavailable
9	Invalid task number for run request to multicopied load module or executable image in RUNW state
10	Invalid priority specification. An unprivileged task can not specify a priority which is higher than its own execution priority (PSB.PRIE).
11	Invalid send buffer address or size (PSB.SBAE)
12	Invalid return buffer address or size (PSB.RBAE)
13	Invalid no-wait mode end action routine address (PSB.EAE)
14	Memory pool unavailable (PSB.MPE)
15	Destination task receiver queue is full (PSB.DTQF)

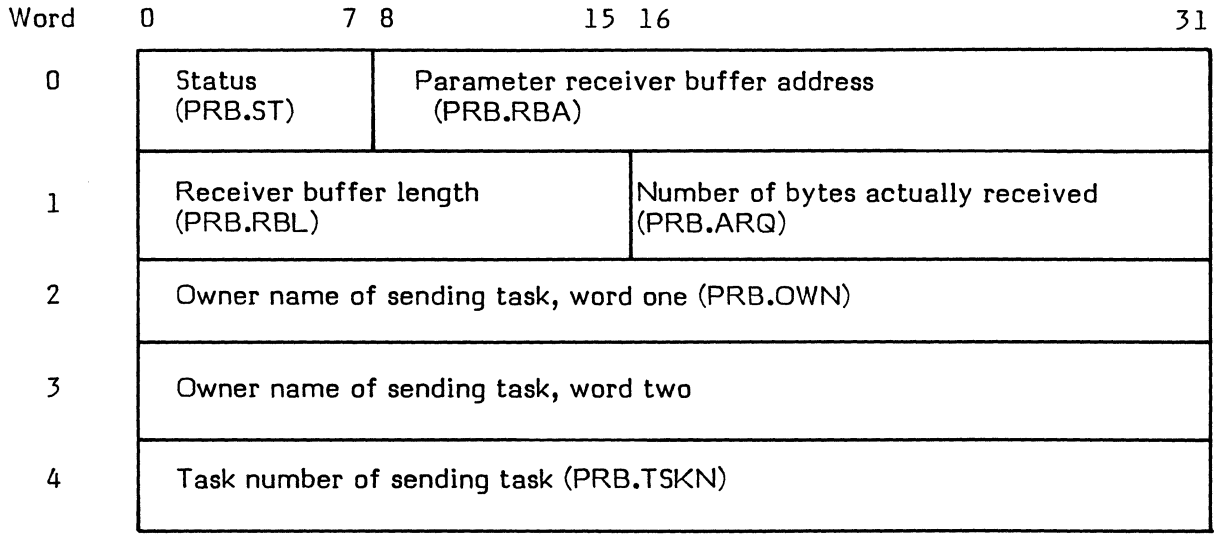
Bits 16-23 User status - defined by the destination task.

Bits 24-31 Options - contains user-request control specification as follows:

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

Parameter Receive Block (PRB)

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



Word 0

Bits 0-7 Status - contains status as follows:

Bit	Meaning if Set
0	Normal status
1	Invalid PRB address
2	Invalid receiver buffer address or size detected during parameter validation (PRB.RBAE)
3	No active send request (PRB.NSRE)
4	Receiver buffer length exceeded (PRB.RBLE)
5-7	Reserved

Bits 8-31 Parameter receiver buffer address - contains the word address of the buffer where any returned parameters are stored.

Word 1

Bits 0-15 Receiver buffer length - contains the length of the receiver buffer (0 to 768 bytes).

Bits 16-31 Number of bytes actually received - set by the operating system and is a maximum equal to the receiver buffer length.

Words 2,3

Bits 0-63 Owner name of sending task - set by the operating system to contain the owner name 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.

Receiver Exit Block (RXB)

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

Word	0 7	8 15	16 31
0	Return status (RXB.ST)	Return parameter buffer address (RXB.RBA)	
1	Options (RXB.OPT)	Reserved	Number of bytes to be returned (RXB.RQ)

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 - contains receiver exit control options as follows:

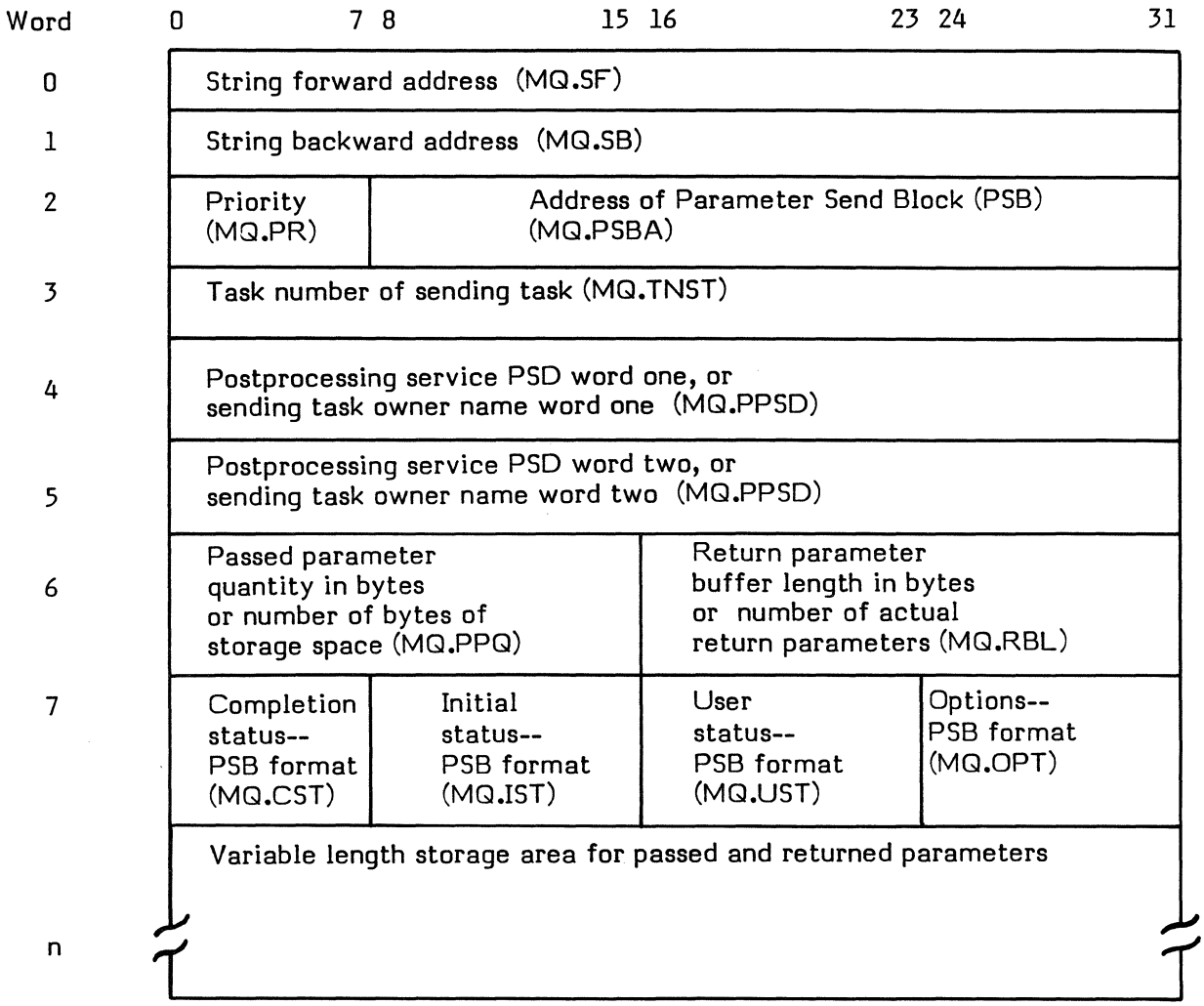
Value	Meaning
0	Wait for next run request (M.XRUNR) Return to point of task interrupt (M.XMSGR)
1	Exit task, process any additional run requests. If none exist, perform a standard exit (M.XRUNR) Not applicable for M.XMSGR

Bits 8-15 Reserved

Bits 16-31 Number of bytes to be returned - contains the number of bytes (0 to 768) to be returned on a message or receiver run exit.

Message or Run Request Queue Entry (MRRQ)

The Message or Run Request Queue Entry (MRRQ) is generated by the system to process a send request. After the MRRQ has been manufactured by the send service, it is attached to the appropriate queue slot in the DQE of the destination task. When the receiver program is entered, register one contains the address of the MRRQ in memory pool. The receiver program can reference the MRRQ directly, without issuing a M.GRUNP or M.GMSGP service call. The same format MRRQ is used for both messages and run requests.



Word 0

Bits 0-31 String forward address - contains the address of next entry of top-to-bottom list.

Word 1

Bits 0-31 String backward address - contains address of next entry in bottom-to-top list.

Word 2

Bits 0-7 Priority - contains the priority (1 to 64) of this request.

Bits 8-31 Address of Parameter Send Block (P^B) - contains the logical address of the PSB in the address space of the task which initiated the request.

Word 3

Bits 0-31 Task number of requesting task - contains the task activation sequence number of the task which issued the request.

Words 4-5

Bits 0-63 Postprocessing service PSD - contains the PSD of the appropriate postprocessing service for the task which issued the request or contains the sending task's owner name.

Word 6

Bits 0-15 Passed parameter quantity - contains the number of bytes sent to the destination task.

Bits 16-31 Return parameter buffer length - contains the length in bytes of the return parameter buffer in the task which issued the request.

Word 7

Bits 0-15 Scheduler status - contains status information to be posted in the scheduler status field of the PSB upon request completion. See PSB format.

Bits 16-23 User status - contains status as defined by the destination task.

Bits 24-31 Options - contains user request control specifications as follows:

Bit	Meaning if Set
24	Request is in no-wait mode
25	Request is in no call back mode (no wait, no status, no return parameters)

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.

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 time1 or M.SUSP taskno,time1	Wait for receipt of next message
Sender Services		
M.SRUNR psbaddr	M.SMSGR psbaddr	Send request
M.ANYW time1	M.ANYW time1 M.EAWAIT time1	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, or contains a negative number of time units to be used as a wait time-out value

1.7 Device Address Specification

Device addresses are specified using a combination of three levels of identification: device type, device channel/controller address, and device address/subaddress.

A device may 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 which results in allocation of the first available device of the type requested on the channel/controller specified.

The third method of device selection requires specification of the device type, channel/controller, and device address/subaddress. This method allows specification of a specific device.

Examples

Type 1 - Generic device class:

```
ASSIGN OUT TO DEV=M9
```

In this example, the file associated with logical file code OUT is allocated to any 9-track tape unit on any channel.

Type 2 - Generic device class and channel/controller:

```
ASSIGN OUT TO DEV=M910
```

In this example, the file associated with logical file code OUT is allocated to the first available 9-track tape unit on channel ten. The specification is invalid if a 9-track tape unit does not exist on the channel.

Type 3 - Specific device request:

```
ASSIGN OUT TO DEV=M91001
```

In this example, the file associated with logical file code OUT is allocated to the 9-track tape unit 01 on channel ten. The specification is invalid if unit 01 on channel ten does not exist or is not a 9-track tape.

GPMC/GPDC devices are specified in keeping with the general structure as defined. For instance, the CRT at subaddress 04 on GPMC 01 whose channel address is 20 would be identified as follows:

```
ASSIGN OUT TO DEV=TY2004
```

A special device type, NU, is available for null device specifications. Files accessed using this device type generate an end-of-file upon attempt to read and normal completion upon attempt to write.

Assignment of logical file codes to the operator console is achieved through usage of the device type CT.

A description of device selection possibilities are as follows:

Disc

DC	Any disc except memory disc
DM	Any moving head or memory disc
DM08	Any moving head disc on channel 08
DM0801	Moving head disc 01 on channel 08
DM0002	Memory disc 02 on channel 00
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

Card Equipment

CR	Any card reader
CR78	Any card reader on channel 78
CR7800	Card reader 00 on channel 78

Line Printer

LP	Any line printer
LP7A	Any line printer on channel 7A
LP7A00	Line printer 00 on channel 7A
LP7EA0	Serial printer A0 on ACM channel 7E

MPX-32 Device Type Codes

Dev Type Code	Device Mnemonic	Device Description
00	CT	Operator console (not assignable)
01	DC	Any disc unit except memory disc
02	DM	Any moving head or memory disc
03	DF	Any fixed head disc
04	MT	Any magnetic tape unit
05	M9	Any 9-track magnetic tape unit
08	CR	Any card reader
0A	LP	Any line printer
0B	PT	Any paper tape reader punch
0C	TY	Any teletype compatible device (other than console)
0D	CT	Operator console (assignable)
0E	FL	Floppy disc
0F	NU	Any null device
10	CA	Communications adapter
11	U0	User-supplied device
12	U1	User-supplied device
13	U2	User-supplied device
14	U3	User-supplied device
15	U4	User-supplied device
16	U5	User-supplied device
17	U6	User-supplied device
18	U7	User-supplied device
19	U8	User-supplied device
1A	U9	User-supplied device
1B	LF	Line printer/floppy controller (used only with SYSGEN)
N/A	ANY	Any nonfloppy disc except memory disc

1.8 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 aligibility. Execution priority is specified when a task is cataloged into the system. Execution eligibility is determined by the task's readiness to run.

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 to 54. Time-distribution tasks operate in the priority range 55 to 64.

Real-time Priority Levels (1 to 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 that interferes 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 can always be pre-empted by a higher priority real-time task that is ready to run.

Time-distribution Priority Levels (55 to 64)

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

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 pre-empted 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 the task's base execution priority level.

Situational Priority Increments

Time-distribution tasks are given situational priority increments in order to increase program 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 migrates 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
Pre-empted by real-time task	Level 55

Time-Quantum Controls

The MPX-32 system allows for the specification of two time-quantum values at system generation time. 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 pre-emption 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 is pre-empted and/or outswapped by any higher priority task. When a task's full time-quantum has expired, it relinks to the bottom of the priority list at base execution priority.

Time-quantum accumulation is the accumulated sum of actual execution times used by this task. A task's quantum accumulation value resets when the task voluntarily relinquishes CPU control; for example, suspend, wait I/O, etc.

State Chain Management

The current state of a task like ready to run or waiting for I/O is reflected by the linkage of the dispatch queue entry associated with the task into the appropriate state chain. 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.

MPX-32 State Queues

Ready-to-Run Queues

1. Current CPU task (in execution) - CURR
2. Current IPU task (in execution) - CIPU
3. IPU requesting state - RIPU
4. Real-time priority levels (1-54) - SQRT
5. Time-distribution priority level 55 - SQ55
6. Time-distribution priority level 56 - SQ56
7. Time-distribution priority level 57 - SQ57
8. Time-distribution priority level 58 - SQ58
9. Time-distribution priority level 59 - SQ59
10. Time-distribution priority level 60 - SQ60
11. Time-distribution priority level 61 - SQ61
12. Time-distribution priority level 62 - SQ62
13. Time-distribution priority level 63 - SQ63
14. Time-distribution priority level 64 - SQ64

Wait Mode Operation Queues

15. Wait mode interactive input - SWTI
16. Wait mode I/O - SWIO
17. Wait mode send message - SWSM
18. Wait mode send run request - SWSR
19. Wait mode low speed output (not implemented) - SWLO

Execution Wait Queues

20. Suspended waiting for message interrupt, timer expiration, or resume - SUSP
21. Waiting for run request or timer expiration - RUNW
22. Operator hold, waiting for continue - HOLD

Wait for Any Operation Complete Queue

23. Waiting for completion of any no-wait I/O, no-wait message, no-wait run request, or any message interrupt or break - ANYW

Waiting for Resource Queues

24. Waiting for disc space - SWDC
25. Waiting for peripheral device - SWDV
26. Reserved
27. Waiting for memory - MRQ
28. Waiting for memory pool - SWMP

1.9 FAT/FPT and Blocking Buffer Allocation

During the task allocation process, separate areas are reserved in a task's TSA for FAT/FPT pairs and blocking buffers. The size of each area is fixed for the duration of a task's execution. The size of the FAT/FPT area limits the number of file codes that a task can have allocated concurrently. The size of the blocking buffer area limits the number of file codes assigned to blocked devices or files that a task can allocate concurrently. The number of entries in each area is established as follows.

FAT/FPT Area

Nonshared task -- one FAT and FPT entry for each cataloged assignment, plus one entry for each TSM assignment that does not override a cataloged assignment, plus the number specified on the cataloger FILES directive.

Shared task -- the number specified on the cataloger FILES directive.

Blocking Buffer Area

Nonshared task -- from the assignments resulting from merging cataloger and TSM assignments, one buffer for each $ASSIGN; plus one buffer for each $ASSIGN to a magnetic tape or disc unit on which the unblocked option is not specified, plus one buffer for each $ASSIGN plus the number specified on the cataloger BUFFERS directive.

Shared task -- the number specified on the cataloger BUFFERS directive.

Cataloger and TSM $ASSIGN directives are modified by the addition of an unblocked specification as follows:

```
$ASSIGN lfc TO file BLOCKED=N
```

The following cataloger directives are added:

FILES number

number specifies the maximum number of dynamically allocated file codes that a nonshared task can allocate concurrently. It specifies the maximum number of file codes that a shared task can have allocated concurrently.

BUFFERS number

number specifies the maximum number of dynamically allocated file codes assigned to blocked files or devices that a nonshared task can allocate concurrently. It specifies the maximum number of file codes assigned to blocked files or devices that a shared task can allocate concurrently.

Files and buffers override parameters are specified to the Parameter Task Activation (M.PTSK) system service. These parameters allow addition of TSM FILES and BUFFERS directives if required by a future "load and go" capability.

1.10 Indirectly Connected Interrupts

An indirectly connected interrupt is an interrupt that is associated with an MPX-32 task. When the interrupt occurs, the associated task is resumed. An interrupt is declared as indirectly connected at system generation (SYSGEN) time. This declaration causes SYSGEN to generate an Indirectly Connected Task Linkage Block (ITLB). The ITLB is permanently associated with the specified interrupt level, but only becomes associated with an MPX-32 task when the M.CONN system service is invoked. A task can be disconnected from an interrupt level by invoking the M.DISCON system service.

Connect Task to Interrupt Service (M.CONN)

The M.CONN system service associates an MPX-32 task with an external interrupt that was declared at system generation time to be indirectly connected. When called, M.CONN is presented the priority level of the interrupt and the task activation sequence number (TASKNO) of the task. The TASKNO is first validated to ensure that it is currently active and has either the same owner name as the calling task, or the owner name of the calling task is privileged or is not restricted from access to tasks of a different owner. If so, the M.CONN service next checks to see if the specified task is already connected to an interrupt. DQE.ILN in the DQE contains the interrupt priority level if the task is already connected. If the task is not previously connected, the M.CONN service searches the Indirectly Connected Task Linkage Table (ITLT) to find the linkage block (ITLB) associated with this interrupt. If one exists and is not already connected, the DQE address of the task being linked is stored in word one of the ITLB to reflect the linkage. DQE.ILN in the DQE is set to contain the interrupt priority level.

Note: The task is automatically disconnected from the interrupt on abort, delete, or exit.

Disconnect Task from Interrupt Service (M.DISCON)

The M.DISCON system service disconnects an MPX-32 task from an external interrupt it had previously been connected to. When called, M.DISCON is presented the task activation sequence number (TASKNO) of the task as an argument with the call. If the specified task is not connected to an interrupt, DQE.ILN in the DQE will be equal to zero and the request will be ignored. Otherwise, DQE.ILN contains the external interrupt priority level. M.DISCON uses this priority level to locate the linkage block (ITLB) in the linkage table (ITLT). The DQE address (word one of the ITLB) is cleared to mark the level as disconnected. DQE.ILN is cleared in the DQE of the specified task.

Indirectly Connected Task Linkage Table (ITLT)

The Indirectly Connected Task Linkage Table (ITLT) is a variable length table built by the system generation program (SYSGEN). It contains an entry for each interrupt specified as being indirectly connectable. An entry is called an Indirectly Connected Task Linkage Block (ITLB) and is 24 words in length. The address of the ITLT is contained in C.ITLT. The number of entries in ITLT is contained in C.NITI. Both C.ITLT and C.NITI are initialized by SYSGEN.

Indirectly Connected Task Linkage Block (ITLB)

An entry in the indirectly connected task linkage table is called an Indirectly Connected Task Linkage Block (ITLB). An ITLB is 24 words long and is used to associate an external interrupt with an indirectly connected task.

Word 0 31

0	Priority level	[DATAW X'YY']
1	DQE address of indirectly connected program	[DATAW 0]
2	Old PSD word one	[DATAW 0]
3	Old PSD word two	[DATAW 0]
4	New PSD word one	[GEN 1/1, 12/0, 19/W($+2W)]
5	New PSD word two	[GEN 1/1, 14/0, 1/1, 1/0, 1/0, 14/0]
6	Increment global interrupt count instruction	[ABM 31,C.GINT]
7	Save register instruction	[STF R0,$+9W]
8	Branch and link to ICP routine	[BL ICP]
9	Address of register save area for S.EXEC5 call	[LA X2, $+7W]
10	Old PSD for S.EXEC5 call	[LD R6, $-8W]
11	Block external interrupts	[BEI]
12	Deactivate interrupt	[DAI X'YY']
13	Branch back for S.EXEC5 call	[BL ICP.20]
14 15	Reserved for future use	
16	Register save area	
23		

Word 0

Bits 0-31 — Priority level - set by SYSGEN to contain the priority level of the associated interrupt.

Word 1

Bits 0-31 — DQE address of indirectly connected program - contains the Dispatch Queue Entry (DQE) address of the task to be resumed on occurrence of this interrupt. Initially set equal to zero by SYSGEN. Initialized by M.CONN system service.

Words 2 to 3

Bits 0-63 — Old PSD - contains the old PSD slot of the interrupt control block. Used for storage of the PSD associated with the interrupted context. Initially set equal to zero by SYSGEN. The dedicated interrupt location (IVL) is initialized by SYSGEN to contain the address of word two of the ITLB.

Words 4 to 5

Bits 0-63 — New PSD - contains the new PSD slot of the interrupt control block to be used on occurrence of this interrupt. Causes execution to begin at ITLB word six in privileged mode, unblocked state, with old map status retained.

Word 6

Bits 0-31 — Increment global interrupt instruction - contains an add bit in memory instruction to increment the global interrupt count. Execution will begin at this location upon occurrence of the associated interrupt. It must be the first instruction executed in ICP. This location is initialized by SYSGEN to contain an ABM 31,C.GINT.

Word 7

Bits 0-31 — Save registers instruction - contains a store file instruction to save all eight registers in words 16 to 23 of the ITLB. This location is initialized by SYSGEN to contain an STF R0,$+9W.

Word 8

Bits 0-31 — Branch and link to ICP routine - executed after the register save instruction on occurrence of the associated interrupt. Transfers control to the single-copied ICP routine. This location is initialized by SYSGEN to contain a BL ICP.

Words 9 to 13 — Branch back for S.EXEC5 call returns control to this location after S.EXEC14 is called in the ICP routine. Set-up is made for exiting the interrupt; then control is transferred back to ICP for the S.EXEC5 call.

Words 14 to 15 — Reserved for future use

Words 16 to 23 — Register save area

Indirectly Connected Interrupt Program (H.ICP)

The Indirectly Connected Interrupt Program (H.ICP) is a single-copied routine that processes all indirectly connected external interrupts. It is entered in unblocked mode with the end address (+1W) of the linkage block (ITLB) in register zero. The global interrupt count will have been incremented within the ITLB and the registers from the interrupted context will have been stored in words 16 to 23 of the block. When ICP is entered, it checks ITLB word one to verify connection of the interrupt to an MPX-32 task. If the interrupt is not connected, it is ignored and ICP transfers back to the ITLB to exit the interrupt. If ITLB word one contains a DQE address, the associated task is resumed by calling S.EXEC14, which returns control to word nine of the ITLB. Set-up is made for exiting the interrupt within the ITLB; then execution is transferred back to ICP.20 for the S.EXEC5 exit.

```
H.ICP       TRR     R0,R1       ITLB END ADDRESS TO R1
            SUI     R1,8W       BACK UP TO DQE POINTER
            LW      R2,0,R1     GET DQE ADDR OF IND CONN TASK
            BCF     ZR,ICP.10   CONTINUE IF CONNECTED
            TRSW    R0          BRANCH BACK TO ITLB TO EXIT
ICP.10      BU      S.EXEC14    RESUME PROGRAM
ICP.20      BL      S.EXEC5
```

1.11 Miscellaneous System Macros

1.11.1 M.BACK

This macro backspaces the current address of a blocked file by the specified number of file or record marks.

Calling Sequence:

```
M.BACK  addr, [R] [,num]
```

addr — is the FCB address

R — specifies record. If not specified, the default is file.

num — is the number of record or file marks to be backspaced. The number specified must be word scaled, for example, one word for one record. If not supplied, the current contents of register four are used.

1.11.2 M.CALL

This macro generates a supervisor call instruction.

Calling Sequence:

```
M.CALL  name,num
```

name is the name of a system module

num is an entry point number (1,2,3,...) within the system module

1.11.3 M.CLSE

This macro marks a file closed to subsequent service. An end-of-file mark can be written and a rewind can be performed.

Calling Sequence:

```
M.CLSE addr , [EOF] , [REW]
```

addr FCB address

EOF specifies an end-of-file mark is to be written

REW specifies the file is to be rewound

1.11.4 M.DFCB

This macro creates a File Control Block (FCB) and sets the appropriate parameters and specifications common to I/O requests that are issued for the file.

Calling Sequence:

```
M.DFCB    label,lfc ,[count], [addr1], [addr2], [addr3],
          [NWT], [NER], [DFI],
          [NST], [RAN], [ASC]  [LDR]  [INT ]  [EVN][,556]
                        [BIN], [NLD], [PCK], [ODD][,800]
```

label is the ASCII character string to be used as the symbolic label for the address of the FCB

lfc is the one- to three-character ASCII string to be used as the logical file code in the FCB

count is the transfer count in bytes

addr1 is the data transfer address

addr2 is the error return address

addr3 is the random access address expressed as the hexadecimal block number (zero origin) relative to the base of the random access file

NWT is the no-wait I/O specification indicator

NER is the inhibit peripheral error processing indicator

DFI is the inhibit data formatting indicator

NST is the inhibit status testing indicator

RAN is the random access mode indicator

ASC or BIN is the forced ASCII or forced binary mode specification for read or punch operations performed when the file code for this file is assigned to a card reader

LDR or NLD is the skip leader or do not skip leader specification when the file code for this file is assigned to a paper tape reader/punch device

INT or PCK is the interchange or packed mode specification when the file code for this file is assigned to a magnetic tape device

EVN or ODD is the even or odd parity specification when the file code for this file is assigned to a magnetic tape device

556 or 800 is the 556 or 800 BPI tape density specification when the file code for this file is assigned to a magnetic tape device

1.11.5 M.DFCBE

This macro creates an expanded File Control Block (FCB) and sets the appropriate parameters and specifications common to I/O requests that are issued for the file.

Calling Sequence:

```
M.DFCBE   label, lfc , [count], [addr1], [addr2], [addr3], [NWT], [NER], [DFI],
          [NST], [RAN], [ASC|BIN] , [LDR|NLD], [INT|PCK] , [EVN|ODD] , [556|800] ,
          [addr4] , [addr5] , [addr6]
```

label is the ASCII character string to be used as the symbolic label for the address of the FCB

lfc is the one- to three-character ASCII string to be used as the logical file code in the FCB

count is the transfer count in bytes

addr1 is the data transfer address

addr2 is the wait I/O error return address

addr3 is the random access address expressed as the hexadecimal block number (zero origin) relative to the base of the random access file

NWT is the no-wait I/O specification indicator

NER is the inhibit peripheral error processing indicator

DFI is the inhibit data formatting indicator

NST is the inhibit status testing indicator

RAN is the random access mode indicator

ASC or BIN is the forced ASCII or forced binary mode specification for read operations performed when the file code for this file is assigned to a card reader

LDR or NLD is the specify skip leader or do not skip leader specification when the file code for this file is assigned to a paper tape reader/punch device

INT or PCK is the interchange or packed mode specification when the file code for this file is assigned to a magnetic tape device

EVN or ODD is the even or odd parity specification when the file code for this file is assigned to a magnetic tape device

556 or 800 is the 556 or 800 BPI tape density specification when the file code for this file is assigned to a magnetic tape device

addr4 is the no-wait I/O normal end-action service address

addr5 is the no-wait I/O error end-action service address

addr6 is the user-supplied blocking buffer

1.11.6 M.EIR

This macro is called by the resident system module's initialization entry points at entry. It stores register zero for later recall by M.XIR, the initialization entry point exit macro.

Calling Sequence:

```
M.EIR
```

1.11.7 M.FCBEXP

This macro defines a File Control Block (FCB) to be used for an execute channel program request.

Calling Sequence:

```
M.FCBEXP  label, lfc [, [cpaddr], [tout], [PCP], [NWI], [NST], [ssize], [sbuffer],
          [nowait], [nowaiterror], [waiterror], [psize], [ppciadr]
```

label	is the ASCII string to use as the symbolic label for the address of the FCB
lfc	is the logical file code, word 0, bits 8 to 31 of the FCB
cpaddr	the logical address of the channel program to be executed
tout	a time-out value specified in seconds
PCP	specifies physical channel program
NWI	specifies no-wait I/O request
NST	specifies status checking not requested
ssize	the size of the user-specified sense buffer
sbuffer	the address of the user-specified sense buffer
nowait	normal no-wait end-action return address
nowaiterror	no-wait end-action error return address
waiterror	wait end-action error return address
psize	size of PPCI status buffer to use
ppciaddr	PPCI end-action address

1.11.8 M.FWRD

This macro advances the current address of a blocked file by the number of file or record marks specified.

Calling Sequence:

```
M.FWRD  addr, [R] [,num]
```

addr	is the FCB address

R specifies record. If not specified, the default is file.

num is the number of record or file marks to be advanced, one word for one record.

1.11.9 M.INIT

This macro initializes device handler parameters through entry point eight. The code generated by this macro is executed by SYSGEN and overlayed.

Calling Sequence:

```
M.INIT  label , [NOP] [,SPA1, [SPA2]...[,SPA15]
```

label is the entry point truncated label; for example, MT0 for magnetic tape handler. This argument must be three ASCII characters. The first two represent the device mnemonic and the third is zero.

NOP specifies that TD 2000 level device status testing is not to be performed

SPA1-SPA15 are the SPA parameters to be initialized. A maximum of 15 parameters can be specified.

Usage:

```
M.INIT MT0,,SPA1,,SPA3
```

When placed as the last source statement in the device handler, this macro provides the necessary code to initialize the handler. The HAT must be modified to specify entry point eight and an additional entry must be made in the table (ACH MT00.8).

1.11.10 M.INITX

This macro is called by the handler initialization macros to combine basic instruction and commands with priority levels and device addresses for later execution within the handler. When this macro is called, register five must be preloaded with the properly positioned priority level or device address.

Calling Sequence:

```
M.INITX  cmd,mask
```

where:

cmd is the basic instruction or command

mask is a mask which is ORed with command

1.11.11 M.IOFF

This macro generates a Block External Interrupt (BEI) instruction that prevents the CPU from sensing all external interrupt requests generated by the I/O channel and RTOM.

Calling Sequence:

```
M.IOFF
```

1.11.12 M.IONN

This macro generates an Unblock External Interrupt (UEI) instruction that causes the CPU to sense all external interrupt requests generated by the I/O channel and RTOM.

Calling Sequence:

```
M.IONN
```

1.11.13 M.IPUOFF

This macro causes the IPU to be put off-line in software by setting bit C.IPUOFF.

Calling Sequence:

```
M.IPUOFF
```

1.11.14 M.IPUON

This macro causes the IPU to be put on-line in software by resetting bit C.IPUOFF.

Calling Sequence:

```
M.IPUON
```

1.11.15 M.IPURTN

This macro allows an IPU executable system module to return to the caller with registers preserved. The system service performs a register pop-up, except for those registers to be preserved, and returns to the location specified by the saved Program Status Word (PSW).

Calling Sequence:

```
M.IPURTN regn [,regn]...
```

regn — is a list of register numbers (0 to 7) identifying the registers to be preserved through the register pop-up. Any register not specified is not

1.11.16 M.IVC

This macro connects a handler entry point to an interrupt vector location.

Calling Sequence:

```
M.IVC  num,addr
```

num is the register number containing the interrupt level

addr is the handler entry point address label

1.11.17 M.KILL

This macro disables the CPU Halt Trap Processor (H.IPHT) and halts the system.

Calling Sequence:

```
M.KILL  addr
```

addr is the address of a four-character ASCII crash code

1.11.18 M.MODT

This macro builds an entry in the module address table.

Calling Sequence:

```
M.MODT  addr,num
```

addr is the address label of the module's HAT table

num is the module number

1.11.19 M.OPEN

This macro controls gating. It results in the termination of context switching being inhibited.

Calling Sequence:

```
M.OPEN
```

1.11.20 M.RTNA

This macro provides the facility to return to the caller from a system module to an address other than that specified by the saved PSW. It is used primarily for denial returns. It operates like the M.RTRN macro. The interrupt handler tests for the presence of an address specification in the parameter and replaces the saved Program Status Word (PSW) if an address is found.

Calling Sequence:

M.RTNA addr,regn [,regn]...

addr — is the register number of the register containing the address where return control resumes

regn-regn — is a list of register numbers (0 to 7) identifying the registers to be preserved through the register pop-up. Any register not specified is not preserved.

1.11.21 M.RTRN

This macro is the complement of M.CALL and allows a system module to return to the caller with registers preserved. The system service performs a register pop-up (except for those registers to be preserved) and returns to the location specified by the saved Program Status Word (PSW).

Calling Sequence:

M.RTRN regn [,regn]...

regn-regn — is a list of register numbers (0 to 7) identifying the registers to be preserved through the register pop-up. Any register not specified is not perserved.

1.11.22 M.SHUT

This macro is used for control gating purposes. It results in context switching being inhibited. This macro should not be used in a user task which is eligible for IPU execution. See M.USHUT.

Calling Sequence:

M.SHUT

1.11.23 M.SPAD

At each register push-down level, 22 scratchpad storage cells are provided for the use of re-entrant system modules. The scratchpad storage macro, M.SPAD, provides a convenient means of referencing the current level of scratchpad storage. The M.SPAD macro performs any memory reference operating on at least a word boundary (LW, STF, ARMD, DVMW), or any bit in memory operation (TBM, SBM, ABM, ZBM).

Calling Sequence:

M.SPAD mnem,reg,spad,index

mnem — is an instruction mnemonic defining the operation to be performed

reg — is the register number (0 to 7) or bit position (0 to 31) on which the operation is to be performed, or null

spad — is the scratchpad cell number (1 to 22) to be referenced by the operation

index — is an index register number (1, 2, or 3) that will be utilized in performing the operation

1.11.24 M.SVCP

This macro establishes any required protect bits in the high order byte of the SVC '1' table. A table is supplied containing 16 bit entries aligned on a halfword boundary. Each entry contains the SVC number in byte zero and the required protect bits in byte one. The following protect bits are defined:

Bit	Meaning if Set
0	Privileged SVC
1	IPU
2	Base mode tasks executable
3-7	Reserved

Calling Sequence:

```
M.SVCP  addr,num
```

addr — is the address of the data table

num — is the number of entries in the table

1.11.25 M.SVCP2

This macro establishes any required protect bits in the high order byte of the SVC '2' table. A table is supplied containing 16 bit entries aligned on a halfword boundary. Each entry contains the SVC number in byte zero and the required protect bits in byte one. The following protect bits are defined:

Bit	Meaning if Set
0	Privileged SVC
1	IPU executable
2	Base mode task executable
3-7	Reserved

Calling Sequence:

```
M.SVCP2  addr,num
```

addr — is the address of the data table

num — is the number of entries in the table

1.11.26 M.SVCT

This macro builds one entry in the SVC '1' table for each SVC type one defined in the calling module's prototype SVC table. Each one word entry contains the address of the corresponding SVC; i.e., the twentieth entry contains the address of the twentieth SVC.

Calling Sequence:

```
M.SVCT  addr,num
```

addr is the address label for the calling module's prototype SVC table

num is the number of SVC entries in the module's prototype SVC table

1.11.27 M.SVCT2

This macro builds one entry in the SVC '2' table for each SVC type two defined in the calling module's prototype SVC table. Each one word entry contains the address of the corresponding SVC. For example, the twentieth entry contains the address of the twentieth SVC.

Calling Sequence:

```
M.SVCT2 addr,num
```

addr is the address label for the calling module's prototype SVC table

num2 is the number of SVC entries in the module's prototype SVC table

1.11.28 M.TRAC

See Chapter 6 - System Trace.

1.11.29 M.TRPINT

This macro generates an entry in the trap vector table.

Calling Sequence:

```
M.TRPINT rpl,tcb
```

rpl is the hexadecimal trap priority level

tcb is the address of the trap context block of the user trap handler

1.11.30 M.TYPE

This macro types a user-specified message and performs an optional read on the system console teletype.

Calling Sequence:

```
M.TYPE outaddr,outcnt [, inaddr][,incnt]
```

outaddr is the output message address

outcnt is the output transfer count

inaddr is the input message address

incnt is the input transfer count

1.11.31 M.USHUT

This macro is used to inhibit context switching of a user task. It should be used in user tasks which are eligible for IPU execution. See M.SHUT.

Calling Sequence:

```
M.USHUT
```

1.11.32 M.XIR

This macro is called by the resident system module's initialization entry points right before they exit. It decrements the number of entry points in the calling module by one so the initialization entry point is no longer included, and returns to the SYSGEN processor.

Calling Sequence:

```
M.XIR  addr
```

addr is the address label of the module's HAT table

1.11.33 DCA.DATA

This macro is used within the SYSGEN entry point of F class handlers to reserve Device Context Area (DCA) space for the number of DCAs specified by the repeat (REPT) count. During SYSGEN execution, one DCA is initialized for each Unit Definition Table (UDT) entry containing the name of the handler. The unused DCAs and the code contained within the SYSGEN entry point are overlayed following execution.

Calling Sequence:

```
DCA.DATA  sbuf [,[xwrds] [,time0,...,timeF] ]
```

sbuf — specifies the sense buffer size (bytes) for automatic sense retrieval by the Extended I/O (XIO) common subroutines following an I/O error indication

xwrds — is the number of extra words to reserve for each DCA. If not specified, the standard DCA size is used.

time0-timeF — specifies the time-out value in seconds for each Input/Output Control System (IOCS) opcode, hexadecimal 0 through F; for example, open, rewind, read, write, etc. If not specified or if zero is specified, no time out is associated with the I/O request.

1.11.34 DCA.INI1

This macro is used within the SYSGEN entry point of F-class handlers to initialize areas of the Device Context Area (DCA), Controller Definition Table (CDT) and Unit Definition Table (UDT) associated with the particular handler. The code generated by this macro is overlayed following SYSGEN execution.

Calling Sequence:

```
DCA.INI1  hname [,[OPIN] ,[IOQCDT] [,COM] ]
```

hname — specifies the handler name, e.g., H.DCXIO for F class disc handlers

OPIN — specifies operator intervention is applicable for this handler

IOQCDT — specifies I/O queue entries are to be linked to the CDT. If not specified, I/O queue entries are linked to the UDT. Because many standard handlers assume I/O queue entries are linked a certain way, this parameter must be used with caution. This parameter is available to allow users flexibility when building handlers.

COM — specifies the handler interfaces with the XIO common subroutines

1.11.35 DCA.INI2

This macro is used within the SYSGEN entry point of F-class handlers to restore the working environment within the SYSGEN entry point following any user added executable code.

Calling Sequence:

```
DCA.INI2
```

1.11.36 HMP.INIT

This macro initializes multiplexed I/O processor (MIOP) device handler parameters with entry point eight. The code generated by this macro is executed by SYSGEN and overlayed.

Calling Sequence:

```
HMP.INIT  label
```

label — is the entry point truncated label; for example, AS0 for the asynchronous communications handler. This argument must be three ASCII characters. The first two represent the device mnemonic and the third is zero.

1.11.37 IB.INIT

This macro initializes multiplexed I/O processor (MIOP) device handler parameters with entry point eight, where register seven contains the Controller Definition Table (CDT) address and register two contains the address of the current context block.

Calling Sequence:

```
IB.INIT
```

1.12 Extended MPX-32 Macros

The extended MPX-32 macros allow existing user modules and service routines to run in the extended mode. These macros select the appropriate coding, extended or nonextended, for a task by testing the state of the Macro Assembler option 16. (For example, if the MBR_DEF macro is specified, the coding for a DEF or SDEF directive is supplied depending on the state of option 16.)

The following are extended macros that directly replace the corresponding Macro Assembler directive:

Macro	Assembler directive
MBR_BEQ	BEQ
MBR_BGE	BGE
MBR_BGT	BGT
MBR_BL	BL
MBR_BLE	BLE
MBR_BLT	BLT
MBR_BNE	BNE
MBR_BNS	BNS
MBR_BS	BS
MBR_DEF	DEF
MBR_EXT	EXT
MBR_TRSW	TRSW

For descriptions of these macros, see the corresponding Macro Assembler directive description in the MPX-32 Utilities Manual.

The following macros do not have corresponding Macro Assembler directives, and must be placed within the code that is to operate in the extended mode:

Macros	Description
MBR_DBG	Calls the system debugger
MBR_DSCT	Directs data into the DSECT

Macros	Description
MBR_ENT	Generates the adaptation sequence required to reference a routine from a nonextended module
MBR_INIT	Tests the state of Macro Assembler option 16
MBR_SSCT	Returns to a local code section in the system section (SSECT) area after an MBR_DSCT has been specified

1.12.1 MBR_DBG (Calls to System Debugger) Macro

The MBR_DBG macro calls the system debugger from the target extended module. This macro references the system debugger extended code entry within the adaptation code sequence.

Syntax: MBR_DBG <symbol>

1.12.2 MBR_DSCT (DSECT Data Separation) Macro

The MBR_DSCT macro specifies that the following code is data, and directs the data into the DSECT. All data and variable constants must have been separated for inclusion in the DSECT section.

Syntax: MBR_DSCT <symbol>

1.12.3 MBR_ENT (Extended Code Routine Entry) Macro

The MBR_ENT macro generates the adaptative sequence required to reference a routine from a nonextended module. Each entry point must have an MBR_ENT macro before the first instruction.

Syntax: MBR_ENT <symbol> Replaces <symbol> EQU $

1.12.4 MBR_INIT (Module Initialization) Macro

The MBR_INIT macro tests the state of option 16. If option 16 is set, MBR_INIT initializes the code location to SSECT EXT_MPX. This macro is required after the program statement of an extended module.

Syntax: MBR_INIT

1.12.5 MBR_SSCT (System Code Separation) Macro

The MBR_SSCT macro specifies that the following code is executable data, and returns to a local code section in the system section (SSECT) area after an MBR_DSCT macro has been specified.

Syntax: MBR_SSCT

Usage:

```
          MBR_DSCT                          SEND DATA TO DATA SECTION
J.MOUNT   DATAD    C'J. MOUNT    '          SYSTEM MOUNT TASK
OPCOM     DATAD    C'OPCOM       '          OPERATOR COMMUNICATIONS TASK
SYS.LFC   DATAW    X'00AA532A    '  SYSTEM LFC '?T*'          (H.TAMM)
LFC3      DATAW    G'(3)         '  LFC FOR SYSTEM FCB3       (H.VOMM)
J.ATAPE   DATAD    C'J.ATAPE     '  ANSI TAPE HANDLER TASK
          LPOOL
          MBR_SSCT                          RETURN TO CODE SECTION
```

CHAPTER 2

SYSTEM TABLES AND VARIABLES

This chapter contains descriptions and format layouts for the tables and variables used by the MPX-32 operating system.

The MPX-32 table structure consists of the following categories:

Batch processing data area which contains the following:

- Job table
- Link file format (batch SLO and SBO)
- Run request format (J.SOEX)
- Spooled file directories

Executive (H.EXEC) data area which contains the following:

- Central Processing Unit (CPU)
- Dispatch Queue Entry (DQE)
- Dispatch Queue Address Table (DAT)

Input/output data area which contains the following:

- Blocking buffer control cells
- Controller Definition Table (CDT)
- Device Context Area (DCA)
- File Assignment Table (FAT)
- File Control Block (FCB)
- File Pointer Table (FPT)
- I/O Queue (IOQ) entry
- I/O table linkages
- Type Control Parameter Block (TCPB)
- Unit Definition Table (UDT)
- XIO Channel Definition Table (CHT)

Memory Management data area which contains the following:

- Map Image Descriptor List (T.MIDL)
- Memory Allocation Table (MATA)
- Memory Attribute List (T.MEML)
- Memory pool management
- Shared Memory Table (SMT)

Resource Management data area which contains the following:

- Allocated Resource Table (ART)
- Device Type Table (DTT)
- Mounted Volume Table (MVT)
- Resource Inquiry Table (M.RIQ)
- Resource Requirement Summary (RRS) entries
- Task Service Area (TSA)
- Volume Assignment Table (VAT)

Status Management data area which contains the following:

- Caller Notification Packet (CNP)

Terminal Services data area which contains the following:

- Terminal Line Buffer

Volume Management data area which contains the following:

- Bad Block Descriptor (M.BB.DEQ)
- Descriptor Allocation Map Descriptor (M.DM.DEQ)
- Descriptor Map (DMAP) Deallocation File Descriptor (M.BD.DEQ)
- Descriptors Descriptor (M.DD.DEQ)
- Directory Descriptor (M.DI.DEQ)
- Directory Entry Table (M.DN.TEQ)
- DQE Address Table (DAT)
- Memory Partition Descriptor (M.ME.DEQ)
- Resource Create Block (RCB)
- Resource Descriptor (M.RDCOM)
- Resource Descriptor Space Definition (M.RDSPD)
- Resource Logging Block (RLB)
- Space Allocation Map Descriptor (M.SM.DEQ)
- Space Map (SMAP) Deallocation File Descriptor (M.BS.DEQ)
- System Master Directory (SMD)
- Volume Descriptor (M.VO.DEQ)

Disc resident structures are:

- Volume format
- Load module structure
- Load module preamble
- Nonshared executable image structure
- Nonshared executable image preamble
- Shared executable image structure
- Shared executable image preamble
- Shared image descriptor

The resident system memory layout and utilization structure is described first.

The communications region is described next.

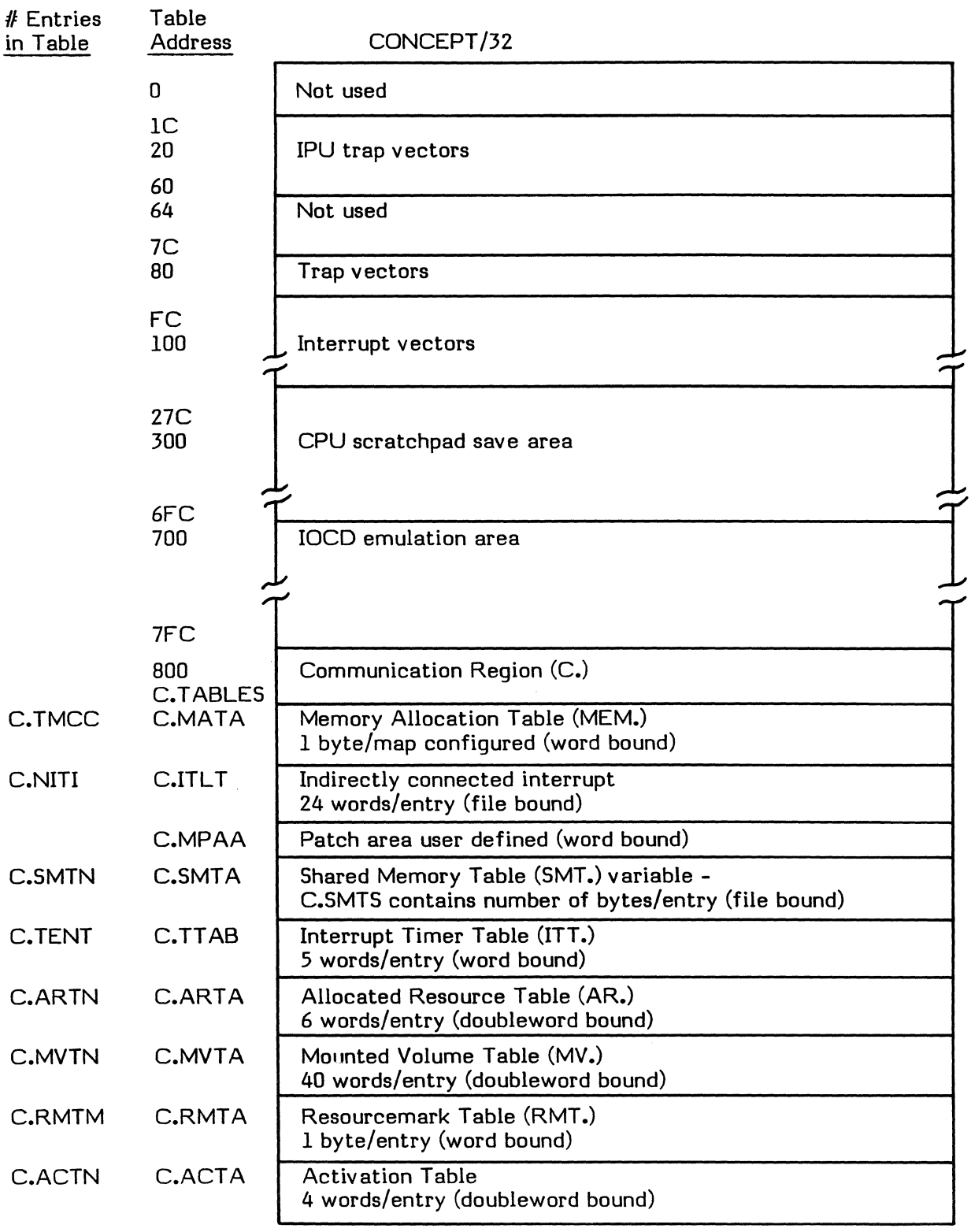
The table formats are then described, arranged in alphabetical order by the table name.

The disc resident resource descriptors are described in alphabetical order after the tables.

The disc resident structures are described last.

2.1 Memory Layout

Resident system memory layout and utilization structures are described below.



C.SEQN	C.SEQA	Sequence Table 4 words/entry (doubleword bound)
C.NQUE	C.DQUE	Dispatch Queue (DQE.) 54 words/entry (file bound)
	C.ADAT+1W	DQE Address Table 1 word/DQE (word bound)
C.CDTN	C.CDTA	Controller Definition Table (CDT.) 24 words/entry (word bound)
C.UDTN	C.UDTA	Unit Definition Table (UDT.) 16 words/entry (word bound)
C.DTTN	C.DTTA	Device Type Table (DTT.) 2 words/entry (doubleword bound)
C.CHTN	C.CHTA	Channel Definition Table (CHT.) 40 words/entry (file bound)
	C.MPL	Master process list 2 words/entry (doubleword bound)
	C.MIDL	Map image list for operating system 1 halfword/operating system map (doubleword bound)
	C.SPAD	CPU scratchpad image 256 words (word bound)
C.SVTN	C.SVTA	SVC Type 1 Vector Table 128 words or user defined (word bound)
	C.SVTA2	SVC Type 2 Vector Table 128 words (word bound)
	C.MIOP	GPMC Jump Table 16 words (file bound)
C.MODN	C.MODD	Module Address Table 16 words (word bound)
Start of resident code. All programs are file bound.		Trap processors
		Interrupt processors
		I/O processors
		System modules: H.ALOC, H.BKDM, H.EXEC, H.FISE, H.IOCS, H.MEMM, H.MONS, H.MVMT, H.REMM, H.REXS, H.SOUT, H.TAMM, H.TSM, H.VOMM
		User operating system resident modules and tasks, if any
		System debugger (H.DBUG1)
		Swapper (H.SWAPR)
	C.SBUF	Memory pool area
End of resident system		
	C.TSAD	Starting logical address of task's TSA
		User task space

2.2 Communications Region

The communications region is an area of main memory reserved for use by MPX-32 for storage of common data. This data is referenced by symbols that are equated to absolute memory locations. Together with each symbol is the length of the variable associated with the symbol. The length is in units, which is also the minimum boundary on which the variable resides.

Bit variables are contained in a set of contiguous words with the symbol C.BIT equated to the address of the first word. Bit variables are equated to bit positions relative to C.BIT. Bit variables are referenced by a combination of the variable symbol and C.BIT; for example, TBM C.AFLK,C.BIT.

Word # (Decimal)	Byte (Hex)	0 7 8 15 16 23 24 31
512-513	800	C.DATE
514	808	C.CAL
515	80C	C.INTC
516-517	810	C.TIME
518-519	818	C.LODC
520-521	820	C.SYMTAB
522-523	828	C.PODC
524-525	830	C.SBUF
526-527	838	C.SIDV
528	840	C.TMAC | C.EMAC
529	844	C.HMAC | C.SMAC
530	848	C.TMCC | C.EMCC
531	84C	C.HMCC | C.SMCC
532-533	850	C.SYSTEM
534-537	858	C.SYPATH
538-539	868	C.PCHFLE
540-541	870	C.TRACE
542-543	878	C.DBGLM
544-557	880	C.SPRDW
558-560	8B8	C.CIPU
561-563	8C4	C.RIPU
564-566	8D0	C.FREE
567-569	8DC	C.PREA
570-572	8E8	C.CURR
573-575	8F4	C.SQRT
576-578	900	C.SQ55
579-581	90C	C.SQ56
582-584	918	C.SQ57
585-587	924	C.SQ58
588-590	930	C.SQ59
591-593	93C	C.SQ60
594-596	948	C.SQ61
597-599	954	C.SQ62
600-602	960	C.SQ63
603-605	96C	C.SQ64
606-608	978	C.SWTI
609-611	984	C.SWIO
612-614	990	C.SWSM
615-617	99C	C.SWSR
618-620	9A8	C.SWLO
621-623	9B4	C.SUSP
624-626	9C0	C.RUNW
627-629	9CC	C.HOLD
630-632	9D8	C.ANYW
633-635	9E4	C.SWDC
636-638	9F0	C.SWDV
639-641	9FC	C.SWFI
642-644	A08	C.MRQ
645-647	A14	C.SWMP
648-650	A20	C.SWGQ

		0 7 8 15 16 23 24 31
651-671	A2C	C.SPCH
672	A80	C.TSAD
673	A84	C.ACTSEQ
674	A88	C.ADAT
675-676	A8C	C.BIT
677	A94	C.CDTA
678	A98	C.CPRI
679	A9C	C.DQUE
680	AA0	C.DTTA
681	AA4	C.FADR
682	AA8	C.FGONR
683	AAC	C.GINT
684	AB0	C.IDLA
685	AB4	C.IDLC
686	AB8	C.ITLT
687	ABC	C.BATSEQ
688	AC0	C.JOBN
689	AC4	C.MGRAN
690	AC8	C.MIDL
691	ACC	C.MIOP
692	AD0	C.MODD
693	AD4	C.MPL
694	AD8	C.MSD
695	ADC	C.MTIM
696	AE0	C.NTIM
697	AE4	C.PATCH
698	AE8	C.POOL
699	AEC	C.SGOS
700	AF0	C.SICTD
701	AF4	C.SMTA
702	AF8	C.ARTA
703	AFC	C.SPAD
704	B00	C.SVTA
705	B04	C.SVTA2
706	B08	C.SWAP
707	B0C	C.SYCS
708	B10	C.TSKN
709	B14	C.TSMDQA
710-717	B18	C.TTBT
718	B38	C.UDTA
719	B3C	C.TTAB
720	B40	C.MATA
721	B44	C.MPAA
722	B48	C.MPAC
723	B4C	C.MPAH
724	B50	C.RMTA
725	B54	C.EMTA
726	B58	C.REV
727	B5C	C.DEBUG
728	B60	C.TDQ1
729	B64	C.TDQ2
730	B68	C.TDQ3
731	B6C	C.REGS

		0 7 8 15 16 23 24 31			
732	B70	C.MVTA			
733	B74	C.ACTA			
734	B78	C.SEQA			
735	B7C	C.SCDIPU			
736	B80	C.CHTA			
737	B84	C.ETLOC			
738	B88	C.ADMASK			
739	B8C	C.IDLA1			
740	B90	C.IDLC1			
741	B94	C.IPUIT1			
742	B98	C.IPUIT2			
743	B9C	C.BTIME			
744	BA0	C.BDATE			
745	BA4	C.TCORR			
746	BA8	C.FSSP			
747	BAC	C.DPTIMO			
748	BB0	C.MDTA			
749	BB4	C.MDTE			
750	BB8	C.SWPLIM			
751	BBC	C.PDQE			
752-759	BC0	C.MPXBR			
760	BE0	C.MPXBRD			
761	BE4	C.IP00			
762-763	BE8	C.PSDBRE			
764-765	BF0	C.PSDBRX			
766-767	BF8	C.PSDMSE			
768-769	C00	C.PSDMSX			
770-771	C08	C.PSDEAE			
772-773	C10	C.PSDEAX			
774	C18	C.DSECT			
775	C1C	C.ADAPT			
776	C20	C.TDEFA			
777	C24	C.SWIOCL			
778	C28	C.CRDUMP			
779	C2C	C.HSTADR			
780	C30	C.CDTN		C.ITRS	
781	C34	C.SMVTI		C.SVTN	
782	C38	C.UDTN		C.RMTM	
783	C3C	C.EMTM		C.NOS	
784	C40	C.NRST		C.MVTN	
785	C44	C.ARTN		C.CHTN	
786	C48	C.HIMAP		C.SVTN2	
787	C4C	C.MDTN		C.MDTAV	
788-789	C50	C.RMS			
790	C58	C.GSLEMC			
791	C5C	Reserved			
792-793	C60	C.BDBUG			
794-810	C68	C.SPRH			
811	CAC	C.DPTRY		C.SMAPS	
812	CB0	C.BPRI	C.DTTN	C.FSFLGS	C.MODN
813	CB4	C.NITI	C.NQUE	C.RRUN	C.SMTN
814	CB8	C.TSMCNT	C.TSMPRI	C.TSMTOT	C.TENT
815	CBC	C.RMTL	C.EMTL	C.CONF	C.MACH
816	CC0	C.ACTN	C.DBTLC	C.SMTS	C.SEQN

		0 7 8 15 16 23 24 31			
817	CC4	C.IPUHIS			
818	CC8	C.SHRHI		C.SHRLO	
819	CCC	C.MAXSWP	C.SPB1		
820-821	CD0	C.SPB1 (cont.)			
822	CD8	C.MMSG	C.MRUN	C.MNWI	C.GSLEGI
823	CDC	C.GSLEPR	C.ADAFL	C.TKILL	C.DELTA
824	CE0	C.MPXBRN	C.DBMAPS	C.SWAPSZ	
825	CE4	C.DTSAVE			
826	CE8	C.SHCPU			
827	CEC	C.SHIPU			
828	CF0	C.UPDT			
829	CF4	C.SWPBUF			
830	CF8	C.MRQTMR			
831	CFC	C.SPARE			
832	D00	C.TABLES			
833	D04	C.USER			

Byte (Hex)	Symbol	Description
800	C.DATE	Current date (Gregorian) as input by operator
808	C.CAL	Calendar devices:

Byte	Description
0	Current century in binary (C.CENT)
1	Current year in binary (C.YEAR)
2	Current month in binary (C.MONTH)
3	Current day in binary (C.DAY)

80C	C.INTC	Interrupt counter (number of interrupts from zero which is midnight) used for time-of-day calculations
810	C.TIME	The system start-up values from C.BTIME and C.BDATE
818	C.LODC	The system listed output device used as a default in operator communications commands:

Byte	Description
0-1	ASCII device type code
2-3	ASCII channel number
4-5	ASCII subaddress
6-7	Reserved

820	C.SYMTAB	Name of the symbol table file
828	C.PODC	The system punched output device used as a default in operator communications commands:

Byte	Description
0-1	ASCII device type code
2-3	ASCII channel number
4-5	ASCII subaddress
6-7	Reserved

830	C.SBUF	First word contains address of memory pool. Second word is set by S.REMM21 to the number of words in memory pool.
838	C.SIDV	The system input device used as a default in operator communications commands:

Byte	Description
0-1	ASCII device type code
2-3	ASCII channel number
4-5	ASCII subaddress
6-7	Reserved

840	C.TMAC	Total count in halfwords of all E, H, and S memory modules available
	C.EMAC	Total count of valid E type memory modules available
844	C.HMAC	Total count of valid H type memory modules available
	C.SMAC	Total count of valid S type memory modules available
848	C.TMCC	Total count in halfwords of all E, H, and S memory modules configured (minus three for the unmapped portion of the debugger, if the system debugger is present)
	C.EMCC	Total count of valid E type memory modules configured (minus one if swap device is E-class and extended memory is present in the system, and minus any E-class map blocks allocated for the unmapped portion of the system debugger)
84C	C.HMCC	Total count of valid H type memory modules configured (minus any H-class map blocks allocated for the unmapped portion of the system debugger)
	C.SMCC	Total count of valid S type memory modules configured (minus any S-class map blocks allocated for the unmapped portion of the system debugger)
850	C.SYSTEM	Name of current system image
858	C.SYPATH	System pathname prototype
868	C.PCHFLE	Patch file name
870	C.TRACE	System trace (M.TRAC) control word
878	C.DBGLM	Debugger load module name
880	C.SPRDW	Reserved
8B8	C.CIPU	Standard format linked list head cell for all IPU tasks ineligible for CPU control, waiting in general queue. C.CIPU is the first of a set of communications region variables which are contiguous in memory. These variables, listed in the order that they appear in memory, are as follows:

C.CIPU
C.RIPU
C.FREE
C.PREA
C.CURR
C.SQRT
C.SQ55
C.SQ56
C.SQ57
C.SQ58
C.SQ59
C.SQ60
C.SQ61
C.SQ62
C.SQ63
C.SQ64

C.SWTI
C.SWIO
C.SWSM
C.SWSR
C.SWLO
C.SUSP
C.RUNW
C.HOLD
C.ANYW
C.SWDC
C.SWDV
C.SWFI
C.MRQ
C.SWMP
C.SWGQ
C.SPCH

8C4	C.RIPU	Standard format linked list head cell for all IPU tasks ready to run, waiting in general queue
8D0	C.FREE	Standard format linked list head cell for free entries in the CPU dispatch queue
8DC	C.PREA	Standard format linked list head cell for CPU dispatch queue entries that are in the preactivation state
8E8	C.CURR	Standard format linked list head cell for the CPU dispatch queue entry of the currently executing task. This list can have a maximum of two entries: one for the current real-time task (if any) and one for the current time-distribution task (if any).
8F4	C.SQRT	Standard format linked list head cell for the list of ready-to-run real-time (priority level 1 to 54) tasks
900	C.SQ55	Standard format linked list head cell for the list of ready-to-run priority level 55 time-distribution tasks
90C	C.SQ56	Standard format linked list head cell for the list of ready-to-run priority level 56 time-distribution tasks
918	C.SQ57	Standard format linked list head cell for the list of ready-to-run priority level 57 time-distribution tasks
924	C.SQ58	Standard format linked list head cell for the list of ready-to-run priority level 58 time-distribution tasks
930	C.SQ59	Standard format linked list head cell for the list of ready-to-run priority level 59 time-distribution tasks
93C	C.SQ60	Standard format linked list head cell for the list of ready-to-run priority level 60 time-distribution tasks
948	C.SQ61	Standard format linked list head cell for the list of ready-to-run priority level 61 time-distribution tasks

954	C.SQ62	Standard format linked list head cell for the list of ready-to-run priority level 62 time-distribution tasks
960	C.SQ63	Standard format linked list head cell for the list of ready-to-run priority level 63 time-distribution tasks
96C	C.SQ64	Standard format linked list head cell for the list of ready-to-run priority level 64 time-distribution tasks
978	C.SWTI	Standard format linked list head cell for all tasks waiting for the completion of wait mode interactive (terminal) input
984	C.SWIO	Standard format linked list head cell for all tasks waiting for the completion of wait mode I/O requests
990	C.SWSM	Standard format linked list head cell for all tasks waiting for the completion of wait mode send message request
99C	C.SWSR	Standard format linked list head cell for all tasks waiting for the completion of wait mode send run request
9A8	C.SWLO	Standard format linked list head cell for all tasks waiting for the completion of low speed output
9B4	C.SUSP	Standard format linked list head cell for all tasks that are in an execution suspend mode, waiting for a message interrupt, a timer expiration, or a resume task request
9C0	C.RUNW	Standard format linked list head cell for all tasks that are ineligible for CPU control, waiting for a run request to be received, or for the expiration of a timer
9CC	C.HOLD	Standard format linked list head cell for all tasks that are ineligible for CPU control, waiting for a continue request to be received
9D8	C.ANYW	Standard format linked list head cell for all tasks that are ineligible for CPU control, waiting for the completion of any no-wait mode I/O request, any no-wait mode send message request, any no-wait mode send run request, or any message or break interrupt
9E4	C.SWDC	Standard format linked list head cell for all tasks ineligible for CPU control, waiting for disc space to become available
9F0	C.SWDV	Standard format linked list head cell for all tasks ineligible for CPU control, waiting for a peripheral device to become available
9FC	C.SWFI	Reserved

A08	C.MRQ	Standard format linked list head cell for all tasks ineligible for CPU control, waiting for memory to become available
A14	C.SWMP	Standard format linked list head cell for all tasks ineligible for CPU control, waiting for memory pool to become available
A20	C.SWGQ	Standard format linked list head cell for all tasks ineligible for CPU control, waiting in general queue
A2C	C.SPCH	Reserved
A80	C.TSAD	Contains the address of the TSA for the currently executing task
A84	C.ACTSEQ	Running count of task activations, used to form right-most 24 bits of task number when a task is activated. SYSGEN initializes this word to zero
A88	C.ADAT	Address of the DQE address table (DAT)
A8C	C.BIT	Symbol associated with the beginning of the bit variables:

Bit	Meaning if Set
0	Accounting file lock indicator (C.AFLK)
1	Swap volume is E-class disc (C.ESWAP)
2	Dump real-time tasks on abort (C.FGPM)
3	Indicates user is using CPU scratchpad for his own needs. IPL alters SPAD locations defined by SYSGEN. Reset indicates SPAD locations not defined by SYSGEN are to be set to zero. (C.SPADOK)
4	List patches indicator (C.LSPT)
5	On-line restart in progress (C.RSTRT)
6	Reserved
7	Continuous batch mode indicator (C.SCBT)
8	J.SOUT banner page inhibit (C.SIBP)
9	SSIN device density is 556 (7-track) (C.SIDD)
10	SSIN device parity is odd (7-track) (C.SIDP)
11	Inhibit context switching in IPU (C.ICSIPU)
12	Task context switch inhibited (C.CSWI)
13	Activation from tape (C.TAPACT)
14-15	Reserved
16	Inhibit magnetic tape mount message (C.SIMM)
17	Memory error detected by H.IP02 (C.MERR1)
18	Memory parity error detected during memory initialization (C.MERR2)
19	Nonpresent memory detected (C.MERR3)
20	Module cannot be loaded (C.NOLOAD)
21	SYSINIT active - IPL or restart (C.SYSB)
22	IPU is off-line (C.IPUOFF)
23	IPU accounting timer present (C.IPUIT)
24	Inhibit operator intervention (C.NOP)
25	Reserved for RJE
26	Absolute context switch inhibit (C.ACSWI)
27	Shadow memory configuration error (C.SMERR)
28	Reserved
29	Dual-port disc mounted (C.DPMT)

		30 Activating tasks specified in the SYSGEN SEQUENCE directive (C.SEQUEN) 31 Reserved for ICS (C.ICS) 32 Reserved 33 Exclusive ANSI tape drive is configured (C.ANSI) 34-63 Reserved
A94	C.CDTA	Address of controller definition table
A98	C.CPRI	Task execution bytes: Byte Description 0 Current execution priority of currently executing task (C.CUP) 1 Base execution priority of currently executing task (C.BUP) 2 I/O priority of currently executing task (C.IOP) 3 State chain index of currently executing task (C.US)
A9C	C.DQUE	Address of CPU dispatch queue area. The CPU dispatch queue area is a variable length table built by SYSGEN. It contains the number of 58-word Dispatch Queue Entries (DQEs) specified at system generation time.
AA0	C.DTTA	Address of device type table
AA4	C.FADR	Reserved
AA8	C.FGONR	Reserved
AAC	C.GINT	Contains the count of all outstanding interrupts and traps (except SVC). It is incremented as the first instruction of every interrupt or trap service routine, and decremented by S.EXEC5, the standard interrupt and trap exit routine.
AB0	C.IDLA	CPU idle time accumulation value in seconds, cleared by SYSGEN. This value is incremented when the countdown value in C.IDLC expires.
AB4	C.IDLC	CPU idle time countdown value, cleared by SYSGEN. This value is used to load the interval timer when there are no tasks ready to run. When a task becomes ready to run, the interval timer is read and the value is stored in this word.
AB8	C.ITLT	Address of Indirectly Connected Task Linkage Table (ITLT). Initialized by SYSGEN.
ABC	C.BATSEQ	Next batch sequence number
AC0	C.JOBN	Maximum number of concurrent batch jobs
AC4	C.MGRAN	Machine dependent map granularity
AC8	C.MIDL	Address of the list of map registers used by the operating system
ACC	C.MIOP	Address of first entry of MIOP jump table

AD0	C.MODD	Address of variable length module address table. Initialized by SYSGEN. The module address table contains entries in module number sequence. Each entry consists of one word that contains the address of the entry point transfer list (HAT) of the associated module.
AD4	C.MPL	Address of master process list. Length of list in words is contained in C.NDQE plus one word. First entry points to C.MSD (hardware requirement).
AD8	C.MSD	Contains map segment descriptor for operating system (BPIX). It points to C.MIDL (hardware requirement).
ADC	C.MTIM	Number of clock interrupts per second. Initialized by SYSGEN.
AE0	C.NTIM	Number of clock interrupts per time unit. Initialized by SYSGEN.
AE4	C.PATCH	System debug patch area
AE8	C.POOL	Address of memory pool
AEC	C.SGOS	Contains the default SGO size of 32 blocks. This is included for compatibility purposes only and is not examined during job processing.
AF0	C.SICTD	Address of MIOP test device status processor, H.SICTD
AF4	C.SMTA	Address of shared memory table area. Size is determined by SYSGEN SHARE directive.
AF8	C.ARTA	Address of allocated resource table
AFC	C.SPAD	Address of CPU scratchpad image
B00	C.SVTA	Address of variable length SVC '1' table. Initialized by SYSGEN. Each entry consists of one word which contains the address of the service associated with the SVC number.
B04	C.SVTA2	Address of variable length SVC '2' table. Initialized by SYSGEN. Each entry consists of one word which contains the address of the service associated with the SVC number.
B08	C.SWAP	Contains the swapper's status and DQE address. (If bit 0 equals zero, the swapper is active. If bit 0 equals one, the swapper is inactive.) Bits 8 through 31 contain the address of the swapper's DQE.
B0C	C.SYCS	Contains the default SYC size of 32 blocks. This is included for compatibility purposes only and is not examined during job processing.

B10	C.TSKN	Task activation sequence number of currently executing task

Byte	Description
0	Contains the DQE entry number of the currently executing task in the range of 1 to 255; when word format is adjusted, it may be used as an index to the DQE address table (DAT) to obtain the DQE for the associated task. The address of the DAT is contained in C.ADAT. (C.PRNO)
1-3	Activation sequence number of currently executing task

B14	C.TSMDQA	Address of DQE for J.TSM. Required for ring processing and message sending.
B18	C.TTBT	Task timer bit table containing 256 bits. Each bit corresponds to a C.DQE entry and is accessed by the DQE entry number (1 to 255). A bit set in this table indicates the associated DQE has an active task timer.
B38	C.UDTA	Address of unit definition table
B3C	C.TTAB	Address of timer table
B40	C.MATA	Address of memory tables
B44	C.MPAA	Low address of the patch area
B48	C.MPAC	Current address of the patch area
B4C	C.MPAH	High address of the patch area
B50	C.RMTA	Address of resourcemark table
B54	C.EMTA	Address of eventmark table
B58	C.REV	MPX-32 release and interim release
B5C	C.DEBUG	Address location of debugger
B60	C.TDQ1	Time-distribution quantum stage one, in interval timer units. Initialized by SYSGEN. This value is used to load the interval timer when CPU control is dispatched to a time- distribution task under one of the following conditions:

- A task is initially selected after activation
- A task is initially selected after the termination of a voluntary wait state (e.g., wait I/O or timed suspend)
- A task is initially selected after in-swap
- A task is reselected after completion of its full quantum

During the quantum stage one interval, the currently executing task is not eligible for out-swap, and may not be pre-empted from CPU control by a higher priority time-distribution task.

B64	C.TDQ2	Time-distribution quantum stage two, in interval timer units. Initialized by SYSGEN. This value is used to load the interval timer when the stage one quantum for the currently executing task expires. (The quantum stage two value may be added to the quantum stage one value to define the full task quantum.)
B68	C.TDQ3	Time-distribution full quantum value, in interval timer units. Initialized by SYSGEN. This value is the sum of the quantum stage one and stage two values.
B6C	C.REGS	TSA address of current task
B70	C.MVTA	Address of mounted volume table
B74	C.ACTA	Address of activation table
B78	C.SEQA	Address of sequence table
B7C	C.SCDIPU	Schedule IPU routine address
B80	C.CHTA	Address of channel definition table
B84	C.ETLOC	Address of event trace logic
B88	C.ADMASK	Maximum address bit mask for machine
B8C	C.IDLA1	IPU idle time accumulation value in seconds, cleared by SYSGEN. This value is incremented when the countdown value in C.IDLC1 expires.
B90	C.IDLC1	IPU idle time countdown value, cleared by SYSGEN. This value is used to load IPU accounting interval timer (if present) when there are no tasks ready to run on the IPU. When a task becomes ready to run, the IPU accounting interval timer is read and the value is stored in this word.
B94	C.IPUIT1	Address of the IPU accounting routine, S.IPUIT1, which performs accounting functions after an IPU trap is fielded. Initialized by SYSGEN.
B98	C.IPUIT2	Address of the IPU accounting routine, S.IPUIT2, which performs accounting functions prior to the starting of the IPU. Initialized by SYSGEN.
B9C	C.BTIME	The current time (in binary) kept as the number of 100 microsecond units
BA0	C.BDATE	The current date (in binary) kept as the number of days since January 1, 1960

BA4	C.TCORR	The correction factor (in 100 microsecond units) which must be subtracted from C.BTIME to get the correct local time. This value is determined by the daylight savings and time zone parameters specified (if any) at IPL.
BA8	C.FSSP	File system stack frame pointer
BAC	C.DPTIMO	Default time-out value applied to dual-processor, shared volume resource assignments
BB0	C.MDTA	Physical starting address of the Memory Resident Descriptor Table (MDT)
BB4	C.MDTE	Physical ending address of the MDT
BB8	C.SWPLIM	Minimum number of maps to be swapped at any one time
BBC	C.PDQE	Address of DQE of a partially swapped task
BC0	C.MPXBR	Base registers save area (eight words--one file each)
BE0	C.MPXBRD	Default logical map address
BE4	C.IP00	Address of the A.IP00 module
BE8	C.PSDBRE	Break entered PSD for base mode task (two words)
BF0	C.PSDBRX	Break exited PSD for base mode task (two words)
BF8	C.PSDMSE	Message entered PSD for base mode task (two words)
C00	C.PSDMSX	Message exited PSD for base mode task (two words)
C08	C.PSDEAE	End action entered PSD for base mode task (two words)
C10	C.PSDEAX	End action exited PSD for base mode task (two words)
C18	C.DSECT	Start address of DSECT for extended MPX-32
C1C	C.ADAPT	Start address of the adapter code region for extended MPX-32
C20	C.TDEFA	Address of TERMPART, if present
C24	C.SWIOCL	Swapper's IOCL address
C28	C.CRDUMP	Address of the crash dump routine
C2C	C.HSTADR	IPU history buffer address
C30	C.CDTN	Number of entries in controller definition table
	C.ITRS	Interval timer resolution, in tenths of microseconds, as derived from the SYSGEN ITIM directive
C34	C.SMVTI	Mounted Volume Table (MVT) index of swap device
	C.SVTN	Number of entries in the SVC '1' table. Initialized by SYSGEN.
C38	C.UDTN	Number of entries in unit definition table
	C.RMTM	Maximum number of resourcemarks
C3C	C.EMTM	Maximum number of eventmarks
	C.NOS	Number of blocks required for SYSGEN code
C40	C.NRST	Number of blocks required for restart code

	C.MVTN	Number of entries in mounted volume table
C44	C.ARTN	Number of entries in allocated resource table
	C.CHTN	Number of entries in channel definition table
C48	C.HIMAP	Number of the last map block of logical address space available to a task
	C.SVTN2	Number of entries in the SVC '2' table. Initialized by SYSGEN.
C4C	C.MDTN	Total hexadecimal number of entries in the MDT. This number is larger than the number specified at SYSGEN time because it includes an extra 25% for collision resolution.
	C.MDTAV	Hexadecimal number of entries currently available in the MDT
C50	C.RMS	Reserved for RMS (two words)
C58	C.GSLEMC	Group swap limit exceeded map count
C5C	Reserved	
C60	C.BDBUG	Base task debugger name
C68	C.SPRH	Spare halfwords
CAC	C.DPTRY	Decimal number of tries to access a dual-port resource
	C.SMAPS	Number of MAPS used by the system (i.e., J.SWAPR)
CB0	C.BPRI	Default software priority level at which batch jobs execute
	C.DTTN	Number of entries in device type table
	C.FSFLGS	Reserved
	C.MODN	Entry number of last entry in module address table. Initialized by SYSGEN.
CB4	C.NITI	Contains the number of 24-word Indirectly Connected Task Linkage Block (ITLB) entries in the Indirectly Connected Task Linkage Table (ITLT). Initialized by SYSGEN.
	C.NQUE	Number of entries (255 maximum) in CPU dispatch queue.
	C.RRUN	Contains the count of memory release events. It is incremented by H.EXEC,9 when a memory scheduler event is reported. It is cleared by the memory scheduler (swapper) when processing of the memory request queue begins. It is decremented by the swapper when memory is deallocated by the swapper. It is cleared by the swapper before H.EXEC,8 is called. H.EXEC,8 will rerun the swapper if C.RRUN is not equal to zero.
	C.SMTN	Number of entries in shared memory table
CB8	C.TSMCNT	Number of currently active TSM devices. Maintained by J.TSM.
	C.TSMPRI	Priority default for TSM-activated tasks. Overrides cataloged priority.

	C.TSMTOT	Number of TSM devices. Initialized by entry point eight of all terminal device handlers.
	C.TENT	Number of timer table entries
CBC	C.RMTL	Low address of user resourcemark area
	C.EMTL	Low address of eventmark area
	C.CONF	Configuration flags:

Bit	Meaning if Set
0	CPU accelerator present (C.CPUACC)
1	IPU accelerator present (C.IPUACC)
2	IPU present (C.IPU)
3	Memory-only system (not valid in Release 2.x series) (C.MEMNLY)
4	Base code removed (C.NOBASE)
5	Ada support module present (C.ADA)
6	Shadow memory configured (C.SHMEM)
7	Shadow memory reserved (C.SHRSV)

C.MACH Machine type currently in use:

Value	Description
0	32/55 (not applicable)
1	32/75 (not applicable)
2	32/27
3	32/67
4	32/87
5	32/97
6	Reserved

CC0	C.ACTN	Number of entries in activation table
	C.DBTLC	TLC address used for system debugger
	C.SMTS	Shared memory table size in bytes
	C.SEQN	Number of entries in sequence table
CC4	C.IPUHIS	Address of IPU history buffer
CC8	C.SHRHI	Interprocessor memory high bound

Bit	Meaning if Set
17	MS^2 memory is configured

	C.SHRLO	Interprocessor memory low bound
CCC	C.MAXSWP	Maximum swap size in megabytes
	C.SPB1	Reserved

CD8	C.MMSG	Nonprivileged task's no-wait message count
	C.MRUN	Nonprivileged task's no-wait run request count
	C.MNWI	Nonprivileged task's no-wait I/O count
	C.GSLEGI	Group ID of a task who's group outswap limits have been exceeded
CDC	C.GSLEPR	Hexadecimal priority of a task who's group outswap limits have been exceeded
	C.ADAFL	Control flag for Ada run-time system
	C.TKILL	Number of seconds before an abort becomes a kill
	C.DELTA	Delta value for real-time IPU tasks
CE0	C.MPXBRN	Number of base registers to load
	C.DBMAPS	Number of MAPS used by the system debugger
	C.SWAPSZ	Swapfile size in megabytes
CE4	C.DTSAVE	Elapsed time before J.DTSAVE resumes
CE8	C.SHCPU	CPU shadow memory. Starting map block number in first halfword. Number of map blocks in second halfword.
CEC	C.SHIPU	IPU shadow memory. Starting map block number in first halfword. Number of map blocks in second halfword.
CF0	C.UPDT	MPX-32 patch or replacement release
CF4	C.SWPBUF	Logical address of J.SWAPR's dedicated system buffer
CF8	C.MRQTMR	Timer used by J.SWAPR
CFC	C.SPARE	Reserved (two words)
D00	C.TABLES	Symbol equated to the absolute memory location at which SYSGEN-built tables begin if no user communication region is SYSGENed. This location is on a word boundary.
D04	C.USER	Symbol equated to the absolute memory location where the user communication region begins. This region is defined by the CDOTS directive to SYSGEN and is on a word boundary. If the CDOTS directive is specified, the SYSGEN-built tables begin at the first word location following the user region.

2.3 Allocated Resource Table (ART)

The Allocated Resource Table (ART) is a system resident structure that provides a central mechanism to control the manipulation of all allocated resources. An entry is made for a resource when it is allocated, and remains while there are active assignments to that resource. Shared resources are given an entry in the ART by the first process to allocate them. The table is linked to each task's service area File Assignment Table (FAT) entry for the respective resource.

When the ART entry is made at assignment, the resource assign count is incremented. The assign count is decremented when the task deallocates the resource. The resource is not physically deallocated until the assign count equals zero; the physical deallocation of the resource is not performed while it is in use.

Other information is kept in the ART when a resource is determined to be implicitly shared. For files, pointers are kept to indicate the position of writers on the file by the current end-of-file and end-of-medium positions. These pointers are identified by relative block number. The current allowable access modes are also noted in the ART entry when the resource is implicitly shared.

The size of the ART is determined at SYSGEN by the ARTSIZE directive.

	0 7	8 15	16 23	24 31
Word 0	UDT index (AR.UDTI)	Resource descriptor block address (AR.BLOCK)		
1	Current access mode (AR.CACM). See Note 1.	Resource Pointer. See Note 2.		
2	Resource allocation flags (AR.FLAGS). See Note 3.		DQE index of exclusive lock owner (AR.XRL)	DQE index of synchronous lock owner (AR.SRL)
3	Number of active assignments (AR.ASSNS)	Number of users/allocations of resource (AR.USERS)	Number of multiprocessor requests queued for this resource (AR.QUE)	Number of readers currently on this resource minus the number of writers, appenders, modifiers and updaters (AR.RDRS)
4	Current EOF position in this file (AR.EOF)			
5	Current EOM position in this file (AR.EOM)			
6	Port number of multiport resource lock owner (AR.MPID)		Reserved	
7	Reserved			

Notes:

1. Bits in AR.CACM are assigned as follows (implicit shared use only):

Bits	Meaning if set
0	Read access (RD.READ)
1	Write access (RD.WRITE)
2	Modify access (RD.MODFY)
3	Update access (RD.UPDAT)
4	Append access (RD.APPDN)
5-7	Reserved

2. Resource pointer is as follows:

Resource	Pointer
Volume	Mounted volume table entry pointer (AR.MVTA)
Segment definition	Number of blocks in segment definition (AR.NBLKS)
Partition	Shared memory table entry pointer (AR.SMTA)
Device	Unit definition table entry pointer (AR.UDTA)

3. Bits in AR.FLAGS are assigned as follows:

Bit	Meaning if Set
0	Allocated for explicit shared use (AR.EXSHR)
1	Allocated for implicit shared use (AR.IMSHR)
2	Allocated as mount device (AR.MNT)
3	Marked for deletion (AR.DELET)
4	Segment definition (AR.SPACE)
5	Memory partition (AR.PART)
6	Device (AR.DEVC)
7	Entry is active (AR.ACTV)
8	Resource marked for truncation (AR.TRUNC)
9-10	Reserved
11	Dual-processor resource is being appended by a task in this system environment (AR.WOWN2)
12	Dual-processor lock is in effect on this resource (AR.DPLK)
13	Dual-processor resource is being written to by a task in this system environment (AR.WOWN)
14	Multiprocessor volume flag (AR.DUALP)
15	Port designation for resource lock owner when resource is treated as dual processor (AR.PORT). This bit is used only when the system is SYSGENED to be compatible to a previous release.

2.4 Blocking Buffer Control Cells

Blocking buffer control cells are built by IOCS for blocked files as the file is written and they become a permanent part of the file. This information is then used by IOCS as the file is read to unblock individual records within the file.

Blocking Buffer Control Word

Word	0	7 8	31
0	Buffer status. See Note.	Next read/write address	

Note: Bits in buffer status are assigned as follows:

Bit	Status
0	Reserved
1	Buffer is empty
2	Buffer is output active
3	Reserved
4	Buffer is free to allocate
5-7	Reserved

Record Control Bytes

0 7	8 15	16 23	24 31
Status bits last record. See Note 1.	Byte count last record	Status bits this record. See Note 2.	Byte count this record

For the last record in a block, bytes two and three -- status bits this record and byte count this record -- are omitted.

Notes:

1. Bits in this field are assigned as follows:

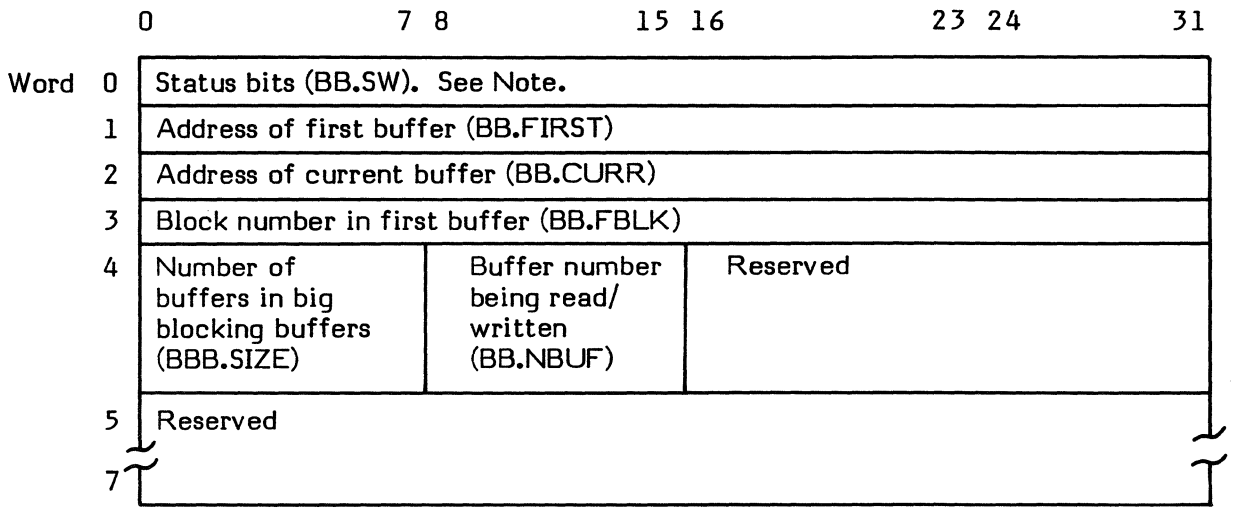
Bit	Meaning if Set
0	End-of-file
1	Beginning-of-block
2	End-of-block
3-7	Reserved

2. Bits in this field are assigned as follows:

Bit	Meaning if Set
0	End-of-file
1-7	Reserved

2.4.1 Blocking Buffer Head Cells

The DFT.BBA field of the FAT contains the address of the eight word blocking buffer head cell. Head cells for system allocated large blocking buffers are built in the TSA. User-supplied large blocking buffer head cell space is allocated from memory pool and deallocated when the file is closed or during exit/abort processing. The total number, as well as the address, of the first head cell are contained in T.BBHCA. The head cell includes the following information:



Note: Status bits in BB.SW are assigned as follows:

Bit	Meaning if Set
0	Blocking buffer status word (BB.SW)
1	Buffer is empty (SW.EMP)
2	Buffer is output active (SW.OUT)
3	Buffer is in use (SW.BBB)
4	Buffer is free to allocate (SW.FRE)
5	Buffer is allocated by H.BKDM (SW.ALL)
6	User-supplied buffer is in use (SW.UBB)
7	Reserved for S.BKDM9 (SW.SVC)
8-31	Read/Write address

2.5 Caller Notification Packet (CNP)

The Caller Notification Packet (CNP) is the mechanism used by the Resource Management Module (H.REMM) and the Volume Management Module (H.VOMM) for handling abnormal conditions that may result during resource requests. All or part of this structure can be used by a particular service being called. The CNP must be on a word boundary.

	0 7	8 15	16 23	24 31
Word 0	Time-out value (CP.TIMO)			
1	Abnormal return address (CP.ABRET)			
2	Option field (CP.OPTS). See Note 1.		Status field (CP.STAT). See Note 2.	
3	Reserved			
4				
5	Automatic open FCB address (CP.FCBA)			

Notes:

1. A bit sequence and/or value used to provide additional information that can be necessary to fully define the calling sequence for a particular service.

2. A right-justified numeric value identifying the return status for this call.

2.6 Channel Definition Table (CHT)

The Channel Definition Table (CHT) is a system resident structure applicable only to F-class and IPS extended I/O devices. The CHT is built by the SYSGEN process, one for each extended I/O channel configured in the system. It serves as a register save area, contains the interrupt context block associated with extended I/O protocol, identifies Controller Definition Tables (CDTs) linked to the channel, and defines other pertinent channel information.

Word	0 7 8 15 16 23 24 31
0	Register save area (CHT.REGS). See Note 1.
7	
8	Old PSD1/old PSD2 (CHT.OPSD)
9	
10	New PSD1/new PSD2 (CHT.NPSD)
11	
12	IOCL address (CHT.IOCL)
13	Status address (CHT.STAD)
14	Flag word (CHT.FLGS). See Note 2.
15	Channel spurious interrupt count (CHT.SPUR) | Channel interrupt priority* (CHT.IPL) | Channel address* (CHT.CHAN)
16	CDT address unit 0* (CHT.CDT0). See Note 3.
17	CDT address unit 1* (CHT.CDT1). See Note 3.
30	
31	CDT address unit 15* (CHT.CDTF). See Note 3.
32	IOP status doubleword (CHT.STDW) (or) Subaddress (CHT.SUBA) | Real IOCD address (CHT.RIOA)
33	Channel status (CHT.CHST) | Cont/device status (CHT.CDST) | Residual byte count (CHT.RBC)
34	Address of XIO.SUB exit entry point (CHT.EXIT). See Note 4.
35	Address of H.IFXIO initialization entry point (CHT.INCH). See Note 5.
36	SIO status stored return address (CHT.RTN)
37	HIO status stored return address (CHT.HRTN)
38	Reserved for future development use.
39	Reserved for future development use.

* Initialized by SYSGEN

Notes:

1. CHT.REGS must begin on a register file boundary.

2. Bits in CHT.FLGS are assigned as follows:

Bit	Meaning if Set
0	INCH (initialize channel) has been performed
1	Status stored response for SIO or HIO instruction
2	SI. routine was called from LI.XIO routine (common XIO routines)
3	Interrupt level was activated by IQ.XIO routine (common XIO routines)
4	Cache controller (CHT.CAC)
5-31	Reserved

3. These fields contain the addresses of the CDT entries for controllers connected to the corresponding XIO channel. Entries for unimplemented controllers are set to zero.

4. CHT.EXIT contains the address of the exit procedure within the common XIO subroutines.

5. CHT.INCH contains the address of the initialization procedure used to initialize the corresponding XIO channel.

2.7 Controller Definition Table (CDT)

The Controller Definition Table (CDT) is a system resident structure used to identify information required by handlers and the I/O processor for a specific controller. The CDT is built by the SYSGEN process, one for each controller configured on the system. The CDT identifies devices (UDTs) associated with the controller, the handler address associated with the controller, and defines other pertinent controller information.

Word	0 7	8 15	16 23	24 31
0	String forward address (CDT.FIOQ)			
1	String backward address (CDT.BIOQ)			
2	Link priority (CDT.LPRI). See Note 1.	Number of entries in list (CDT.IOCT). See Note 2.	Class (CDT.CLAS). See Note 3.	Flags (CDT.FLG2). See Note 4.
3	CDT index (CDT.INDX)		Device type code (CDT.DTC)	Interrupt priority level (CDT.IPL)
4	Number units on controller (CDT.NUOC)	Number requests outstanding (CDT.IORO)	Channel number (CDT.CHAN)	Subaddress of first device (CDT.SUBA)
5	Program number if reserved (CDT.PNRC)	Interrupt handler address (CDT.SIHA) or controller information block (CDT.CIF)		
6	Flags (CDT.FLGS). See Note 5.	UDT address of first device on controller (CDT.UDTA)		
7	I/O status (CDT.IOST). See Note 6.	TI address (CDT.TIAD) or SI address if extended I/O (CDT.SIAD)		
8	UDT address unit 0* (CDT.UT0)			
9	UDT address unit 1* (CDT.UT1)			
23	UDT address unit 15* (CDT.UTF)			

*Initialized by SYSGEN

Notes:

1. Always zero (head cell)

2. Number of entries in list (zero if none)

3. Values in CDT.CLAS are assigned as follows:

Value	Meaning
X'0D'	TCW type with extended addressing capability
X'0E'	TCW type
X'0F'	Extended I/O

4. Bits in CDT.FLG2 are assigned as follows:

Bits	Meaning if Set
0-7	Reserved for future use

5. Bits in CDT.FLGS are assigned as follows:

Bit	Meaning if Set
0	Extended I/O device (CDT.FCLS)
1	I/O outstanding (set by handler, reset by IOCS) (CDT.IOU1)
2	GPMC device (CDT.GPMC)
3	Initialization (INC) needs to be performed for this controller (CDT.FINT)
4	D-class (16MB GPMC) (CDT.XGPM)
5	Used only when IOQs are linked to the CDT. Set when SIO is accepted by the controller. Reset when IOQ is unlinked from the CDT or when I/O is reported complete to IOCS in the case of operator intervention type errors (CDT.IOU5).
6	IOP controller (CDT.IOP)
7	Controller malfunction (CDT.MALF)

6. Bits in CDT.IOST are assigned as follows:

Bit	Meaning if Set
0	IOQ linked to UDT (CDT.NIOQ)
1	Multiplexing controller (CDT.MUXC)
2	Use standard XIO interface
3	D-class GPMC (CDT.XGPS)
4	Cache controller (CDT.CAC)
5	H.F8XIO has determined if the controller is pre-8512-2 or not (CDT.CKFL)
6	Controller not pre-8512-2 (CDT.FLOW)
7	Reserved for FMS

7. CDT.SIZE is 24 words

2.8 Device Context Area (DCA)

A Device Context Area (DCA) exists for each active subchannel and serves as a storage area for information regarding the subchannel and its operation. The DCAs are physically located at the end of each device-dependent handler (H.??XIO) and must be doubleword bounded. The first 33 words of each DCA are identical; however, additional words can be added to suit the needs of the particular device. The following represents the first 33 words of each DCA.

Word	0 7 8 15 16 23 24 31
0	DCA size (DCA.SIZE)
1	Device address (DCA.UADD) | Reserved
2	CHT address (DCA.CHTA)
3	CDT address (DCA.CDTA)
4	UDT address (DCA.UDTA)
5	IOQ address (DCA.IOQA)
6	Lost interrupt count (DCA.LINC)
7	Spurious interrupt count (DCA.SINC)
8	Total retry count this device (DCA.RETC)
9	Flags (DCA.FLAG). See Note 1. | Retry count this request (DCA.RCNT)
10	UDT address (DCA.NUDT). See Note 2.
11	Status word one (DCA.WST1)
12	Status word two (DCA.WST2)
13	Number of reserves outstanding (DCA.RESC)
14	Time-out value opcode 0 (DCA.TIM0). See Note 3.
15	Time-out value opcode 1. See Note 3.
29	Time-out value opcode F. See Note 3.
30 31	Sense IOCD (DCA.SENI)
32	Sense buffer (DCA.SENS)

Notes:

1. Bits in DCA.FLAG are assigned as follows:

Bit	Meaning if Set
0	Interrupts not expected
1	HIO issued at LI.XIO
2	HIO needs to be reissued
3	Device rewinding or seeking
4	Sense issued without an IOQ
5	Device is an XIO magnetic tape
6-15	Reserved for common subroutine usage
16-23	Reserved for device dependent handler usage

2. This UDT address is the UDT address of the device for which an SIO or HIO was issued when a status stored response was generated for this device. It indicates the need to reissue the I/O request for that device.

3. Time-out values corresponding to opcodes 0 through F (16 entries).

2.9 Device Type Table (DTT)

The Device Type Table (DTT) is a system resident structure used to identify device types that are configured in the system and their associated controllers. The DTT is built by the SYSGEN process and its entries are linked to the associated Controller Definition Table (CDT).

Valid device type codes are listed in Appendix A of the MPX-32 reference manuals.

<table>
<tr><td>Word 0</td><td>7 8</td><td>15 16</td><td>31</td></tr>
<tr><td>0</td><td>Device type code (DTT.COD) See Note 1</td><td colspan="2">Address of first CDT entry of this type (DTT.CDTA)</td></tr>
<tr><td>1</td><td>Number of controller entries (DTT.CNT)</td><td>Flags (DTT.FLGS). See Note 2.</td><td>ASCII device mnemonic (DTT.NAM). See Note 3.</td></tr>
</table>

Notes:

1. For example, 01 = any disc, 04 = any magnetic tape, 08 = any reader card, and 0A = any line printer.

2. Used by job control and cataloger to validate ASSIGN3 statements with bits assigned as follows:

Bit	Meaning if Set
0	Entry of device address not legal
1	Entry of size or reel ID required
2	Entry of reel ID required
3-7	Reserved

3. For example, X'4443' (DC) = any disc; X'4D54' (MT) = any tape

2.10 Directory Entry Table (M.DN.TEQ)

The Directory Entry Table (M.DN.TEQ) contains information pertinent to resources defined in a directory. Each resource defined in a directory has an M.DN.TEQ associated with it.

Word 0	7 8	15 16	23 24	31
0 3	Resource name (DN.IDNAM)			
4	Binary creation date (DN.DATE)			
5	Binary creation time (DN.TIME)			
6	Absolute block number of resource descriptor (DN.DOFF)			
7	Resource ID flags (DN.RDFLG). See Note 1.		Resource type (numeric value) (DN.RTYPE). See Note 2.	
8	Number of entries that collided with this entry (DN.COLCT)			
9	Number of hashes required to locate this entry (DN.HSHCT)			
10	Directory entry flags (DN.FLAGS). See Note 3.			
11	Directory entry index (DN.DIRI)			
12 13	Owner name of directory entry creator (DN.OWNR)			
14 15	Filler (DN.FILL)			

Notes:

1. Internal flags reserved for MPX-32

2. Values for DN.RTYPE are as follows:

Value	Meaning
1	Volume type (DN.VOL)
2	Resource descriptor description (DN.RESRC)
3	Descriptor map descriptor (DN.DMAP)
4	Space map descriptor (DN.SMAP)
5	Root directory descriptor (DN.ROOT)
6	System image descriptor (DN.IMAGE)
7	Bad block descriptor (DN.BDBLK)
8	Value for spool file descriptor (DN.SYM)
9	Extra segment definition descriptor (DN.XSEGD)
10	Permanent file (DN.FILE)
11	Permanent directory (DN.DIR)
12	Temporary file (DN.TFILE)
13	Temporary directory (DN.TDIR)
14	Static memory partition (DN.MEM)
15	Dynamic memory partition (DN.TMEM)
16	Device descriptor (DN.DEVC)
17	Resource descriptor for the DMAP bad block deallocation file (DN.BDMAP)
18	Resource descriptor for the SMAP bad block deallocation file (DN.BSMAP)

3. Bits in DN.FLAGS are assigned as follows:

Bit	Meaning if Set
0	Active entry (DN.ACTIV)
1-31	Reserved

2.11 Dispatch Queue Area

The Dispatch Queue Area is a variable length doubleword-bounded table built by the system generation program. It contains a maximum of 255 Dispatch Queue Entries (DQEs). The address of the dispatch queue area is contained in C.DQUE. The number of DQE entries is contained in C.NQUE. 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 core-resident information necessary to describe the task to the system. 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 scheduler to describe the task's current status. When a task exits, its DQE is again linked to the free list.

2.12 Dispatch Queue Entry (DQE)

The Dispatch Queue Entry (DQE) contains all of the core-resident information required to describe an active task to the system. It is always linked to the CPU scheduler state chain that describes the current execution status of the associated task.

Dispatch Queue Entry (DQE) Table

Word # (Decimal)	Byte (Hex)	0 7 8 15 16 25 26 31
0	0	DQE.SF
1	4	DQE.SB
2	8	DQE.CUP | DQE.BUP | DQE.IOP | DQE.US
3	C	DQE.NUM/DQE.TAN
4-5	10	DQE.ON
6-7	18	DQE.LMN
8-9	20	DQE.PSN
10	28	DQE.USW
11	2C	DQE.USHF
12	30	DQE.MSD
13	34	DQE.KCTR
14	38	DQE.MMSG | DQE.MRUN | DQE.MNWI | DQE.GQFN
15	3C	DQE.UF2 | DQE.IPUF | DQE.NWIO | DQE.SOPO
16	40	DQE.CQC
17	44	DQE.SH | DQE.SHF | DQE.TIFC | DQE.RILT
18	48	DQE.UTS1
19	4C	DQE.UTS2
20	50	DQE.DSW
21	54	DQE.PRS
22	58	DQE.PRM
23	5C	Reserved | DQE.MSPN | DQE.MST
24	60	DQE.PSSF
25	64	DQE.PSSB
26	68	DQE.PSPR | DQE.PSCT | DQE.ILN | DQE.RESU
27	6C	DQE.TISF
28	70	DQE.TISB
29	74	DQE.TIPR | DQE.TICT | DQE.SWIF | DQE.UBIO
30	78	DQE.RRSF
31	7C	DQE.RRSB
32	80	DQE.RRPR | DQE.RRCT | DQE.NSCT
33	84	DQE.MRSF
34	88	DQE.MRSB
35	8C	DQE.MRPR | DQE.MRCT | DQE.NMRR | DQE.NMMR
36	90	DQE.RTI | Reserved | DQE.ATI | Reserved
37	94	DQE.SAIR/DQE.TAD
38-40	98	DQE.ABC
41	A4	Reserved
42-43	A8	DQE.SRID
44-51	B0	DQE.CDIR/DQE.CVOL
52	D0	Reserved
53	D4	DQE.ACX2
54	D8	DQE.MRQ | DQE.MEM | DQE.MEMR
55	DC	DQE.MRT | Reserved | DQE.RMMR
56	E0	DQE.MAPN | DQE.CME
57	E4	DQE.CMH | DQE.CMS

Byte (Hex)	Symbol	Description
0	DQE.SF	String forward linkage address; Field length = 1W; Standard linked list format; Contains address of next (top-to-bottom) entry in chain.
4	DQE.SB	String backward linkage address; Field length = 1W; Standard linked list format; Contains address of next (bottom-to-top) entry in chain.
8	DQE.CUP	Current user priority; Field length = 1B; Standard linked list format; This priority is adjusted for priority migration based on situational priority increments. Situational priority increments are based on the base level priority (DQE.BUP) of the task.
	DQE.BUP	Base priority of user task; Field length = 1B; Used by scheduler to generate DQE.CUP (current priority) based on any situational priority increments.
	DQE.IOP	I/O priority; Field length = 1B; Initially set from base priority; Used for I/O queue priority.
	DQE.US	State chain index for this user task; Field length = 1B; Range: zero through X'1E'; Indicates current state of this task, such as ready-to-run priority, I/O wait, resource block, etc.

Label	Index	Task description
FREE	00	DQE is available (in free list)
PREA	01	Activation in progress
CURR	02	Currently executing task or is pre-empted time-distribution task in quantum stage one
SQRT	03	Ready to run (PRI. LEV. 1 to 54)
SQ55	04	Ready to run (PRI. LEV. 55)
SQ56	05	Ready to run (PRI. LEV. 56)
SQ57	06	Ready to run (PRI. LEV. 57)
SQ58	07	Ready to run (PRI. LEV. 58)
SQ59	08	Ready to run (PRI. LEV. 59)
SQ60	09	Ready to run (PRI. LEV. 60)
SQ61	0A	Ready to run (PRI. LEV. 61)
SQ62	0B	Ready to run (PRI. LEV. 62)
SQ63	0C	Ready to run (PRI. LEV. 63)
SQ64	0D	Ready to run (PRI. LEV. 64)

Label	Index	Task description
SWTI	0E	Waiting for terminal input
SWIO	0F	Waiting for I/O
SWSM	10	Waiting for message complete
SWSR	11	Waiting for run request complete
SWLO	12	Waiting for low speed output
SUSP	13	Waiting for timer expiration, resume request, or message interrupt
RUNW	14	Waiting for timer expiration, or run request
HOLD	15	Waiting for a continue request
ANYW	16	Waiting for timer expiration, no-wait I/O complete, no-wait message complete, no-wait run request complete, message interrupt, or break interrupt
SWDC	17	Waiting for disc space
SWDV	18	Waiting for device allocation
SWFI	19	Waiting for file system
MRQ	1A	Waiting for memory
SWMP	1B	Waiting for memory pool
SWGQ	1C	Waiting in general wait queue
CIPU	1D	Current IPU task in execution
RIPU	1E	IPU requesting state

C DQE.NUM DQE entry number;
Field length = 1B;
Used as an index to DQE address table (DAT);
Range: one through "N" (for MPL index compatibility);
Used by scheduler to set C.PRNO to reflect the currently executing task. This value is also used as the MPL index. It is used by the scheduler to initialize the CPIX in the PSD before loading the map for this task.

DQE.TAN Task activation sequence number;
Field length = 1W;
This number is assigned by the activation service and uniquely identifies a task.

Note: The most significant byte of this value is the DQE entry number and is accessible as DQE.NUM.

10 DQE.ON Owner name;
Field length = 1D.

18 DQE.LMN Load module name;
Field length = 1D.

20 DQE.PSN Pseudonym associated with task;
Field length = 1D;
This parameter is an optional argument accepted by the pseudo task activation service. It can be used to uniquely identify a task within a subsystem, such as multibatch. It contains descriptive information useful to the system operator or to other tasks within a subsystem. Conventions used to generate a pseudonym are determined by the associated subsystem. A system-wide convention should be used to establish pseudonym prefix conventions to avoid confusion between subsystems.

28 DQE.USW User status word;
Field length = 1W.

2C DQE.USHF Scheduling flags;
Field length = 1W;
Used by the scheduler to indicate special status conditions.

Bit	Meaning if Set
00	Load protection image requested (DQE.LPI)
01	Single copy load module (DQE.SING)
02	Task is indirectly connected (DQE.INDC)
03	Task is privileged (DQE.PRIV)
04	Task has message receiver (DQE.MSGR)
05	Task has break receiver (DQE.BRKR)
06	Task quantum stage one expired (DQE.QS1X)
07	Task quantum stage two expired (DQE.QS2X)
08	In-swap I/O error (DQE.INER)
09	Wait I/O request outstanding (DQE.WIOA)
10	Wait I/O complete before in-progress notification (DQE.WIOC)
11	Inhibit message pseudointerrupt (DQE.INMI)
12	Batch origin task (DQE.BAOR)
13	Running in TSM environment (DQE.TMOR)
14	Task abort in progress (DQE.ABRT)
15	Task is in pre-exit state (DQE.PRXT)
16	Run receiver mode (DQE.RRMD)
17	Wait send message outstanding (DQE.WMSA)
18	Wait message complete before link to wait queue (DQE.WMSC)
19	Wait mode send run request outstanding (DQE.WRRA)
20	Wait mode send run request complete before link to wait queue (DQE.WRRC)
21	Debug associated with task (DQE.DBAT)
22	Real-time task (DQE.RT)
23	Time-distribution task initial dispatch (DQE.TDID) Set by: . H.ALOC1 on activation of T/D task. . S.EXEC51 when task is linked to wait state. . H.EXEC7 on completion of inswap or other memory request.

Reset by:
- S.EXEC20 on initial dispatch of task after activation
- Wait state termination
- In-swap

	24	Task delete in progress (DQE.DELP)
	25	Task abort (with abort receiver) in progress (DQE.ABRA)
	26	Abort receiver established (DQE.ABRC)
	27	Asynchronous abort/delete inhibited (DQE.ADIN)
	28	Asynchronous delete deferred (DQE.ADDF)
	29	Task is inactive (DQE.INAC)
	30	Asynchronous abort deferred (DQE.AADF)
	31	Activation timer in effect (DQE.ACTT)

30 DQE.MSD Physical address of MIDL in TSA;
Field length = 1W.

34 DQE.KCTR Kill/abort timer;
Field length = 1W.

38 DQE.MMSG Maximum number of no wait messages allowed to be sent by this task;
Field length = 1B.

DQE.MRUN Maximum number of no-wait run requests allowed to be sent by this task;
Field length = 1B.

DQE.MNWI Maximum number of no-wait I/O requests allowed to be concurrently outstanding for this task;
Field length = 1B.

DQE.GQFN Contains the generalized queue (SWGQ) function code;
Field length = 1B;
Function codes are queued as follows:

Code	Meaning
01	Volume resource (QVRES)
02	ART space (QART)
03	Mount in progress (QMNT)
04	Resourcemark lock (QRSM)
05	Reserved for eventmark (QEVM)
06	Read wait for writer (QGEN)
07	Shared memory table (QSMT)
08	Synchronous resource lock (QSRL)
09	Mounted volume table (QMVT)
0A	Dual-port lock (QDPLK)

3C DQE.UF2 Scheduling flags;
Field length = 1B;

Bit	Meaning if Set
0	Enable debug mode break (DQE.EDB)
1	Generalized wait queue time-out (DQE.GQTO)
2	Task interrupts are synchronized (DQE.SYNC)
3	Task is part of a job (DQE.JOB)
4	ACX-32 task flag (DQE.ACX)
5	Special arithmetic function requested (DQE.AF)
6	Reserved
7	Run request terminated (DQE.RRT)

DQE.IPUF IPU flag byte;
Field length = 1B;

Bit	Meaning if Set
0	IPU inhibit flag (DQE.IPUH)
1	IPU bias flag (DQE.IPUB)
2	CPU only (DQE.IPUR)
3	OS execution direction flag (set when PSD is in user area) (DQE.OSD)
4	Base register task (DQE.BASE)
5	Ada task (DQE.ADA)
6-7	Reserved

DQE.NWIO Number of no-wait I/O requests;
Field length = 1B.

DQE.SOPO Priority bias only swapping control flags;
Field length = 1B;

Bit	Meaning if Set
0	SWGQ state priority-based swapping (DQE.GQPO)
1	Swap inhibit due to bit map access (DQE.BMAP)
2	Inhibit swap device while accessing MDT (DQE.MDTA)
3	User swap inhibit flag (DQE.USWI)
4	User swap on priority only flag (DQE.USPO)
5-7	Reserved

40 DQE.CQC Current quantum count;
Field length = 1W;
Used by the scheduler to accumulate elapsed execution time for the task to compare the level unique stage one and stage two time-distribution values.

44 DQE.SH Used by J.SWAPR to swap shadow memory;
Field length = 1B.

DQE.SHF — Shadow memory flag;
Field length = 1B;

Bit	Meaning if Set
0	Task requests shadow memory (DQE.SHAD)
1	IPU shadow memory requested (DQE.SHI)

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

Value	Meaning
00	Not active
01	Request interrupt
02	Resume program from suspend (SUSP) queue
03	Resume program from any-wait (ANYW) queue
04	Resume program from run-request-wait (RUNW) queue
05	Resume program from generalized (SWGQ) queue
06	Resume program from peripheral device (SWDV) queue
07	Resume program from disc space (SWDC) queue

DQE.RILT — Request Interrupt (RI) level for timer;
Field length = 1B;
Identifies the interrupt level to be requested upon timer expiration.

48 DQE.UTS1 — User timer slot word 1;
Field length = 1W;
Current timer value;
Contains negative number of timer units before time out.

4C DQE.UTS2 — User timer slot word 2;
Field length = 1W;
Reset timer value;
Contains negative number of time units;
Used to reset the current timer value when it expires.

50 DQE.DSW — Base mode debugger status word (PCALL);
Field length = 1W.

54 DQE.PRS

Peripheral requirement specification;
Field length = 1W;

Bit	Description
0-7	Reserved
8-15	Device type code
16-23	Channel address
24-31	Subchannel address or contains first word of SWGQ ID.

58 DQE.PRM

Peripheral requirements mask;
Field length = 1W;

Value	Meaning
X'00FF0000'	Any device of this type code
X'00FFFF00'	Any device of the specified type code on the specified channel
X'00FFFFFF'	The specified device as described by type code, channel, and subchannel address, or contains second word of SWGQ ID.

5C Reserved

Field length = 2B.

DQE.MSPN

TSA maps required to span MIDLS and MEMLs;
Field length = 1B.

DQE.MST

Static memory type specification;
Field length = 1B;

Value	Memory Class
01	E
02	H
03	S

This field is used to specify the type of memory required for in-swap.

60 DQE.PSSF

Pre-emptive system service head cell string forward linkage address;
Standard head cell format;
Field length = 1W;
Contains address of next (top-to-bottom) entry in chain.

64	DQE.PSSB	Pre-emptive system service head cell string backward linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
68	DQE.PSPR	Pre-emptive system service head cell dummy priority (always zero); Standard head cell format; Field length = 1B.
	DQE.PSCT	Pre-emptive system service head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.ILN	Interrupt level number; Field length = 1B; Identifies associated interrupt level for interrupt connected tasks.
	DQE.RESU	Reserved usage index; Field length = 1B.
6C	DQE.TISF	Task interrupt head cell string forward linkage address; Standard head cell format; Field length = 1W; Contains address of next (top-to-bottom) entry in chain.
70	DQE.TISB	Task interrupt head cell string backward linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
74	DQE.TIPR	Task interrupt head cell dummy priority (always zero); Standard head cell format; Field length = 1B.
	DQE.TICT	Task interrupt head cell number of entries in list; Standard head cell format; Field length = 1B.

DQE.SWIF — Swapping inhibit flags;
Field length = 1B;

Bit	Task Meaning if Set
0	Resident (DQE.RESP)
1	Locked in memory (DQE.LKIM)
2	Unbuffered I/O in progress (DQE.IO)
3	Outswapped (DQE.OTSW)
4	Leaving system (DQE.TLVS)
5	Forced unswappable during terminal output (DQE.FCUS)
6	Forced unswappable because swap file has not been allocated for it (DQE.FCRS)
7	Imbedded in the operating system (DQE.INOS)

DQE.UBIO — Number of unbuffered I/O requests currently outstanding;
Field length = 1B.

78 DQE.RRSF — Run receiver head cell string forward linkage address;
Standard head cell format;
Field length = 1W;
Contains address of next (top-to-bottom) entry in chain.

7C DQE.RRSB — Run receiver head cell string backward linkage address;
Standard head cell format;
Field length = 1W;
Contains address of next (bottom-to-top) entry in chain.

80 DQE.RRPR — Run receiver head cell-dummy priority (always zero);
Standard head cell format;
Field length = 1B.

DQE.RRCT — Run receiver head cell number of entries in list;
Standard head cell format;
Field length = 1B.

DQE.NSCT — Number of map blocks out swapped;
Field length = 1H.

84 DQE.MRSF — Message receiver head cell string forward linkage address;
Standard head cell format;
Field length = 1W;
Contains address of next (top-to-bottom) entry in chain.

88 DQE.MRSB — Message receiver head cell string backward linkage address;
Standard head cell format;
Field length = 1W;
Contains address of next (bottom-to-top) entry in chain.

8C	DQE.MRPR	Message receiver head cell dummy priority (always zero); Standard head cell format; Field length = 1B.
	DQE.MRCT	Message receiver head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.NWRR	Number of no-wait mode run requests outstanding; Field length = 1B.
	DQE.NWMR	Number of no-wait mode messages requests outstanding; Field length = 1B.
90	DQE.RTI	Requested task interrupt flags; Field length = 1B;

Bit	Meaning if Set
0	Reserved
1	Priority one end action request. Used for pre-emptive system services. (DQE.EA1R)
2	Debug break request (DQE.DBBR)
3	User break request (DQE.UBKR)
4	Priority two end action request (DQE.EA2R)
5	Message interrupt request (DQE.MSIR)
6-7	Reserved

Reserved — Field length = 1B.

DQE.ATI — Active task interrupt flags;
Field length = 1B;

Bit	Meaning if Set
0	Reserved
1	Priority one active end action (DQE.AEA1)
2	Active debug break (DQE.ADM)
3	Active user break (DQE.AUB)
4	Priority two active end action (DQE.AEA)
5	Active message interrupt (DQE.AMI)
6-7	Reserved

Reserved — Field length = 1B.

94	DQE.SAIR	System action task interrupt request;

Bit	Meaning if Set
0	Request for delete of this task (DQE.DELR)
1	Reserved
2	Hold task request (DQE.HLDR)
3	Abort task request (DQE.ABTR)
4	Exit task request (DQE.EXTR)
5	Suspend task request (DQE.SUSR)
6	Run receiver mode request (DQE.RRRQ)
7	Reserved

DQE.TAD — TSA address (logical);
Field length = 1W;
Byte zero contains DQE.SAIR.

98 DQE.ABC — Abort code;
Field length = 3W.

A4 Reserved

A8 DQE.SRID — Used swap space linked list;
Field length = 2W.

B0 DQE.CDIR — Load module RID at activation;
Field length = 8W.

DQE.CVOL — Current working volume at activation;
Field length = 8W.

D0 Reserved

D4 DQE.ACX2 — Advance communication word;
Field length = 1W.

D8 DQE.MRQ — Memory request doubleword;
Reserved field length = 1B.

DQE.MEM — Type of memory requested;
Field length = 1B;

Value	Memory Class
01	E
02	H
03	S

	DQE.MEMR	Number of memory blocks required; Field length = 1H.
DC	DQE.MRT	Memory request type code; Field length = 1B;

Value	Meaning
00	In-swap only
01	Preactivation request
02	Activation request
03	Memory expansion request
04	IOCS buffer request
05	Shared memory request
06	System buffer request
07	Release swap file space

If DQE.MRT equals 05, the next three bytes will contain the address of the shared memory table entry.

	Reserved	Field length = 1B.
	DQE.RMMR	Map register for requested memory; Field length = 1H.
E0	DQE.MAPN	Inclusive span of maps in use; Field length = 1H.
	DQE.CME	Number of swappable class E map blocks currently allocated; for resident tasks, if not zero, reflects the total number of map blocks in use. Field length = 1H.
E4	DQE.CMH	Number of swappable class H map blocks currently allocated; for resident tasks, if not zero, reflects the total number of map blocks in use. Field length = 1H.
	DQE.CMS	Number of swappable class S map blocks currently allocated; for resident tasks, if not zero, reflects the total number of map blocks in use. Field length = 1H.

2.13 Dispatch Queue Address Table (DAT)

The Dispatch Queue Address Table (DAT) is a variable length table built by the system generation program. It contains a maximum of 255 single word entries. It is accessed by the word adjusted DQE entry number, and contains the address of the associated DQE in the CPU Dispatch Queue Area. The address of the DAT minus one word is contained in C.ADAT. The number of DAT entries is contained in C.NQUE and is equal to the number of DQEs.

2.14 File Assignment Table (FAT)

The File Assignment Table (FAT) is used to provide an association between a logical file code (LFC) and a resource. It also coordinates access to the resource referenced by an LFC. The FAT is linked to the Unit Definition Table (UDT) and the Controller Definition Table (CDT) when the resource is allocated.

The FAT must contain information related to the requestor of the resource, such as position within the file (segment and byte within the segment) and current access mode. For efficiency considerations, information pertaining to allowable access modes, segmentation, and extendibility are also included.

Word	0 7	8 15	16 23	24 31
0	Status bits (DFT.STB). See Note 1.	Access flags or system file code (DFT.ACF). See Note 2.	CDT index (DFT.CDTX)	
1	Flags (DFT.FLGS). See Note 3.	Number of FPTs assigned (DFT.NAS)	UDT index (DFT.UDTX)	
2	Segment definition area address (DFT.SEGA) or Volume name for dismount message (DFT.VNAM)			
3	Relative file block position (DFT.POS)			
4	Relative EOM block position (DFT.EOM). See Note 4.			
5	Relative EOF block number (DFT.EOF)			
6	Current segment position in file (DFT.CSEG) or device specification mask (DFT.MASK)		Number of segments (DFT.NSEG)	
7	Relative end block number of current segment (DFT.SEGE) or Unformatted medium identifier (MTF.REEL). See Note 5.		Append record pointer (DFT.AREC). See Note 5.	
8	File attributes field (DFT.ATTR). See Note 6.			
9	Append block number (DFT.ABLK) or volume number for multivolume media (MTF.VOL)			
10	Blocking buffer head cell address (DFT.BBA)			
11	Associated VAT index (DFT.VATX)	Number of opens on this FAT (DFT.OPCT)	Current access mode (DFT.CACM)	Resource type code (DFT.TYPE)
12	Address of parent directory resource descriptor (DFT.PDIR)			
13	Relative offset of parent directory entry (DFT.DOFF)			
14	Allocated resource table entry pointer (DFT.ARTA)			
15	Assigned access restrictions (DFT.ACCS). See Note 7.			

Notes:

1. Bits in DFT.STB are assigned as follows:

Bit	Meaning if Set
0	File open
1	File opened read/write
2	Permanent file
3	Blocking buffer output active
4	Unformatted medium
5	Volume resource
6	Read only access
7	TSM associated FAT

2. Bits 0-4 in DFT.ACF are assigned as follows:

   Volume resource only:

Bit	Meaning if Set
0-1	Reserved
2	$ read on SYC
3-4	Reserved

   Unformatted medium only:

Bit	Meaning if Set
0	Mount message has been inhibited or tape is shared
1	Multivolume tape
2	Mount message has been output
3	Tape at EOT
4	Tape at BOT

   Bits 5-7 in DFT.ACF apply only to volume usage and contain one of the following values:

Value	Meaning if Set
0	Not a system file
1	SYC file
2	Multivolume magnetic tape was generated by an MPX-32 release 3.3 or later source system
3	SLO file
4	SBO file

3. Bits in DFT.FLGS are assigned as follows:

Bit	Meaning if Set
0	Blocking buffer present
1	SMAP or DMAP assignment
2	Multivolume magnetic tape was generated by an MPX-32 release 3.3 or later source system
3	File has been assigned to the null device
4	This FAT entry is not in use
5	TSM I/O (task is swappable)
6	ANSI labeled tape assignment
7	Reserved

4. Byte 3 of word four contains tape density for high speed tape (DFT.DENS) and EOM does not apply (DFT.EOM).

5. For an ANSI labeled tape assignment, this address contains the six-character Volume Identifier (VID).

6. Bits in DFT.ATTR are assigned as follows:

Bit	Meaning if Set
0	File is automatically extendable
1	File is implicitly shared
2	File data has been modified
3	Unblocked specified at assignment
4	File opened for random access
5	File opened in blocked mode
6	Expanded FCB
7	Resource descriptor opened for modify
8	Current access mode specified at assignment
9	Resource to be marked blocked at close
10	Enqueue inhibit
11	Spool option requested
12	EOF update required
13	Reserved for IOCS
14	File assigned to nonpublic volume
15	Segmented file
16	Task in resource queue when deleted
17	The date and time of last change field in the resource descriptor is not changed on a rewrite
18	Data in file is blocked (DFT.BLKD)
19-31	Reserved

7. Bytes in DFT.ACCS are assigned as follows:

Byte	Definition
0-1	Bit pattern from RR.ACCS if specified at assignment (see Section 2.31 for details on RR.ACCS). If not specified, the bit pattern is from the appropriate access restriction field (RD.AOWNR, RD.AUGRP, RD.AOTHR) in the resource descriptor. See M.RDCOM, Section 2.41.1 for details.
2	Bits are assigned as follows:

Bit	Meaning if Set
0	Assigned for explicit shared use
1	Assigned for exclusive use
2-7	Reserved

3	Bit pattern specified in byte three of RD.SFLGS in the associated resource descriptor. See M.RDSPD, Section 2.41.2 for details.

2.15 File Control Block (FCB)

The File Control Block (FCB) is used to convey information about requested I/O operations and to report their status to the requestor. The table entry is generally located in the task's address space.

The task's FCB is linked to the File Assignment Table (FAT) when the resource is opened. This completes the logical connection from the task to the requested resource for subsequent use. The FCB is then linked to an I/O Queue (IOQ) entry when an operation for that logical connection is requested. When this is done, the status for the requested operation code is posted in the respective FCB.

Word	0	1 3	4 7	8 11	12 15	16 23	24 31
0	*	Opcode (FCB.OPCD)		Logical file code (FCB.LFC)			
1	Quantity (FCB.TCW)				Data address		
2	General control flags (FCB.GCFG)			Special flags (FCB.SCFG)	Random access address (FCB.CBRA)		Console teletype flag (FCB.CONF)
3	Status flags (FCB.SFLG)			Test status	DCC status	Device status	
4	Record length (bytes) (FCB.RECL)						
5	Reserved			I/O queue address (FCB.IOQA)			
6	Special status (FCB.SPST)		Reserved	Wait I/O error return address (FCB.ERRT)			
7	Index to FPT			FAT address (FCB.FATA)			
8	Reserved			Expanded data address (FCB.ERWA)			
9	Expanded transfer quantity (bytes) (FCB.EQTY)						
10	Expanded random access address (FCB.ERAA)						
11	Extended I/O status word one (FCB.IST1)						
12	Extended I/O status word two (FCB.IST2)						
13	Reserved			No-wait I/O normal end-action service address (FCB.NWOK)			
14	Reserved			No-wait I/O error end-action service address (FCB.NWER)			
15	User-supplied blocking buffer address and number of 192 word buffers (FCB.BBA)						

* Reserved

Word 0

Bit 0 — This field is the system flag. This bit is set when an execute channel program has a postprogram control interrupt.

Bits 1-7 — Operation code - a hexadecimal digit specifies the type of function requested of the device handler. The allowable functions and their definitions are unique to each peripheral device.

Bits 8-31 — Logical file code - any combination of three ASCII characters is allowed.

Word 1

Note: Words 8 and 9 are used instead of word 1 if bit 6 of word 2 is set.

Bits 0-11 — Quantity - three hexadecimal digits specify the number of data items to be transferred. This quantity must include the carriage control character, if applicable. The transfer quantity is in units determined by the address in bits 12 to 31.

Bits 12, 30-31 — Format code - these bits specify byte, halfword, or word addressing for data transfers. They are interpreted as follows:

Type of Transfer	F (12)	C (30,31)
Byte 0	1	00
Byte 1	1	01
Byte 2	1	10
Byte 3	1	11
Left halfword	0	01
Right halfword	0	11
Word	0	00

If a halfword or word transfer is specified for a device which accepts only bytes, IOCS adjusts the quantity accordingly. If a byte transfer is specified for a device which accepts only halfwords or words, IOCS will adjust the quantity accordingly if the number of bytes is an even multiple of the requested transfer mode and the data address is on the correct boundary. Otherwise, the request is treated as a specification error.

Bits 13-29 — Data address - the initial address data areas for read or write operations.

Word 2

Bits 0-7 General control flags - these eight bits enable the user to specify the manner in which an operation is to be performed by IOCS. The interpretation of these bits is shown below:

Bit	Meaning if Set
0	IOCS returns to the user immediately after the I/O operation is queued. If reset, IOCS exits to the calling program only when the requested operation has been completed.
1	Error processing is not performed by either the device handler or IOCS. An error return address is ignored and a normal return is taken to the caller; however, the device status is posted in the FCB unless bit three is set. If reset, normal error recovery is attempted. 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 and the error return address is taken if provided. Otherwise, the task is aborted.
2	Data formatting is inhibited. Otherwise, data formatting is performed by the appropriate device handler. See bit eight for further explanation.
3	Device handlers perform no status checking and no status information is returned. All I/O appears to complete without error. Otherwise, status checking is performed and status information is returned as necessary.
4	File accessing occurs in the random mode. Otherwise, sequential accessing is performed.
5	Blocked file is specified (disc or tape assignments only).
6	Expanded FCB is present (words 8 to 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	This bit is reserved for internal IOCS use. It indicates the user's FCB is being used for physical I/O during blocked data handling and the FCB parameters are in the task's scratchpad.

8 Device format definition. Special definitions for 7-track magnetic tape, ALIMs, etc. are indicated in bits 9 to 12. Normally, bit 8 is examined only when bit 2 (data formatting inhibit) is set. The meaning is interpreted as shown in Table 2-1.

9-12 Special control specification. This field contains device control specifications unique to certain devices. Interpretation and processing of these specifications are performed by the device handlers. A bit setting is meaningful only when a particular type of device is assigned as indicated in Table 2.1, columns two and three. Column one indicates default control.

13-31 Random access address. This field contains a block number (zero origin) relative to the beginning of the disc file, and specifies the base address for read or write operations.

If bit 6 of word 2 is set, the expanded random access address in word 10 (FCB.ERAA) is used instead of bits 13 through 31.

For devices where random access is invalid, bits 13 through 31 have the following assignments:

Bits	Meaning if Set
13	Software read flow control required (FCB.RXON)
24-31	Console teletype type control parameter block flag type.

For High Speed Data (HSD) interface applications, word two bit meanings are as follows:

Bits	Meaning
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.
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.
10	Disable time out 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.
11	Set UDDCMD from least significant byte of word two. 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.
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 2-1. (Page 1 of 2)
Special Control Flags

Device	Default (Bit 2=0)	Override (Bit 2=1)	Bit 8	Bit 9	Bit 10	Bit 11	Bit 12
Card reader (CR)	Read 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 1 = binary read				
Line printer (LP)	Interpret first character as carriage control	No carriage control interpretation					
High speed data (HSD) interface (generic handler)			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				
(MT 9-track only)	N/A						
Discs (DM,DF,FL)	Report EOF if X'0FE0FE0F' encountered in word 0 of 1st block during read of unblocked record	No EOF reporting on unblocked reads					

Table 2-1. (Page 2 of 2)
Special Control Flags

Device	DFI (Bit 2=0)	DFI (Bit 2=1)	DFD Bit 8	NOECHO Bit 9	ULCONV Bit 10	RECD Bit 11	PURGE Bit 12
ALIM (Asynchronous Line Interface Module) Terminals (TY)	Read: receive data (bytes) defined for transfer count Write: formatted	Bit 2 / Bit 8 / Bit 9 Read: 0 / 1 / 0 = Blind mode reset 0 / 0 / 1 = Echo on read 1 / N/A / N/A = Receive data 0 / 0 / 0 = Receive data Write: 0 / N/A / 0 = Formatted write 0 / N/A / 1 = Initialize device 1 / N/A / N/A = Unformatted write			On read: 1 = Inhibit conversion of lower case characters to upper case 0 = Convert		
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 no form control	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 character to upper case 1 = Inhibit conversion of lower case characters to upper case.	Read: 0 = No special character detect 1 = Special character detect Write 0 = Normal write 1 = Write with input subchannel monitoring plus software flow control	Read: 0 = Do not purge type ahead buffer 1 = Purge type ahead buffer

Word 3

Bits	Meaning
0-31	Status word - 32 indicator bits are used by IOCS to indicate the status, error, and abnormal conditions detected during the current or previous operation. The assignment of these bits is shown below:

Bit	Meaning if Set
0	Operation in progress. Request has been queued. (Note: Reset after post I/O processing complete.)
1	Error condition found
2	Invalid blocking buffer control pointers have been encountered during file blocking or unblocking
3	Write protect violation
4	Device inoperable
5	Beginning-of-medium (BOM) (load point) or illegal volume number (multivolume magnetic tape)
6	End-of-file
7	End-of-medium (end of tape, end of disc file)

Nonextended I/O devices only:

Bits	Definition
8-11	Specifies general testing status as received from an 8000 level test device instruction
12-15	Specifies DCC testing status as received from a 4000 level test device instruction
16-31	Specifies a device status as received from a 2000 level test device instruction. These bits are not applicable for the paper tape, card reader, and teletypewriter. Bit meanings for 2000 level testing for nonextended I/O devices are shown in Table 2-2.

Table 2-2
Device Status (2000 Level) Nonextended I/O

FCB Word 3 Bits	16	17	18	19	20	21	22	23	24
All F-class devices	Echo	PPCI	Incorrect length	Program check	Data check	Control check	Interface check	Channel chaining check	Busy
High speed data interface (generic handler) D class	CD termination	Error status format (See word 3 description)							
GPMC	Device dependent								

FCB word 3 bits	25	26	27	28	29	30	31
All F-class devices	Status modifier	Controller end	Attention	Channel end	Device end	Unit check	Unit exception
High speed data interface (generic handler) D class	External termination	IOCB address error	Error on TI address fetch	Device EOB	EP5 Error precluded request queueing	Nonexecutive channel program IOCB type in error 00-Data transfer 01-Device status 10-command transfer	
GPMC	Device dependent	Transmission error	Incorrect length	Unusual end	Illegal order	Interrupt pending	Channel end

Extended I/O Devices Only:

Bits	Definition
8	Zero
9	Zero
10	Last command exceeded time-out value and was terminated
11-15	Zero
16-23	Channel status bits assigned as follows:

Bits	Meaning if Set
16	Echo
17	Post program controlled interrupt
18	Incorrect length
19	Channel program check
20	Channel data check
21	Channel control check
22	Interface check
23	Chaining check

24-31 Controller/device status bits assigned as follows:

Bit	Meaning if Set
24	Busy
25	Status modified
26	Controller end
27	Attention
28	Channel end
29	Device end
30	Unit check
31	Unit exception

Word 4

Bits	Definition
0-31	Record length - this field is used by IOCS to indicate the actual number of bytes transferred during read/write operations. If an error occurs during an execute channel request, this word contains the residual transfer count from the request.

Word 5

Bits	Definition
0-7	Reserved
8-31	I/O queue address - this field is set by IOCS to point to the I/O queue for an I/O request initiated from this FCB

Word 6

Bits	Definition
0	No-wait normal end action not taken
1	No-wait error end action not taken
2	Kill command, I/O not issued
3	If set, exceptional condition has occurred in the I/O request
4	If set, software read flow control required
5-7	Reserved
8-31	Wait I/O error return address - this field is set by the user and contains the address to which control is to be transferred in the case of an unrecoverable error when control bits one and three of word two are reset. If this field is not initialized and an unrecoverable error is detected under the above conditions, the user is aborted.

Word 7

Bits	Definition
0-7	Index to FPT - this field points to the last FPT in the File Pointer Table (FPT)
8-15	FAT address - this field points to the File Assignment Table (FAT) entry associated with all I/O performed for this FCB. This field is supplied by IOCS.

Note: Words 8 through 15 are valid only if bit 6 of word 2 is set.

Word 8

Bits	Definition
0-7	Reserved
8-31	Expanded data address - start address of data area for read or write operations. Must be a word address. (or) Expanded data/command chain address - word address that points to the data or command chain list if using execute channel entry point (H.IOCS,10)

Word 9

Bits	Definition
0-31	Expanded quantity - number of bytes of data to be transferred (or) For GPMC devices which support data/command chaining - expanded number of data/command chain doublewords. (or) For XIO requests for data/command chaining (execute channel H.IOCS,10), it is used to indicate the number of bytes of sense (bits 0 to 7) and user-specified sense buffer address (bits 8 to 31).

Word 10

Bits	Definition
0-31	Expanded random access address - this field 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 (or) For High Speed Data (HSD) Interface requests in nonexecute channel program format - this word defines a device command

Word 11

Bits	Definition
0-31	Status word one - for extended I/O, these are the 32 bits returned by the sense command (or) For communications adapter interface - external asynchronous interrupt (EAI) status if bit 12 of word two is set

Word 12

Bits	Definition
0-31	Status word two - second status word as returned from the extended I/O hardware (or) For High Speed Data (HSD) Interface applications - this word contains status sent from the user's device

Word 13

Bits	Definition
0-7	Reserved
8-31	No-wait I/O normal completion service address return. This user service must be terminated by calling H.IOCS,34 (no-wait I/O end action return) (or) For High Speed Data (HSD) interface applications - this address plus one word is the location where control is transferred on asynchronous notification

Word 14

Bits	Definition
0-7	Reserved
8-31	No-wait I/O error completion service address return - this user service must be terminated by calling H.IOCS,34 (no-wait I/O end action return)

Word 15

Bits	Definition
0-7	Number of 192 word buffers if specifying large blocking buffers. A value of one or zero in this field specifies one blocking buffer.
8-31	Blocking buffer address - defined for device-independent I/O or a post programmed controlled interrupt end-action receiver for device-dependent I/O

2.16 File Pointer Table (FPT)

The File Pointer Table (FPT) provides the linkage between the File Control Block (FCB) and the File Assignment Table (FAT). It also allows for multiple logical file code assignments to be made equivalent to the same FAT. The linkage to the FAT is performed at assignment. The linkage to the FCB is performed at opening. The FPT resides in the task's service area.

FPT entries one to six are reserved for the system as follows:

Entry 1 - System LFC *s*
Entry 2 - Load module LFC *LM
Entry 3 - H.VOMM resource descriptor LFC (1)
Entry 4 - H.VOMM directory LFC (2)
Entry 5 - H.VOMM DMAP/SMAP LFC (3)
Entry 6 - H.VOMM modify resource descriptor LFC X'FFFEE'

Each FPT entry has the following format:

Word	0 7	8 31
0	Reserved	Logical file code (FPT.LFC)
1	Flags (FPT.FLGS). See Note 1.	FCB address (FPT.FCBA)
2	Reserved	FAT address (FPT.FATA)

Notes:

1. Bits in FPT.FLGS are assigned as follows:

Bit	Meaning if Set
0	Reserved
1	Multiple FPT entries exist that point to the same FAT (i.e., $ASSIGN4 or $ASSIGN lfc TO LFC = lfc statements)
2	Reserved
3	FPT open
4	This FPT entry is not in use
5	Pseudo-SYC assignment (used by TSM)
6	Pseudo-FPT for unassigned temporary file
7	Reserved

2.17 I/O Queue (IOQ) Entry

The I/O Queue (IOQ) Entry is dynamically allocated from memory pool and contains information required to queue and process an I/O request. These entries are variable in length and support multiple device commands that are built starting at the end of the standard IOQ entry. The I/O queue consists of one or more I/O queue entries linked to either a Controller Definition Table (CDT) or a Unit Definition Table (UDT). See Figures 2-1 and 2-2.

Word	0 7	8 15	16 23	24 31
0	String forward address (IOQ.SFA)			
1	String backward address (IOQ.SBA)			
2	Queue priority (IOQ.PRI)	I/O type (IOQ.TYPE)	Channel number (IOQ.CHNO)	Subaddress (IOQ.SUBA)
3	Reserved (IOQ.RTN)			
4	PSD1 of task interrupt routine (IOQ.PSD). See Note 1.			
5	PSD2 of task interrupt routine			
6	Status (IOQ.STAT). See Note 2.	FCB or TCPB address (IOQ.FCBA)		
7	Program number (IOQ.PRGN)	CDT address (IOQ.CDTA)		
8	Handler function word one (IOQ.FCT1)			
9	Handler function word two (IOQ.FCT2). See Note 3.			
10	Handler function word three (IOQ.FCT3). See Note 4.			
11	Handler function word four (IOQ.FCT4)			
12	32-bit flag word (IOQ.FLGS). See Note 5.			
13	FAT address (IOQ.FATA)			
14	Number of bytes transferred (IOQ.UTRN). See Note 6.		Number of words in OS buffer (IOQ.WOSB). See Note 6.	
15	OS buffer address (IOQ.FBUF)			
16	User's buffer address (IOQ.TBUF)			
17	I/O returned status word one (IOQ.IOST)			
18	I/O returned status word two (IOQ.IST1). See Note 7.			
19	I/O returned status word three (IOQ.IST2). See Note 8.			
20	UDT address (IOQ.UDTA)			
21	Control information from word two of FCB (IOQ.CONT)			
22	Address of context block (IOQ.CBLK) or device context area address (IOQ.DCAA)			
23	Mode bits (extended I/O) (IOQ.MODE). See Note 9. (or) Word address of set mode bits (IOQ.MOWD)	Queue priority temporary storage (IOQ.PSAV)	Number of extra words in this queue entry (IOQ.XTRA)	
24	Device inoperable buffer address (for I/O error processing) (IOQ.INOP)			
25	Address of first word of dynamic IOCD list (extended I/O)(IOQ.IOCD). See Note 10.			

Notes:

1. For no-wait I/O, this field is set to point to the I/O postprocessing routine (S.IOCS1). When I/O completes, control is passed to this service.

2. Bits in IOQ.STAT are assigned as follows:

Bit	Meaning if Set
0	I/O queue is active. This bit is reset by the device handler when physical I/O transfer completes.
1	Sense command was issued on behalf of this I/O request (extended I/O)
2	Error retry was issued (rezero and retry entire IOCD list) (extended I/O)
3	Operator intervention required. Do not restart I/O
4	Reserved
5	Read ECC was issued (extended I/O)
6	Error retry was issued (retry entire IOCD list) (extended I/O). Backspace write or read sequence performed for extended I/O tape.
7	Reserved

3. For extended I/O devices, IOQ.FCT2 contains the 24-bit virtual address of the data or IOCL (bits zero to seven are zero).

4. For extended I/O devices, IOQ.FCT3 contains the adjusted byte transfer count in bits 0 to 31 (maximum is C.ADMASK plus one).

5. Bits in IOQ.FLGS are assigned as follows:

Bit	Meaning if Set
0	Multiplexed controller
1	OPCOM console request
2	TCW has been absolutized
3	IOQ will be linked to the UDT, not the CDT
4	Deallocate OS buffer
5	Extended I/O
6	Error found
7	System console queue
8	Data move required (OS to user buffer)
9	Rewind command in IOCD list for magnetic tape or reserve command in IOCD list for disc (extended I/O)
10	Nonexecute channel read command (extended I/O)
11	Nonexecute channel write command (extended I/O)
12	Special handler postprocessing required (handler EP6)
13	H.CT00 has been called with an FCB, not with a TCPB (i.e., not by H.IOCS,14)
14	Reserved
15	Terminal input
16	Terminal output
17	Task swappable during I/O
18	Release command in IOCD list for disc (extended I/O)
19	No-wait I/O (not TSM)
20	I/O restart entry

Bits	Meaning if Set
21	Nondevice access I/O performed
22	Kill command issued for this I/O request
23	Execute channel program (extended I/O)
24	User privileged
25	D-class controller (GPMC) only
26	Physical I/O performed for a user requesting blocked I/O
27-28	Reserved
29	EOF testing required for disc
30	Movement in file is in negative direction
31	Continuous EOF search (disc and floppy disc only)

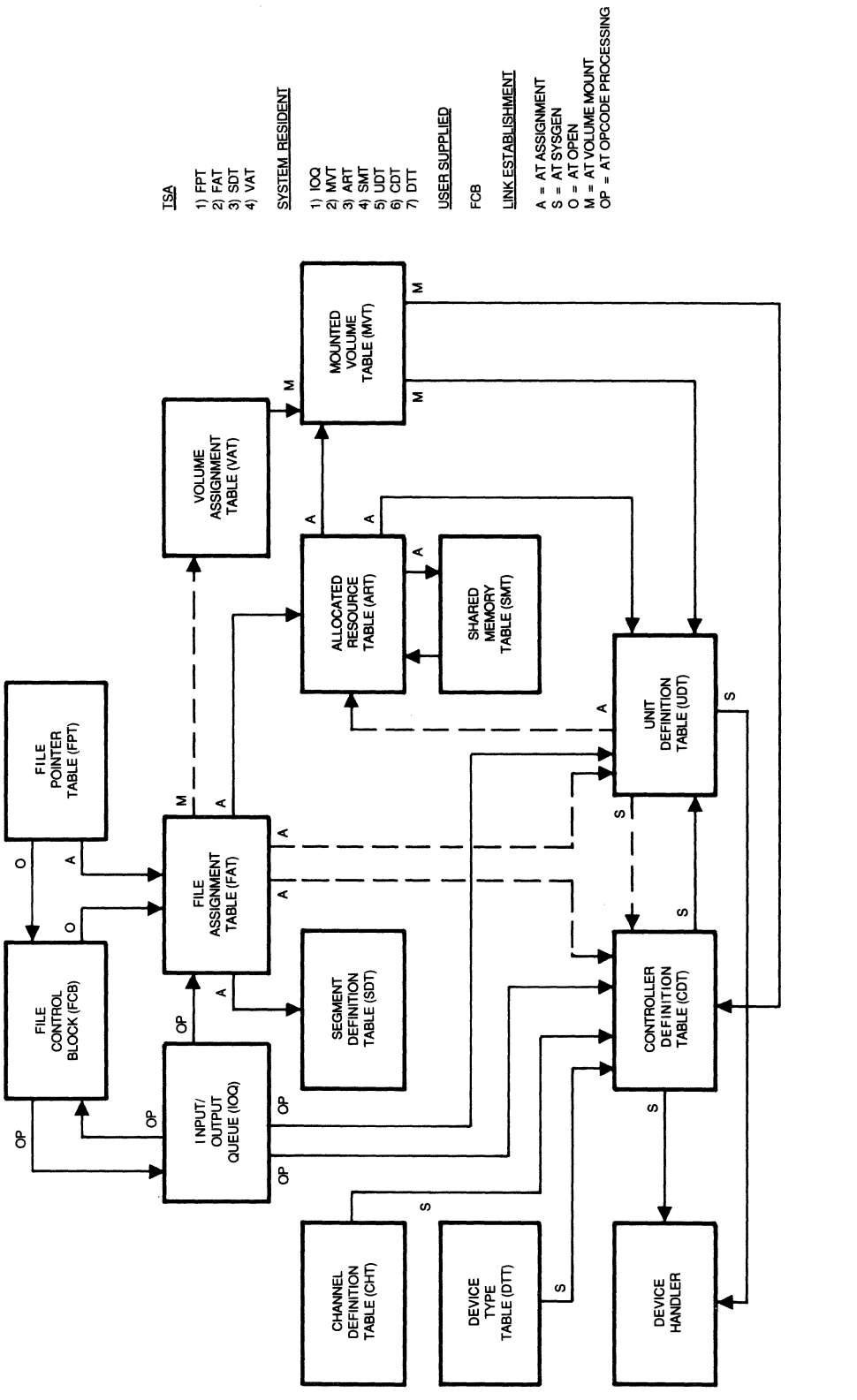
6. For extended I/O devices, IOQ.UTRN is a full word (bits 0 to 31) and IOQ.WOSB is not applicable.

7. For extended I/O devices, IOQ.IST1 is initialized to the start address within the I/O queue for any dynamic IOCDs.

8. For extended I/O devices, IOQ.IST2 is initialized to the stop address within the I/O queue for any dynamic IOCDs.

9. Mode bits are peculiar to each device.

10. This field contains the absolute data or IOCL address associated with the I/O request.



830659A

Figure 2-1. I/O Table Linkages

For F-class devices, the device specific tables in MPX-32 mirror the hardware configured at SYSGEN time as illustrated in the following figure. For each I/O channel configured, there is a corresponding Channel Definition Table (CHT). For each controller on that channel, there is a Controller Definition Table (CDT) linked to that CHT. For each device on each controller, there is a Unit Definition Table (UDT) linked to the corresponding CDT. Handler specific device information is retained in the Device Context Area (DCA) that is then linked to the corresponding UDT.

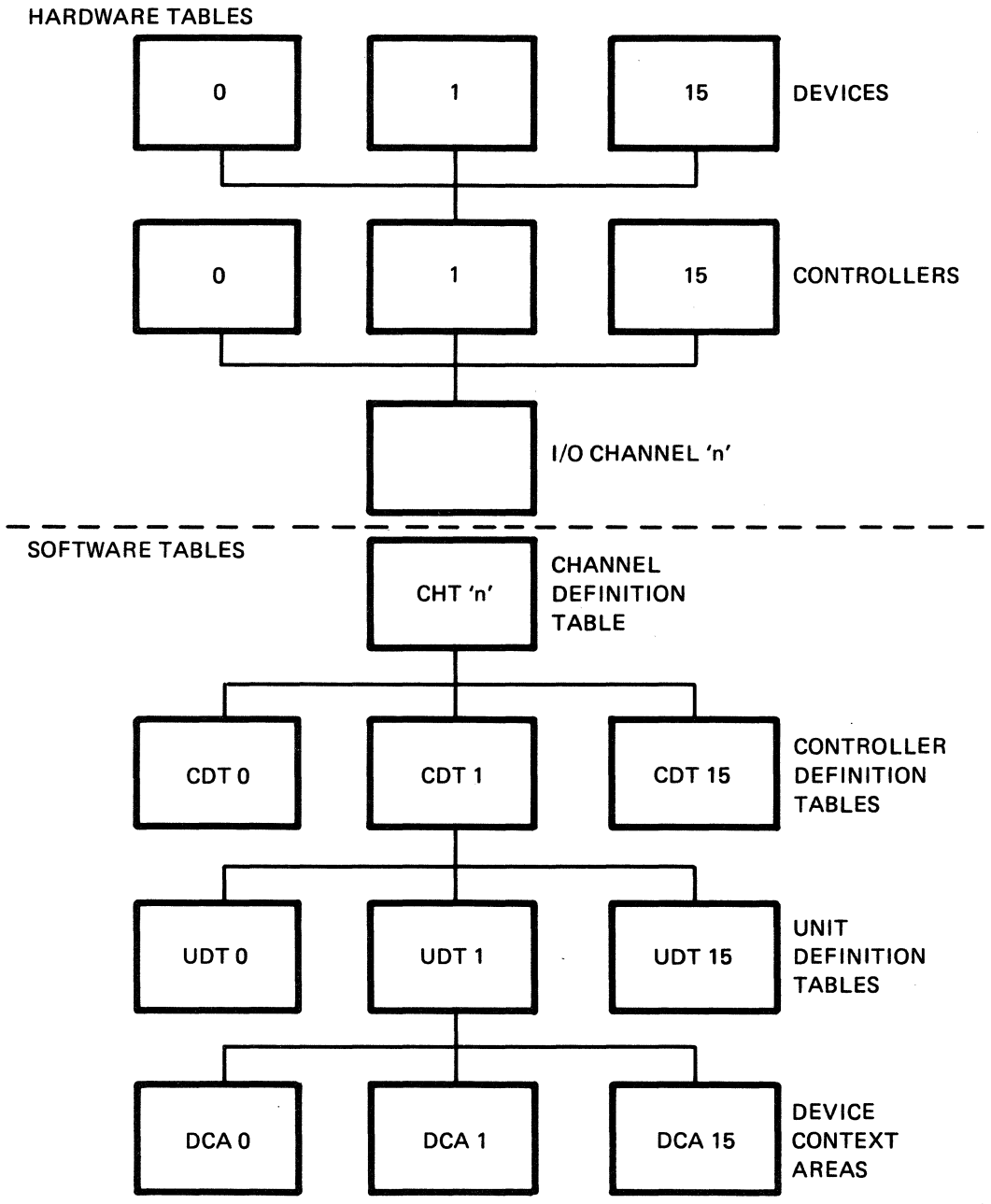
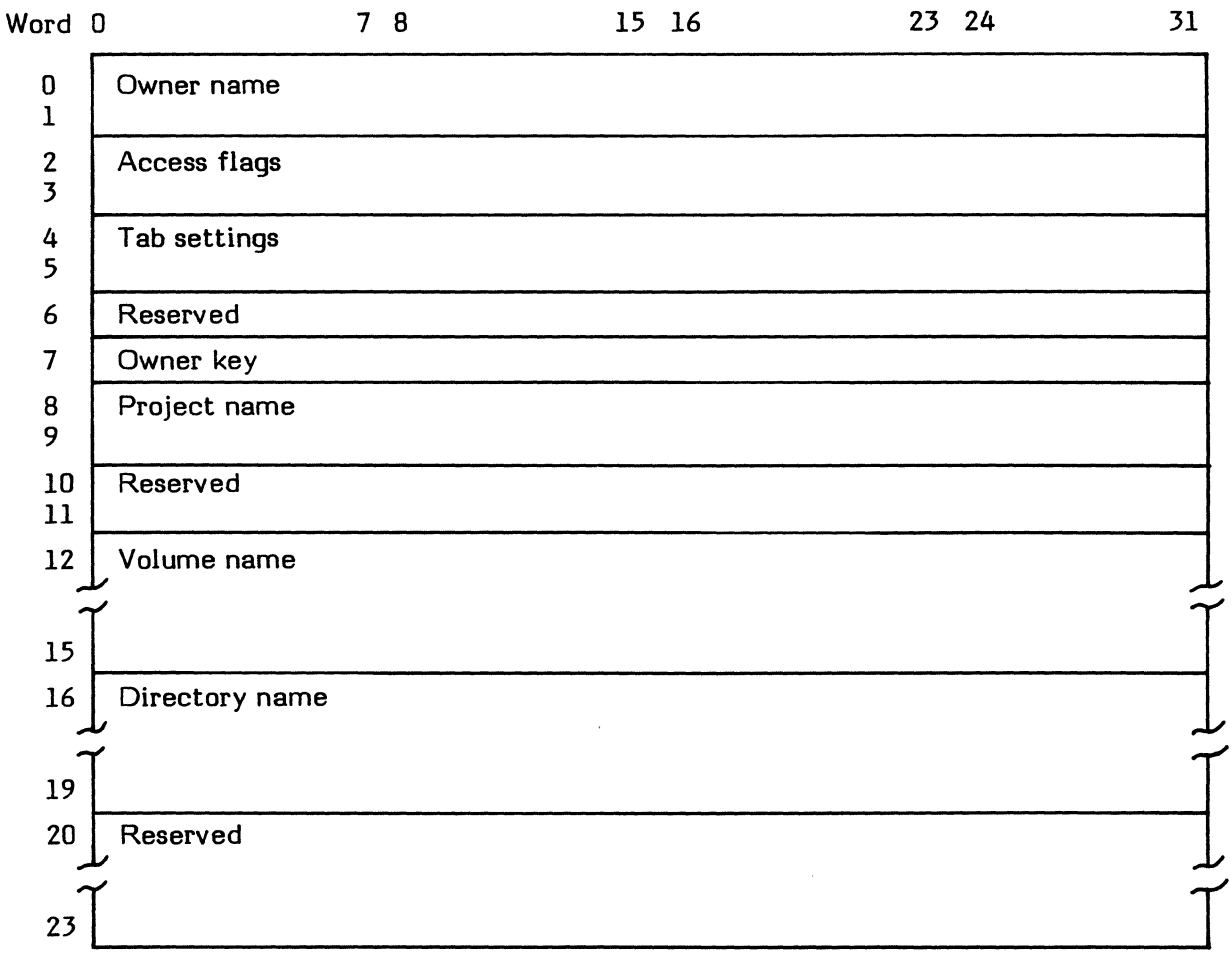


Figure 2-2. Handler Tables and Corresponding Hardware

2.18 M.KEY Entry Format

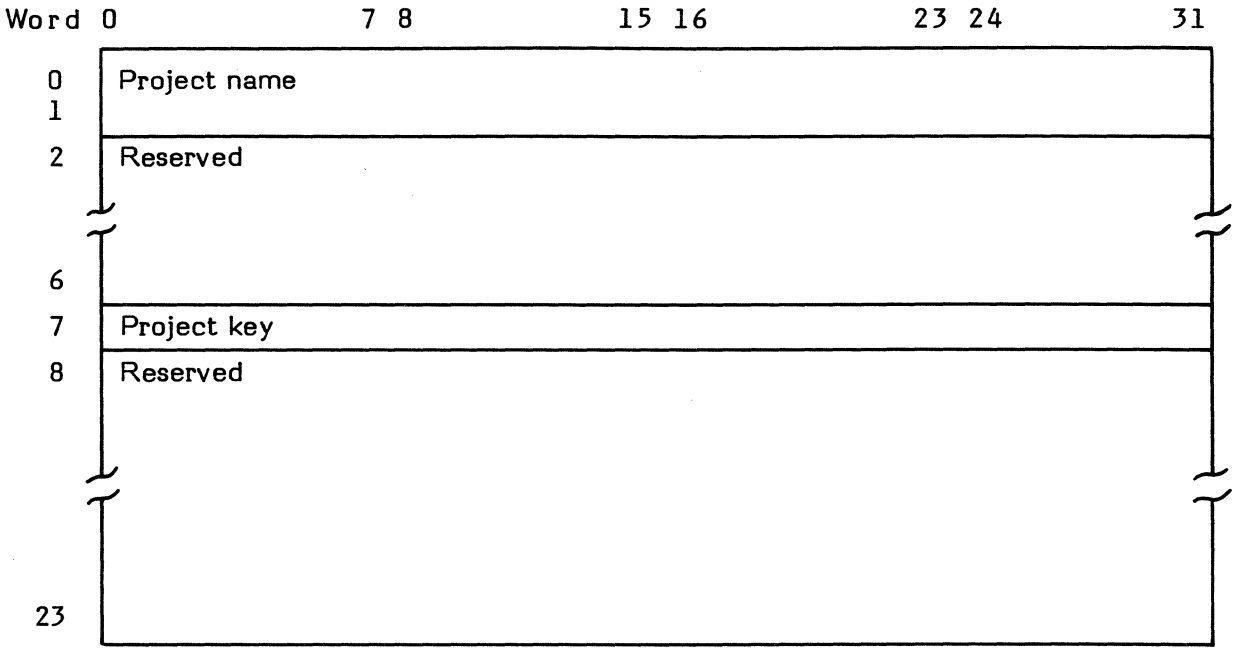
The M.KEY entry file is built by the KEY editor and interpreted by J.TSM. It is an unblocked, nonextendible file. Blank fields default to SYSTEM. All fields are in ASCII except the following:

- Access flags - See Task Service Area (TSA) description of T.ACCESS for bit assignments.
- Tab settings - One byte per tab position. A zero indicates end of tabs. Maximum of eight tab positions.
- Key - Compressed key associated with owner name. Compression is done by H.FISE,8. A zero indicates no key.



2.19 M.PRJCT Format

The M.PRJCT file is built by J.PRJCT and is interpreted by J.TSM. It is an unblocked, nonextendible file. The project key is compressed by H.FISE,8. A zero indicates no key.



2.20 Map Image Descriptor List (MIDL)

The Map Image Descriptor List (MIDL) entries are halfwords that contain the physical map block numbers that correspond to a task's logical map blocks. The MIDL size varies depending on the type of task. The contents of T.MIDLA point to the first MIDL entry.

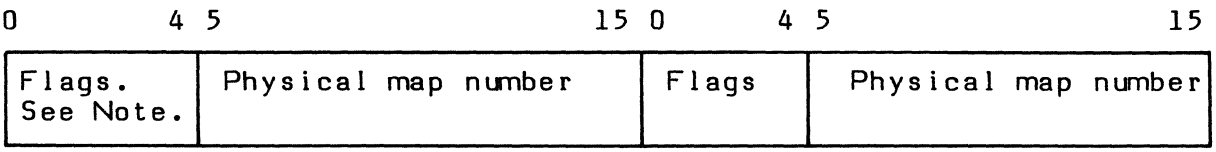
For nonbase mode tasks, the maximum size of the MIDL is 256 maps minus the operating system size in maps.

For base mode tasks, the size of the MIDL is the program's size, in maps, plus 16 maps if the task is greater than 2MB. If not, the MIDL size is 256 maps.

If the TSM $SPACE command is used, the size of the MIDL is the logical address space, specified in maps, minus the operating system size in maps.

The Memory Attribute List (MEML) entries correspond one to one to the MIDL entries. The MEML describes the attributes associated with each map block.

MIDL entries have the following format:



Note: Flag bits are assigned as follows:

Bit	Meaning if Set
0	Map number is valid (MIDL.VAL)
1	First protection granule is write protected (MIDL.PRO)
2	Second protection granule is write protected (MIDL.PR2)
3	Third protection granule is write protected (MIDL.PR3)
4	Fourth protection granule is write protected (MIDL.PR4)

If bit zero (MIDL.VAL) is set, the physical map number contains a valid map block number that represents an entry in the Memory Allocation Table (MATA).

2.21 Memory Allocation Table (MATA)

The Memory Allocation Table (MATA) contains the current status of each 2KW map block of main memory that is present in a configuration. The address of this table is contained in C.MATA.

Each MATA entry consists of a flag byte representing the status of a configured map block. There is one flag byte for every map block configured. The flag bytes are positional, relative to the first block of the configured class of memory.

0 7 8 15 16 23 24 31

Number of map blocks configured in system (MEM.CNT)		Starting map number for memory table (MEM.SMN)	
Flag bits (MEM.STAT). See Note.	MEM.STAT	MEM.STAT	MEM.STAT
MEM.STAT	MEM.STAT	etc.	etc.

Note: Flag bits in MEM.STAT are defined as follows:

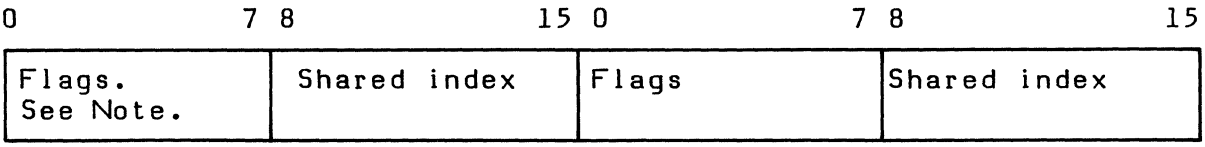
Bit	Meaning if Set
0	Map block is allocated (MEM.ALL)
1	Map block is shared (MEM.SHR)
2	Map block is multiprocessor shared (MEM.PRO)
3	Malfunction exists (MEM.MAL)
4	Map block is nonpresent (MEM.CON)
5	Map block is semiconductor (MEM.TYP)
6	Defined below (MEM.CL1)
7	Defined below (MEM.CL2)

Bit 6	Bit 7	Class
0	0	E
0	1	H
1	0	S
1	1	Undefined

2.22 Memory Attribute List (MEML)

The Memory Attribute List (MEML) entries correspond one-to-one to the Map Image Descriptor List (MIDL) entries. The MEML describes the attributes associated with each map block. The maximum number of MEML entries is the same as MIDL entries. See Section 2.20. The contents of T.MEMLA point to the first MEML entry.

MEML entries have the following format:



Note: Flag bits are assigned as follows:

Bit	Meaning if Set
0	E-type memory (MEML.TYE)
1	H-type memory (MEML.TYH)
2	S-type memory (MEML.TYS)
3	Map is shared (MEML.SHR)
4	Map is swappable (MEML.SWP)
5	Map is valid (MEML.VAL)
6	Map block used by system (MEML.SYS)
7	Map is outswapped (MEML.OUT)

If bit three (MEML.SHR) is set, the shared index contains the index into the associated shared memory table where the map block has been allocated.

2.23 Memory Pool Management

Memory pool is an area of main memory beginning at the high address end of resident MPX-32. Its size is specified at SYSGEN and it occupies an area up to the next map block boundary. It is used as temporary storage space by various system services. It contains, at any one time, line buffers, I/O queues, messages, IOCD lists, etc., that are dynamic and not predefined in size. C.SBUF contains the address of memory pool. C.SBUF+1W contains the number of words in memory pool.

Areas within memory pool are allocated in multiples of two words (doubleword bounded). A free list and allocated list of buffer areas are maintained. They are in double-linked list format and are linked to head cells located in S.MEMM9 to control allocation and deallocation of memory pool areas. Entries are linked in ascending memory address order.

An allocation request for memory pool is obtained from the first free space large enough to satisfy the request. A four-word header is built and linked by address to the allocated head cell. The free area entry header is updated to reflect the size reduction caused by the requested allocation.

A deallocation request for memory pool causes the allocated list to be searched for the allocated entry. Verification is made on the buffer address and size provided by the caller. If valid, the entry is moved to the free list and agglomeration attempted with previous and next entries.

When the debugger and event trace are present in the system, the debugger prompt is displayed if an abnormal condition is detected when using the S.MEMM10 system subroutine to deallocate memory pool. The following register definitions are displayed in response to the debugger DR command:

Register	Contents
1	Abort condition as follows:

Abort Condition	Description
1	Buffer address in register three is not in memory pool
2	Buffer address in register three is not allocated
3	Invalid byte count is specified in the deallocation request

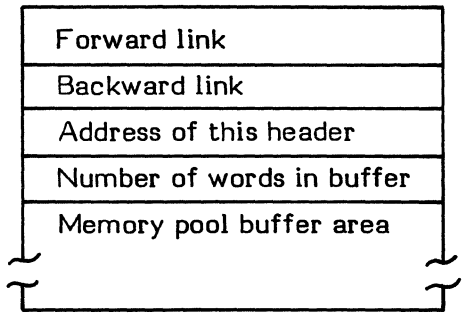
2	Buffer header address if register one is three
3	Buffer address from the caller if register one is one or two
4	Free head cell address if register one is one or two
7	Invalid byte count from the caller if register one is three

If the debugger TE command is entered to the debugger prompt and register one is one or two, the request is ignored.

If the debugger TE command is entered to the debugger prompt and register one is three, the request is continued using the corrected count obtained from the buffer header that was originally allocated (register two address plus three words).

The address in register two points to the following structure:

Buffer Header Entry

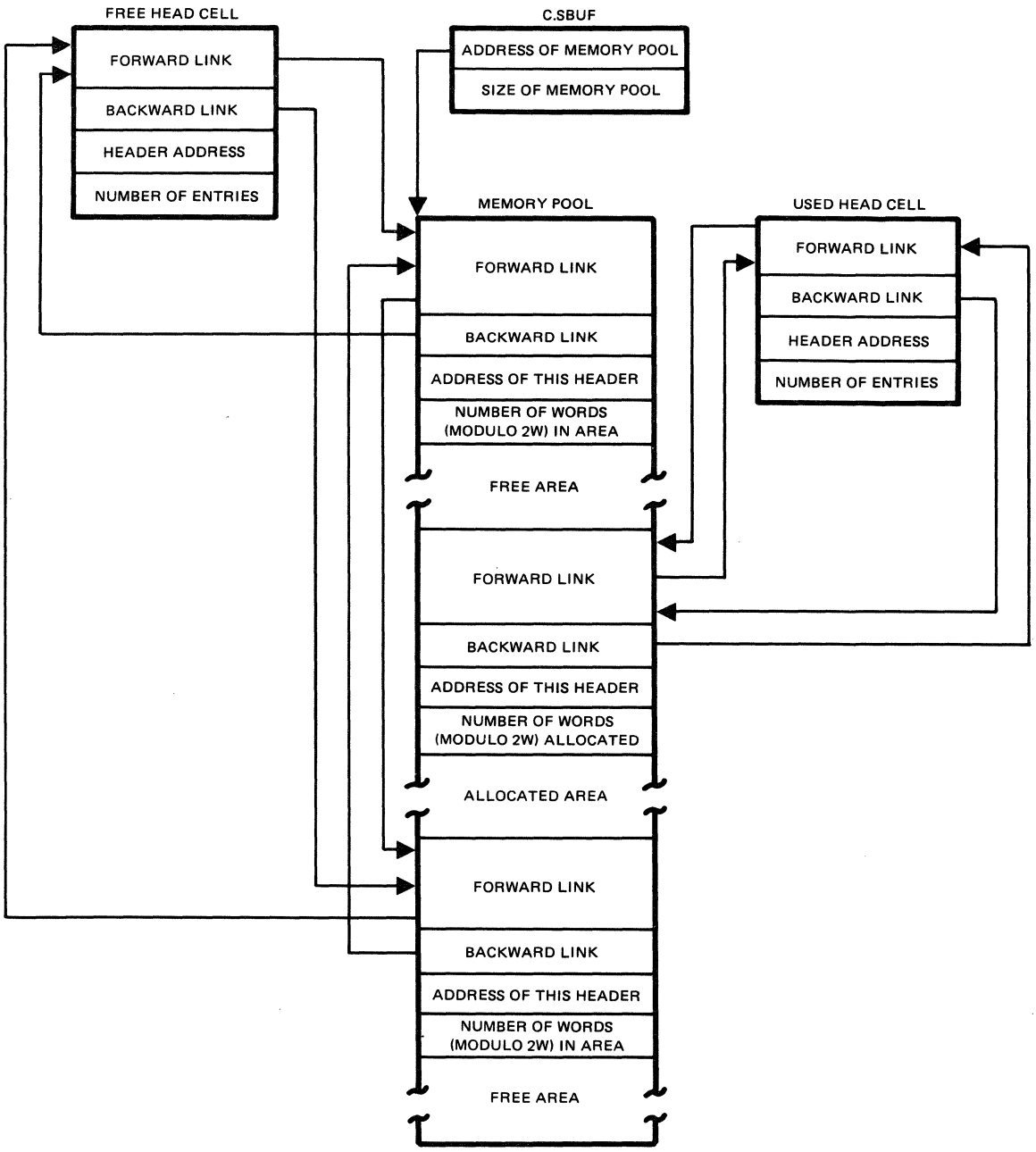


Forward link	Link address to next buffer header entry
Backward link	Link address to last buffer header entry
Address of this header	Address of this buffer header entry
Number of words in buffer	Number of words, excluding header, in this area
Memory pool buffer area	User buffer area

The address in register four points to the following structure:

Free Head Cell

Forward link	Next free entry address
Backward link	Last free entry address
Address of this header	Address of this head cell
Number of entries in list	Number of entries in free list

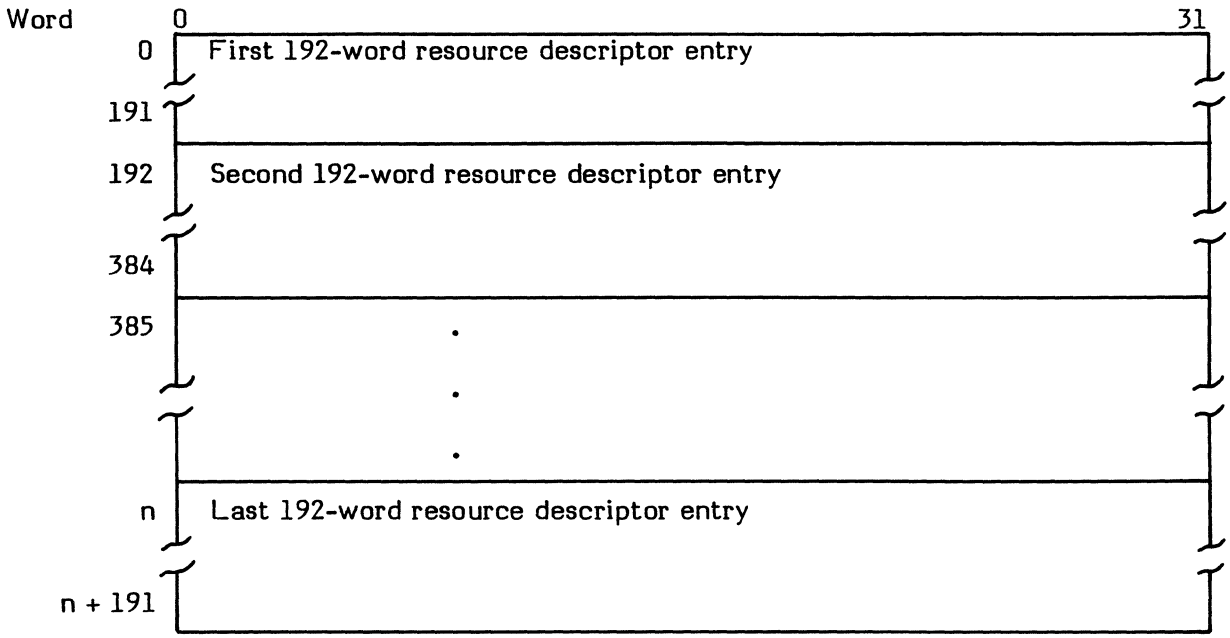


840817B

Figure 2-3. Memory Pool Diagram

2.24 Memory Resident Descriptor Table (MDT)

The Memory Resident Descriptor Table (MDT) is a table of resource descriptors that resides in main memory. The resource descriptors in the MDT are exact copies of the resource descriptors that reside on the disc.



2.25 Message or Run Request Queue (MRRQ)

This parameter block is created by H.EXEC by a user generated Parameter Send Block (PSB). It is used for message and run request processing.

Word	0 7	8 15	16 23	24 31
0	String forward address (MQ.SF)			
1	String backward address (MQ.SB)			
2	Priority (MQ.PR)	Parameter send block address (MQ.PSBA)		
3	Task number of sending task (MQ.TNST)			
4 5	Postprocessing service parameter send block (MQ.PPSD)			
6	Passed parameter quantity (MQ.PPQ)		Sending task return buffer length (MQ.RBL)	
7	Completion staus (MQ.CST). See Note 1.	Initial status (MQ.IST). See Note 2.	User status (MQ.UST)	Options (MQ.OPT). See Note 3.

Notes:

1. Bits in MQ.CST are the same as the Parameter Send Block (PSB) format and are assigned as follows:

Bits	Meaning if Set
0	Operation in progress (PSB.IOP)
1	Destination task aborted (PSB.DTA)
2	Destination task deleted (PSB.DTD)
3	Return parameters truncated (PSB.RPT)
4	Send parameters truncated (PSB.SPT)
5	End action routine not processed (PSB.EANP)
6-7	Reserved

2. Codes in MQ.IST are the same as the Parameter Send Block (PSB) format and are assigned as follows:

Bits	Meaning if Set
0	Normal initial status
1	Task number invalid (PSB.TSKE)
2	Load module name error (PSB.LMNE)
3	Reserved
4	Load module format error (PSB.LMFE)
5	DQE space unavailable (PSB.DQEE)

Bits	Meaning if Set
6	I/O error reading directory (PSB.SMIO)
7	I/O error reading load module (PSB.LMIO)
8-9	Reserved
10	Invalid priority (PSB.PRIE)
11	Invalid send buffer address (PSB.SBAE)
12	Invalid return buffer address (PSB.RBAE)
13	Invalid end action address (PSB.EAE)
14	Memory pool unavailable (PSB.MPE)
15	Destination task receiver queue full (PSB.DTQF)

3. Bits in MQ.OPT are the same as the Parameter Send Block (PSB) format and are assigned as follows:

Bits	Meaning if Set
24	No-wait mode (PSB.NWM)
25	No call-back mode (PSB.NCBM)
26-31	Reserved

2.26 Module Address Table

The Module Address Table contains the addresses of the resident system modules as they are referenced by the M.CALL macros. The address of this table is contained within C.MODD.

Word	0 7 8 15 16 23 24 31
0	Reserved
1	Address of H.EXEC
2	Address of H.MONS
3	Address of H.IOCS
4	Address of H.FISE
5	Address of H.ALOC
6	Address of H.MEMM
7	Address of H.TSM
8	Address of H.TAMM
9	Address of H.REXS
10	Address of H.REMM
11	Address of H.VOMM
12	Reserved for ACX

2.27 Mounted Volume Table (MVT)

The Mounted Volume Table (MVT) is a system resident table which is built at SYSGEN (its size is variable and determined by the SYSGEN process). The MVT holds entries that record the volumes currently mounted on the system. Each entry is identified by its name, and is associated with the device it is mounted on. The entry is created when the volume is initially mounted.

Subsequent users other than the initial requestor of a particular volume must also mount the volume even though the requested volume is currently mounted, unless it is a public volume. The subsequent requests cause the use count in the entry for that volume to be incremented by one. Each user dismounting the volume decrements the use count by one. The volume is then required to be physically dismounted (i.e., a dismount message issued) and the entry deleted when the use count is equal to zero (i.e., the volume is no longer in use).

Word	0 7 8 15 16 23 24 31
0	Volume name (MV.VOLNM)
3	
4	CDT address of volume device (MV.CDTA)
5	UDT address of volume device (MV.UDTA)
6	Current number of users of this volume (MV.USERS)
7	Current number of temporary files allocated (MV.TFILS)
8	Space map start address (MV.SMAPS)
9	Space allocation map length in blocks (MV.SMAPL)
10	Number of allocation units reflected in space map (MV.SMAPU)
11	Number of allocation units currently available (MV.SMAPC). See Note 1.
12	Descriptor allocation map start address (MV.DMAPS)
13	Descriptor allocation map length in blocks (MV.DMAPL)
14	Number resource descriptors in descriptor map (MV.DMAPU)
15	Number resource descriptors currently available (MV.DMAPC)
16	Root directory segment definition (MV.ROOTS)
17	Root directory segment definition (MV.ROOTL)
18	Blocks per allocation unit (MV.BLKAU)
19	Number of blocks in system area (MV.SYSBL)
20	Reserved
34	
35	Volume flags (MV.FLAGS). See Note 2.
36	Associated SYSID, if multiprocessor volume (MV.SYSID)
37	Port (MV.MPID). See Note 3. / Reserved
38	Reserved
39	

Notes:

1. This number includes some blocks allocated by the operating system.

2. Bits in MV.FLAGS are assigned as follows:

Bits	Meaning if Set
0	System volume (MV.SYS)
1	Operator mounted volume (MV.OPER)
2	Fixed media volume (MV.FIXED)
3	Public volume (MV.PUBLC)
4	Entry is active (in use) (MV.ACTV)
5	Mount in progress (MV.MNT)
6	Inhibit mount message (MV.NOMSG)
7	Swap volume (MV.SWP)
8	Volume device off-line (MV.OFFLN)
9	Volume not safe for use (MV.UNSAF)
10	Dismount pending (no mounts allowed) (MV.DMNT)
11	DMAP locked (MV.DLOCK)
12	Space map lock (MV.SLOCK)
13	Root directory locked (MV.RLOCK)
14	Multiprocessor volume flag (MV.DUALP)
15	Reserved
16	Reserved for future development (MV.PRIM)
17-26	Reserved
27	Mounted for shared use (MV.SHRBL)
28-31	Reserved

3. Port number under which multiport volume was mounted.

2.28 Resource Create Block (RCB)

The Resource Create Block (RCB) is a doubleword bounded data structure which defines the attributes of a resource (permanent file, temporary file, memory partition, or directory) created by a Volume Management Module (H.VOMM) entry point.

Word	0 7 8 15 16 23 24 31
0	Resource owner name (RCB.OWNR)
1	
2	Resource project group name (RCB.USER)
3	
4	Resource owner rights specifications (RCB.OWRI). See Note 1.
5	Resource project group rights specifications (RCB.UGRI). See Note 1.
6	Resource others rights specifications (RCB.OTRI). See Note 1.
7	File management flags (RCB.SFLG). See Note 2.
8	Maximum file extension increment (RCB.MXEI)
9	Minimum file extension increment (RCB.MNEI)
10	Maximum file size (RCB.MXSZ) (or) starting physical page (RCB.PPAG)
11	Original resource size (RCB.OSIZ)
12	Resource starting address (RCB.ADDR)
13	Resource fast ID location (RCB.FAST)
14	Option flags (RCB.OPTS). See Note 3.
15	Default override (RCB.FREE). See Note 4.

All RCB fields are optional. If the owner and project group names are not specified, the names default to those used by the calling task. All other fields not specified use the system defaults.

Notes:

1. Bits in RCB.OWRI, RCB.UGRI and RCB.OTRI are assigned as follows:

Bits	Meaning if Set
0	Read access (RCB.READ)
1	Write access (RCB.WRIT)
2	Modify access (RCB.MODI)
3	Update access (RCB.UPDA)
4	Append access (RCB.APPN)
5-7	Reserved
8	Traverse directory access (RCB.TRAV)
9	Delete resource access (RCB.DELE)
10	Delete directory entry access (RCB.DEEN)
11	Add directory entry access (RCB.ADEN)
12-31	Reserved

Bits 2, 3, and 4 are not valid for use with memory partitions.

2. Bits in RCB.SFLG are assigned as follows (for any bit not set, system defaults apply):

Bits	Meaning if Set
0-7	Resource type, equivalent to file type code, interpreted as two hexadecimal digits, 0 through FF (RCB.FTYP)
8-10	Reserved
11	File EOF management required (RCB.EOFM)
12	Resource fast access (RCB.FSTF)
13	Resource not to be saved (RCB.NSAV)
14	Reserved for MPX-32 usage
15	Reserved
16	File is executable (RCB.EXEC)
17	Owner ID set on access (RCB.OWID)
18	Project group ID set on access (RCB.UGID)
19	Reserved
20	Maximum file extension increment is zero. System default value not used. (RCB.MXEF)
21	Minimum file extension increment is zero. System default value not used. (RCB.MNEF)
22	Reserved
23	File zeroed on creation/expansion (RCB.ZERO)
24	File automatically extendible (RCB.AUTO)
25	File manually extendible (RCB.MANU)
26	File contiguity is desired (RCB.CONT)
27	Resource is sharable (RCB.SHAR)
28	Link access (RCB.LINK)
29-30	Reserved
31	File data initially recorded as blocked (RCB.BLOK)

3. Bits in RCB.OPTS are assigned as follows:

Bits	Meaning if Set
0	Owner has no access rights (RCB.OWNA)
1	Project group has no access rights (RCB.USNA)
2	Others have no access rights (RCB.OTNA)
3-7	Reserved
8	Spool file type (RCB.SPOO)
9-15	Reserved
16-23	Maximum segments at creation (RCB.SEGN)
24-31	Defines memory class:

Value	Class
0	S (default)
1	E
2	H
3	S

4. Bits in RCB.FREE are assigned as follows (these bits override any corresponding bit set in RCB.SFLG and the system defaults):

Bits	Meaning if Set
0-7	Must be zero
11	File EOF management not required
12	Fast access not required
13	Resource can be saved
23	Do not zero file on creation/extension
24	File is not automatically extendible
25	File is not manually extendible
26	File contiguity is not desired
27	Resource is not sharable
31	File data initially recorded as unblocked

2.29 Resource Inquiry Table (M.RIQ)

The Resource Inquiry Table (M.RIQ) contains information specific to an allocated resource. The information is returned in the form of a series of pointers to various data structures within the system which describe the resource. For memory partitions only, words zero and five apply. For volume resources, words two through four apply to the device where the volume is mounted.

Word	0 7 8 15 16 23 24 31
0	Address of allocated resource table entry (RIQ.ART)
1	Address of file assignment table entry (RIQ.FAT)
2	Address of unit definition table entry (RIQ.UDT)
3	Address of device type table entry (RIQ.DTT)
4	Address of controller definition table entry (RIQ.CDT)
5	Address of shared memory table entry (RIQ.SMT)
6	Address of file pointer table entry (RIQ.FPT)
7	Address of mounted volume table entry (RIQ.MVT)

2.30 Resource Logging Block (RLB)

The Resource Logging Block (RLB) is a word-bounded data structure used to pass information between H.VOMM and the caller. The information is used to locate a directory entry and resource descriptor for a single resource or for all resources defined in a particular directory.

Word	0 7	8 15	16 23	24 31
0	Pathname vector or RID address (RLB.TGT)			
1	Resource directory buffer address (192W) (RLB.BUFA). See Note 1.			
2	Associated mounted volume table entry address (RLB.MVTE)			
3	Parent directory RD block address (RLB.RDAD)			
4	Type (RLB.TYPE). See Note 2.	Buffer offset (RLB.BOFF)		
5	Length. See Note 3.	Return buffer address (RLB.DIRA)		
6	User FCB address (RLB.FCB)			
7	Flags. See Note 4.	Reserved (RLB.INT)		

Notes:

1. Optional. If not specified, a resource directory is not returned.

2. Bits in RLB.TYPE are assigned as follows:

Bits	Meaning if Set
0	Indicates recall (RLB.RECA)
1-7	Reserved

3. This word contains the address of a buffer and its length in words (the buffer can be up to 16 words long).

4. Bits in the flags byte are assigned as follows:

Bits	Meaning if Set
0-1	Reserved
2	Directory entry and resource descriptor for specified directory are returned
3	Reserved
4	Resource is located
5-7	Reserved

2.31 Resource Requirement Summary (RRS) Entries

The Resource Requirement Summary (RRS) is a doubleword bounded data structure used to identify the resources required by a task to the resource manager. Resources are statically allocated using the information in the RRS entry. The RRS is generally built by processors requiring static allocation of resources, such as TSM, cataloger, etc., or supplied as an argument for dynamic allocation.

For compatibility purposes, release 1.x RRS formats can be used. The details of these formats can be found in Chapter 2 of a release 1.x Technical Manual.

Type 1 - Assign by Pathname

Word	0 7	8 15	16 23	24 31
0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Plength (RR.PLEN)	Reserved
2	Access (RR.ACCS). See Note 2.			
3	Options (RR.OPTS). See Note 3.			
4	Pathname (variable length) (RR.NAME1)			
n				

Type 2 - Assign to Temporary File

Word	0 7	8 15	16 23	24 31
0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Initial file size (RR.PLEN)	
2	Access (RR.ACCS). See Note 2.			
3	Options (RR.OPTS). See Note 3.			
4	Volume name (16 characters; left-justified, blank-filled) (RR.NAME1) (Volume name is optional)			
7				

Type 3 - Assign to Device

Word	0 7	8 15	16 23	24 31
0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Density (RR.DENS). See Note 4.	Zero
2	Access (RR.ACCS). See Note 2.			
3	Options (RR.OPTS). See Note 3.			
4	Device type (RR.DT3). See Note 5.	Volume number (RR.VLNUM)	Channel number See Note 6. (RR.CHN3)	Subchannel number (RR.SCHN3)
5	Unformatted ID (1-4 characters) (RR.UNFID)			

Type 4 - Assign to LFC

Word	0 7	8 15	16 23	24 31
0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Zero	
2	Zero	Logical file code (RR.SFC)		
3	Options (RR.OPTS). See Note 3.			

Type 5 - Assign by Segment Definition

Word	0 7	8 15	16 23	24 31
0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	UDT index (RR.UDTI)	Reserved
2	Access (RR.ACCS). See Note 2.			
3	Options (RR.OPTS). See Note 3.			
4	Starting block number (RR.STBLK)			
5	Number of blocks (RR.NBLKS)			

Type 6 - Assign by Resource ID

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

Type 7 - Reserved for Future Use

Type 8 - Reserved for Future Use

Type 9 - Mount by Device Mnemonic

Word	0 7	8 15	16 23	24 31
0	Zero			
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Zero	
2	Access (RR.ACCS). See Note 2.			
3	Options (RR.OPTS). See Note 3.			
4	Volume name (16 characters; left-justified, blank-filled) (RR.NAME1)			
7				
8	Device type (RR.DT9). See Note 7.	Reserved	Channel number (RR.CHN9). See Note 8.	Subchannel number (RR.SCHN9)
9	Zero			

Type 10 - Assign to ANSI Tape

Word	0 7	8 15	16 23	24 31
0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Format (RR.FORM)	Protect (RR.PROT)
2	Access (RR.ACCS). See Note 2.			
3	Options (RR.OPTS). See Note 3.			
4	Record length (RR.RECL)		Block size (RR.BSIZE)	
5	Generation number (RR.GENN)			
6	Generation version number (RR.GENV)			
7	Absolute termination date (RR.EXPIA)			
8	Relative termination date (RR.EXPIR)		Logical volume identifier (RR.LVID)	
9	RR.LVID (cont.)			
10	17-character file identifier (RR.AFID)			
13				
14	RR.AFID (cont.)	Reserved		
15	Reserved			

Type 11 - Assign to Shadow Memory

Word	0 7	8 15	16 23	24 31
0	Zero			
1	Type (RR. TYPE). See Note 1.	Size (RR.SIZE)	Shadow flags (RR.SHAD)	
2	Start address (RR.SADD)			
3	End address (RR.EADD)			

Notes:

1. Bits in RR.TYPE are assigned as follows:

Value	Meaning if Set
1	Assign by pathname (RR.PATH)
2	Assign to temporary file (RR.TEMP)
3	Assign to device (RR.DEVC)
4	Assign to secondary LFC (RR.LFC2)
5	Assign to segment definition (RR.SPACE)
6	Assign by resource ID (RR.RID)
7	Reserved for future use
8	Reserved for future use
9	Mount by device mnemonic (RR.MTDEV)
10	Assign to ANSI labeled tape (RR.ANS)
11	Assign to shadow memory (RR.SHRQ)
12 - 255	Reserved

2. Bits in RR.ACCS are assigned as follows:

Bits	Meaning if Set
0	Read access allowed (RR.READ)
1	Write access allowed (RR.WRITE)
2	Modify access allowed (RR.MODFY)
3	Update access allowed (RR.UPDAT)
4	Append access allowed (RR.APPND)
5-15	Reserved
16	Explicit shared use requested (RR.SHAR)
17	Exclusive use requested (RR.EXCL)
18	Assign as volume mount device (RR.MNT)
19-31	Reserved

3. Bits in RR.OPTS are assigned as follows:

Bits	Meaning if Set
0	Treat as SYC file (RR.SYC) (TSM/JOB only)
1	Treat as SGO file (RR.SGO) (TSM/JOB only)
2	Treat as SLO file (RR.SLO)
3	Treat as SBO file (RR.SBO)
4	Explicit blocked option (RR.BLK)
5	Explicit unblocked option (RR.UNBLK)
6	Inhibit mount message (RR.NOMSG)
7	Reserved for system use
8	Automatic open requested (RR.OPEN)
9	User buffer address in FCB (RR.BUFF)
10-11	Reserved for system use
12	Mount with no-wait (RR.NOWT)
13	Mount as public volume (RR.PUBLC)
14	Set by H.VOMM for special case handling of VOMM assignments (RR.VOMM)
15	File is spooled when deallocated (RR.SEP)
16	ANSI labeled tape on RRS type 3 (RR.ANSI)
17-31	Reserved

4. RR.DENS contains the density specification for XIO high speed tape units. When specified, this field has the following bit significance:

Bit	Meaning if Set
0	Indicates 800 bpi nonreturn to zero inverted (NRZI)
1	Indicates 1600 bpi phase encoded (PE)
6	Indicates 6250 bpi group coded recording (GCR)

If this field is zero, 6250 BPI is set by default.

5. RR.DT3 specifies whether or not a channel is present and specifies the device type:

Bits	Meaning if Set
0	Channel present
1-7	Device type

6. RR.CHN3 specifies whether or not a subchannel is present and specifies the channel number:

Bits	Meaning if Set
0	Subchannel is present. Examined only if bit zero of RR.DT3 is set.
1-7	Channel number

7. RR.DT9 specifies whether or not a channel is present and specifies the device type:

Bits	Meaning if Set
0	Channel present
1-7	Device type

8. RR.CHN9 specifies whether or not a subchannel is present and specifies the channel number:

Bits	Meaning if Set
0	Subchannel is present. Examined only if RR.DT9 is set.
1-7	Channel number

2.32 Shared Memory Table (SMT)

Each entry in the Shared Memory Table (SMT) defines a shared memory area, such as CSECT, Global Common or Datapool. The number of entries in the SMT is established by the SYSGEN SHARE directive.

C.SMTA contains the address of the SMT; C.SMTN contains the number of entries in the SMT. Each entry is doubleword bounded.

Word	0 7 8 15 16 23 24 31
0	Resource identifier (SMT.RID)
7	
8	Partition name (SMT.NAME)
9	
10	Owner name or task number associated with this partition inclusion (SMT.TNUM)
11	
12	Owner name of partition creator (SMT.OWNR)
13	
14	Project group of partition creator (SMT.OWNR)
15	
16	Swap file resource ID (SMT.SRID)
17	
18	Compatibility version level (SMT.COMP)
19	Pathname identifier (SMT.PNID)
20	SMT index (SMT.IND) | Address of associated allocated resource table entry (SMT.ARTA)
21	Partition flags (SMT.FLAG). See Note 1.
22	Starting 512-word page number (SMT.PAGE) | Total number of pages (SMT.PTOT)
23	Starting map block (SMT.MAPS) See Note 2. | Number of map blocks (SMT.MAPN). See Note 2.
24	Start of DSECT (SMT.DSES) | Number of blocks in DSECT (SMT.DSEN)
25	Start of CSECT (SMT.CSES) | Number of blocks in CSECT (SMT.CSEN)
26	Memory type (SMT.MTY) | Number of tasks not outswapped (SMT.UCNT) | Number of words of memory pool used for the SMT's MIDL (SMT.POOL)
27	File offset of the write back read/write section within the shared image disc file (SMT.WBKS) | Reserved
28	Address of the map image descriptor list (SMT.MIDL)
29	Size in bytes of the read/write section to be written back (SMT.WBKN)
30	Number of swappable E-class memory map blocks (SMT.CME) | Number of swappable H-class memory map blocks (SMT.CMH)
31	Number of swappable S-class memory map blocks (SMT.CMS) | Number of outswapped map blocks (SMT.OTSW)

	0 7 8 15 16 23 24 31	
32 33	Time when shared CSECT was cataloged or time when executable image was linked (SMT.TIME)	
34 35	Date when shared CSECT was cataloged or date when executable image was linked (SMT.DATE)	
36	Spare bytes (SMT.SPBT)	Swapper trial use count (SMT.TUC)
37	Reserved	
38 39	Reserved	

Notes:

1. Bits in SMT.FLAG are assigned as follows:

Bit	Meaning if Set
0	Entry defines CSECT partition (SMT.CSCT)
1	Entry defines static common (SMT.STCM)
2	Entry defines dynamic common (SMT.DYCM)
3	Entry defines shared image (SMT.SHIM)
4	Partition is swappable (SMT.SWBL)
5	Partition is currently outswapped (SMT.OUTS)
6	SMT is unstable (SMT.BLDG)
7	Multicopy shared image (SMT.MULT)
8	Previously outswapped (CSECT only) (SMT.PRSW)
9	No automatic dequeue (SMT.LOCK)
10	Owner has read access (SMT.OW.R)
11	Owner has write access (SMT.OW.W)
12	Project group has read access (SMT.PR.R)
13	Project group has write access (SMT.PR.W)
14	Others have read access (SMT.OT.R)
15	Others have write access (SMT.OT.W)
16	Write back on last deallocation required (SMT.WRBK)
17	SMT active (SMT.ACTV)
18	Partition established by OPCOM (SMT.OPCM). See Note 3.
19	Partition requires shadow memory (SMT.SHAD)
20	Shared image is sharing by owner group (SMT.SHBO)
21	Entry defines memory disc (SMT.MD)
22	Trial swap (SMT.TS)
23	CSECT is loaded (SMT.CSLD)

2. For a single-copy shared image partition, SMT.MAPN reflects the size of both the read only and read/write sections and SMT.MAPS reflects the start of the read-only section.

   For a multicopy shared image partition, SMT.MAPN reflects the size of the read-only section and SMT.MAPS reflects the start of the read only section. Data for the read/write section is obtained from disc.

3. An OPCOM-established dynamic partition or shared image remains in memory when its assign count goes to zero. A user task can establish a resident dynamic partition or shared image by including it, setting SMT.OPCM, and then excluding the partition or shared image.

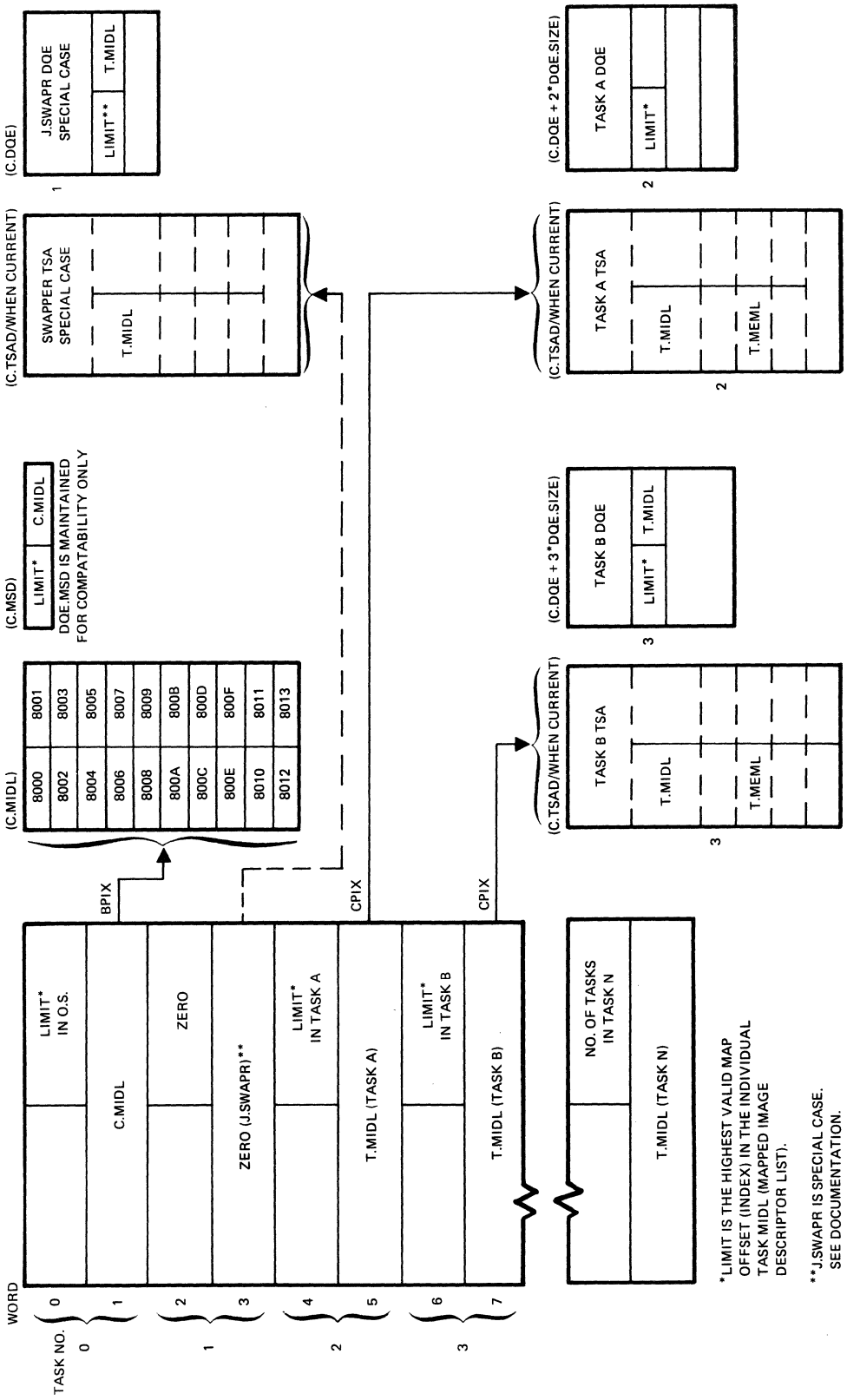
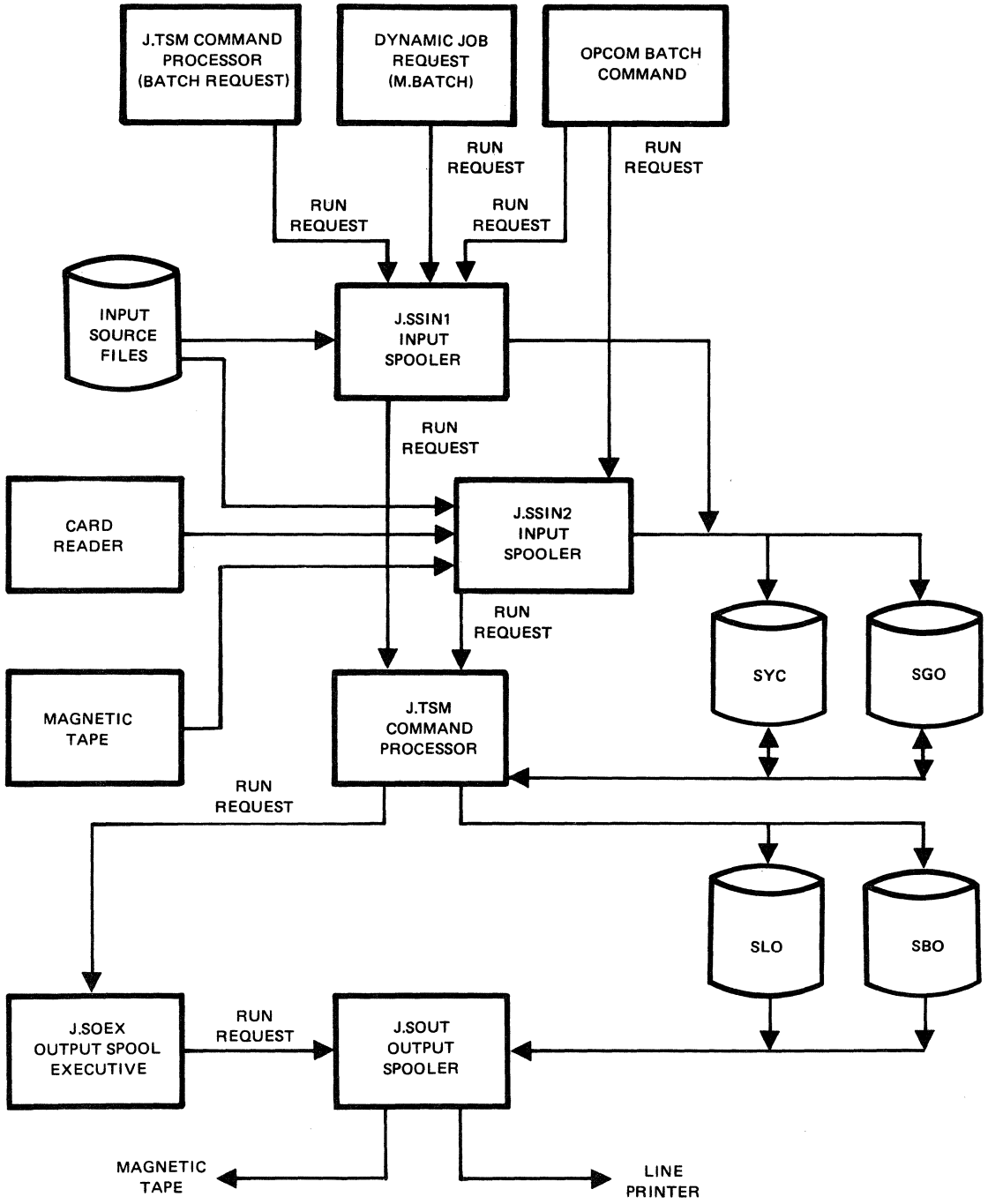


Figure 2-4. MPX-32 Map Structure for CONCEPT/32

2.33 Spooled File Data Structures

The organization of input and output for spooled files is shown in the following illustration. The remainder of this section describes the input and output spooling processes separately.

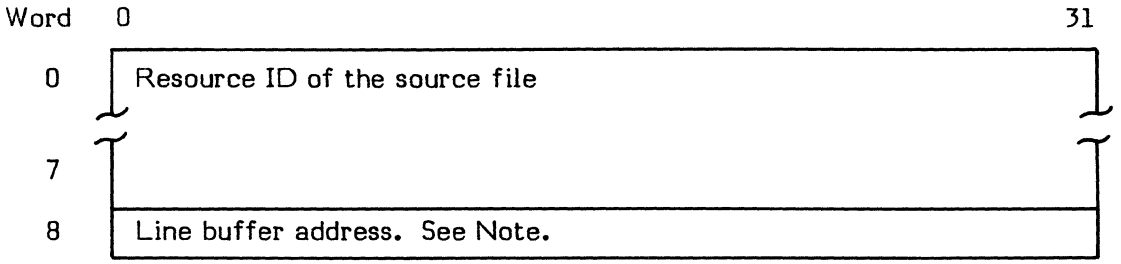


850090

Figure 2-5. Spooled File Data Structures

2.33.1 J.SSIN Run Request

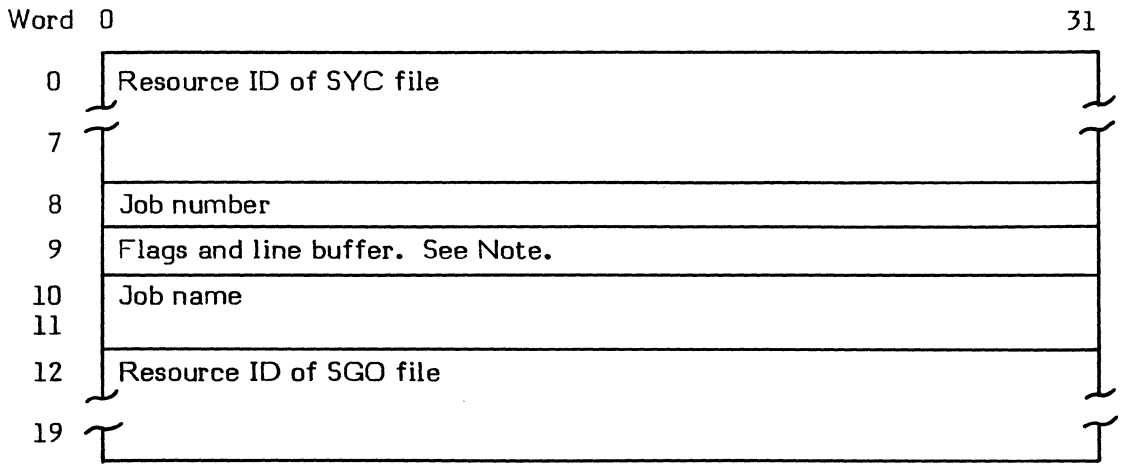
Input spooling is accomplished by run requesting a spool request to J.SSIN1 or J.SSIN2. Primary input source files must be blocked with either compressed or uncompressed data. The format of the J.SSIN run request is as follows:



Note: Line buffer address applies only to jobs batched by the command processor, J.TSM.

2.33.2 J.TSM Run Request

Each J.SSIN input spool request results in the creation of an SYC and SGO file and the initiation of a run request to J.TSM for batch processing. The format of the J.TSM run request is as follows:

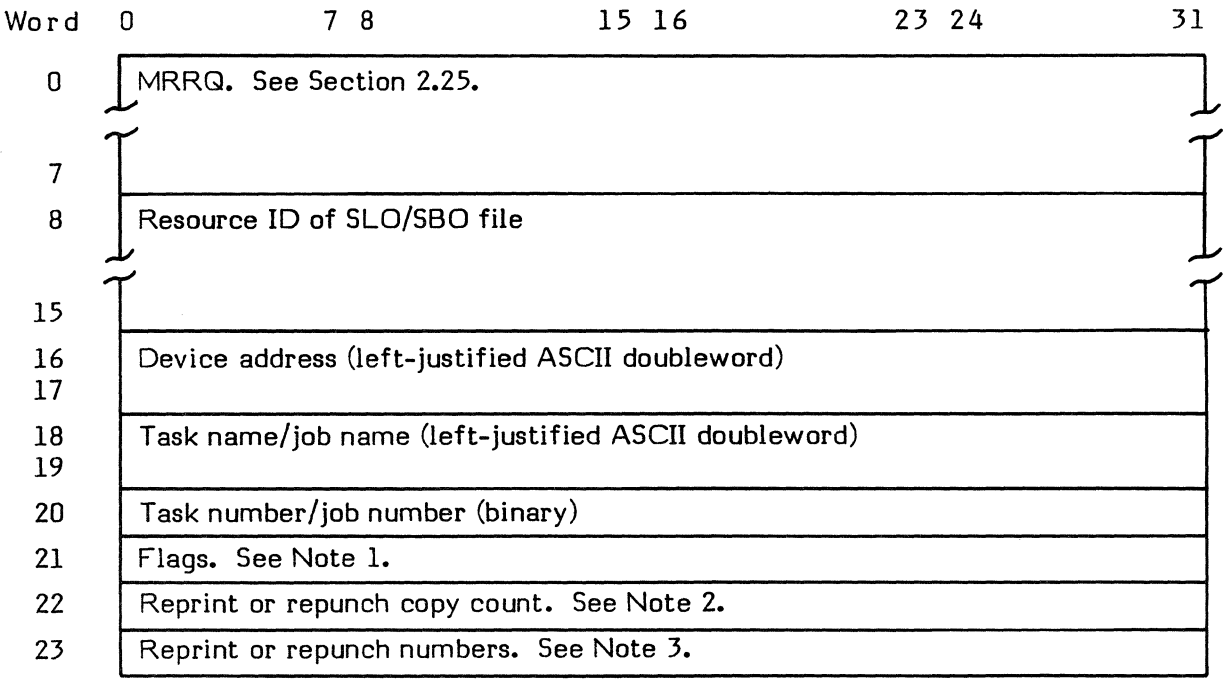


Note: Bits are assigned as follows:

Bits	Meaning if Set
0	Sequential job
1	$DEFM specified in job stream
2	Batch job
3-31	Reserved

2.33.3 J.SOEX Run Request

J.SOEX is the first of two phases of output spooling. Spooled files which are to be output to a device for processing are queued to J.SOEX by a run request to determine device availability. The format of the J.SOEX run request is as follows:



Notes:

1. Flag bits are assigned as follows:

Bits	Meaning if Set
0	Do not delete after deassignment
1	Copy/reprint request
2	Output in unformatted mode
3-23	Reserved
24-31	Value is as follows:

Value	Meaning
0	Print output
1	Punch output
2	Plot output
3-255	Reserved

2. This word contains the number of copies to be printed or punched in addition to the original.

3. The first halfword contains the beginning page number to be reprinted or repunched and the second halfword contains the ending page number to be reprinted or repunched.

Comments:

The actual J.SOEX run request is words 8 to 21. The run request is always appended to an MRRQ. See Message or Run Request Queue in Chapter 2.

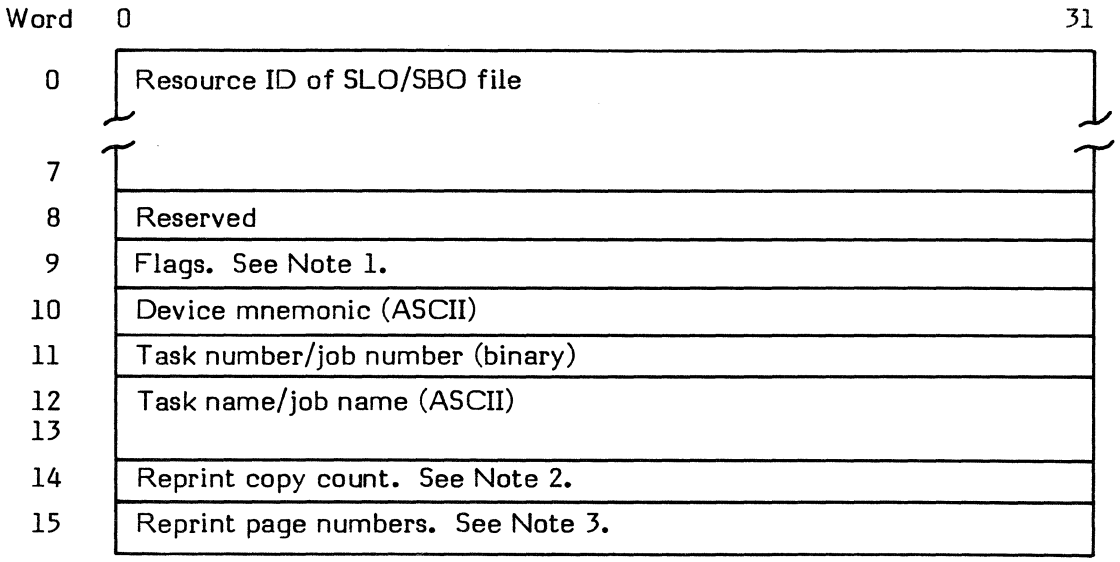
Temporary job-related SLO/SBO files are converted to permanent, non-SLO/SBO files prior to the run request of J.SOEX. Temporary job-related SLO/SBO files are output at end-of-job processing.

Permanent job-related SLO/SBO files are output by the M.DASN service as each is deassigned.

Nonjob-related SLO/SBO files are output by the M.DASN service. Temporary real-time SLO/SBO files are deleted after being output; permanent files are not. Real-time users may elect to use spooling directly by formatting and executing a run request to J.SOEX. The advantage is the specification of an output device.

2.33.4 J.SOUT Run Request

J.SOUT is the second phase of output spooling. When J.SOEX determines that a device is available for output processing, J.SOEX uses M.PTSK to activate J.SOUT. As part of the activation, a parameter buffer is passed to J.SOUT. The format of the J.SOUT parameter buffer is as follows:



Notes:

1. Bits are assigned as follows:

Bits	Meaning if Set
0-1	Reserved
2	Beginning a new job
3	Job is batch origin
4	SLO output request
5	Output device specified
6	Permanent file - do not delete
7	Reserved
8	Inhibit banner page
9	Display output - not formatted
10-31	Reserved

2. This word contains the number of copies to be printed in addition to the original.

3. The first halfword contains the beginning page number to be reprinted and the second halfword contains the ending page number to be reprinted.

2.34 System Master Directory (SMD)

The System Master Directory (SMD) is not supported as of MPX-32 release 2.0, but its format is retained for compatibility purposes. H.FISE compatible services and the file manager utility are the only users of this structure.

Disc File SMD Entry

Word	0 7	8 15	16 31
0 1	File name (SMD.AFN)		
2	File type (SMD.FTYP). See Note 1.	Start disc address (starting block number) (SMD.SBN)	
3	File indicators (SMD.FIN). See Note 2.	Length in 192-word blocks (SMD.BIF)	
4 5	User name (SMD.AUN)		
6	Compressed password (SMD.PWD)		UDT index (SMD.UDTX)
7	Reserved (SMD.NU)		

Memory Partition SMD Entry

Word	0 7	8 15	16 31
0 1	File Name (SMD.AFN)		
2	Starting logical page or number (SMD.SLP)		Starting physical page number or zero (SMD.SPP)
3	File indicators (SMD.FIN). See Note 2.	Memory class (SMD.MC)	Length in pages (SMD.LIP)
4 5	Reserved		
6	Compressed password (SMD.PWD)		Reserved
7	Reserved (SMD.NU)		

Notes:

1. SMD.FTYP specifies a two-character hexadecimal file type that is output by the Volume Manager in ASCII. Default is 00.

   00-9F are available for customer usage. A0-FF are reserved for Gould CSD development's usage. Known file type codes are:

Value	Description
42	Toolkit blocked mode
43	Toolkit card mode
46	Toolkit formatted mode
50	Toolkit packed mode
55	Toolkit unblocked mode
B0	Base mode object file
BA	Base mode shared image (or BASIC Save file)
BB	Base mode object library file
BC	Base mode macro library file
BE	Base mode load module file
C0	Spooled output file
CA	Cataloged load module
D0	Memory disc save task (J.MDSAVE) file
DB	Symbolic debugger command file
ED	Saved text editor file
EE	Stored text editor file
FE	Text editor work file
FF	SYSGEN generated file

2. Bits in SMD.FIN are assigned as follows:

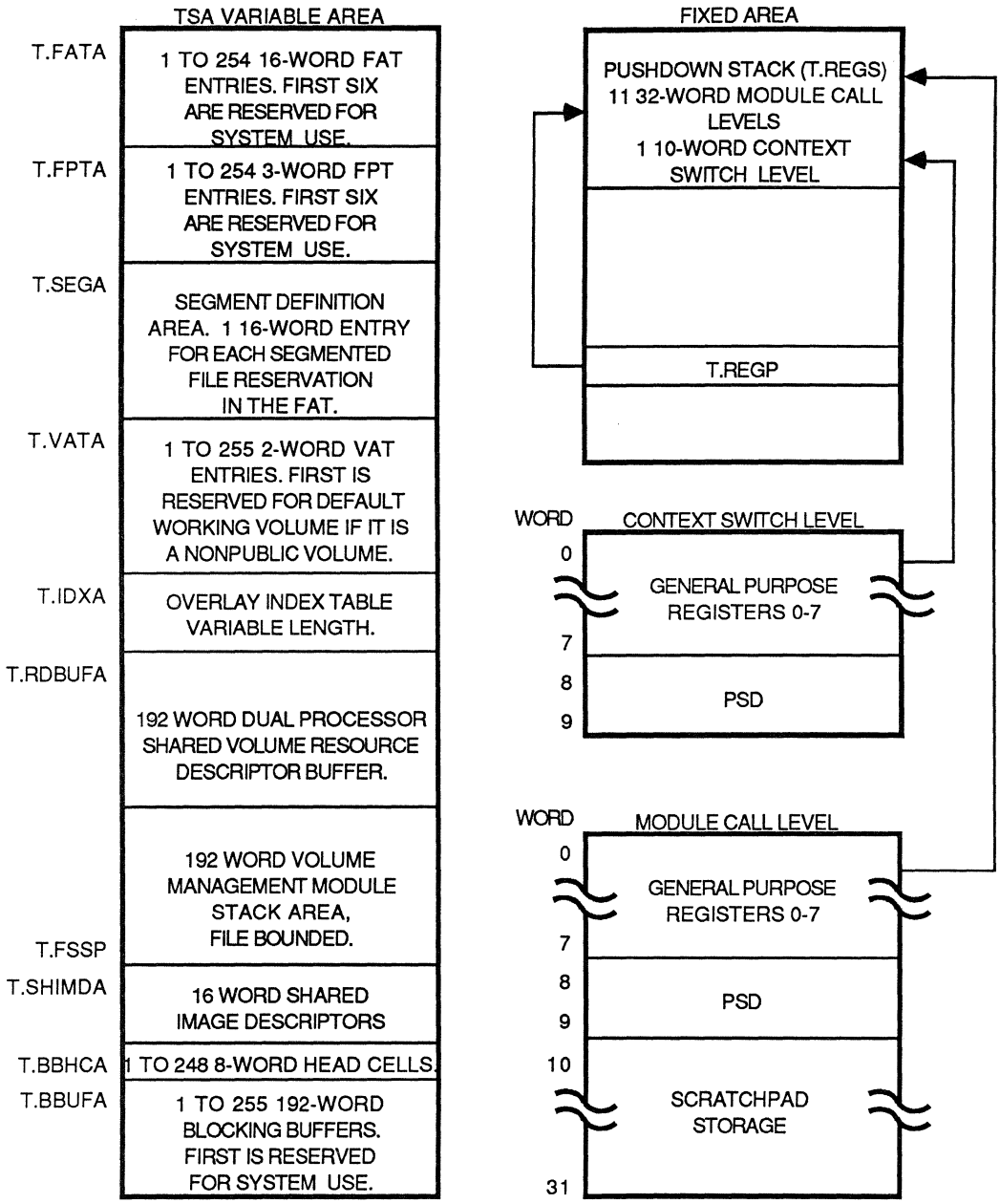
Bit	Meaning if Set
0	Permanent file is active
1	SYSGEN memory partition
2	No-save option is in effect
3	Fast file
4	Collision mapping
5	Non-SYSGEN memory partition
6	Password is required to write
7	Password is required to read/write

2.35 Task Service Area (TSA)

The Task Service Area (TSA) is a section of memory associated with each active task which is used by MPX-32 for storage of task-unique information. A TSA is allocated for each task when the task becomes active and is deallocated when the task terminates. The size of each task's TSA is fixed for the duration of the task's execution. However, the sizes of TSAs 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 six buffers' FAT and FPT entries are reserved for MPX-32 use and are present in every TSA.

The pushdown area in the TSA provides re-entrancy 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. When context switch away from a task occurs, 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.



860602

Figure 2-6. TSA Structure

TSA Structure

Word # (Decimal)	Byte (Hex)	0 7	8 15	16 23	24 31
0-361	0	T.REGS			
362-363	5A8	T.PRJCT			
364	5B0	T.PGOW			
365	5B4	T.PARENT			
366	5B8	T.REGP			
367	5BC	T.FSSP			
368-385	5C0	T.CONTXT			
386-389	608	T.FREE			
390-393	618	T.USED			
394	628	T.FIRST			
395	62C	T.LAST			
396	630	T.ITAC			
397	634	T.BASEP			
398-413	638	T.BFCB			
414-421	678	T.PROT			
422-429	698	T.BREGS			
430-437	6B8	T.MPP			
438	6D8	T.DBHAT			
439	6DC	T.PRNO			
440	6E0	T.ABRTA			
441	6E4	T.BBUFA			
442	6E8	T.VATA			
443	6EC	T.VATN	T.WORK	T.SEGN	T.BIT3
444	6F0	T.FATA			
445	6F4	T.FPTA			
446	6F8	T.SEGA			
447	6FC	T.BREAK			
448	700	T.MSGR			
449	704	T.BIAS			
450	708	T.TEND			
451	70C	T.END			
452	710	T.TRAD			
453	714	T.LINNO		T.UKEY	
454	718	T.BIT1	T.BIT2	T.BBUFN	T.FILES
455	71C	T.DSOR		T.DSSZ	
456	720	T.CSOR		T.CSSZ	
457	724	T.MEML1		T.MEML2	
458	728	T.MIDL1		T.MIDL2	
459	72C	T.EAOR		T.EASZ	
460-461	730	T.ACCESS			
462-463	738	T.SYCS/T.LINBUF			
464-471	740	T.SGOS			
472-479	760	T.SLOS			
480-487	780	T.SBOS			
488-491	7A0	T.CDIR			
492-499	7B0	T.CVOL			

Word # (Decimal)	Byte (Hex)	0 7 8 15 16 23 24 31
500	7D0	T.IPUAC
501	7D4	T.CURH
502	7D8	T.CRHX
503	7DC	T.SYCF
504	7E0	T.SGOF
505	7E4	T.SLOF
506	7E8	T.SBOF
507	7EC	T.CPUSH | T.SPARB | T.DBOR
508	7F0	T.DPINFO | T.TASKK | T.BIT4
509	7F4	T.RDBUFA
510	7F8	T.EXCPAD
511	7FC	T.RORG
512	800	T.RWORG
513	804	T.DBSTAT
514	808	T.MIDLA
515	80C	T.MEMLA
516	810	T.MEMLO
517	814	T.STKSZ
518-521	818	T.DBNAME
522-523	828	T.EXPSD
524	830	T.IDXA
525	834	T.WORK | T.NSI
526	838	T.SHIMDA
527	83C	T.PREL
528	840	T.DBSTW2
529	844	T.BBHCA
530	848	T.BIT5
531	84C	T.SHTBL
532	850	T.SMTMLT
533	854	T.SIGSTK
534-541	858	T.MPXBR
542	878	T.OSREGS
543	87C	T.NSTAT | T.SPARHW
544-574	880	T.SPARES
575	8FC	T.ABPSD
576	900	T.MIDL

Byte (Hex)	Symbol	Description
0	T.REGS	Task pushdown stack (362 words)
5A8	T.PRJCT	Current project group name for file allocation (2 words)
5B0	T.PGOW	Task option word
5B4	T.PARENT	Task sequence number of parent task (activator)
5B8	T.REGP	Current level of pushdown in stack area
5BC	T.FSSP	Address of the current pushdown level
5C0	T.CONTXT	Debug context area (18 words, must be on an 8 word boundary)
608	T.FREE	Free list head cell (4 words)
618	T.USED	Allocated list head cell (4 words)
628	T.FIRST	First logical memory address mapped into user task
62C	T.LAST	Last logical memory address mapped into user task
630	T.ITAC	Interval timer based accounting (CPU)
634	T.BASEP	Address of file system buffer
638	T.BFCB	System service file control block (16 words)
678	T.PROT	Reserved (8 Words)
698	T.BREGS	Task base register context area (8 words)
6B8	T.MPP	Memory pool pointers (8 words)
6D8	T.DBHAT	Address of debug halfword address table
6DC	T.PRNO	Address of task's dispatch queue entry
6E0	T.ABRTA	Address of task's abort receiver
6E4	T.BBUFA	Address of task's blocking buffer
6E8	T.VATA	Address of task's volume assignment table
6EC	T.VATN	Number of entries in volume assignment table (1 byte)
	T.WORK	Work space scratch area (1 byte)
	T.SEGN	Number of dynamic segment definition areas (1 byte)

	T.BIT3	TSA bit field assigned as follows:

Bits	Meaning if Set
0	P-task RRS error encountered (T.RRERR)
1	Static assignment in progress (T.SASSN)
2	M.GE or M.GD service in use (T.GEGD)
3	M.MEMB service in use (T.MEMB)
4	Base mode save (T.BRSAVE)
5	Modify descriptor in progress (T.MOD)
6	Retry suspended (multiport only) (T.SUSP)
7	ACX-32 privileged task (T.ACXP)

6F0	T.FATA	Address of task's file assignment table
6F4	T.FPTA	Address of task's file pointer table
6F8	T.SEGA	Address of segment definition area
6FC	T.BREAK	Address of task's break receiver
700	T.MSGR	Address of task's message receiver
704	T.BIAS	Starting address of task's DSECT area
708	T.TEND	Ending address of TSA when task is loaded
70C	T.END	Last location loaded by the task loader within the DSECT plus one word
710	T.TRAD	Transfer address of task's main segment
714	T.LINNO	Number of lines on a TSM screen (one halfword)
	T.UKEY	Compressed original user key (one halfword)
718	T.BIT1	Bit variables (one byte) assigned as follows:

Bits	Meaning if Set
0	First 4K of E-class I/O buffer in use (T.EIO1)
1	Second 4K of E-class I/O buffer in use (T.EIO2)
2	Arithmetic exception trap (T.EXCP)
3	Reserved (T.EBUF)
4	Suspend after activation (T.WAIT)
5	Debugger required (T.DBG)
6	CSECT to share (T.SHR)
7	Command file is active in TSM (T.COMFIL)

	T.BIT2	Bit variables (one byte) assigned as follows:

Bits	Meaning if Set
0	E-class wait buffer (T.EBUF1)
1	E-class no-wait buffer (T.EBUF2)
2	User area of descriptor is being modified (T.MODU)
3	Base mode debugger (T.BASE)
4	System Administrator attribute (T.SAM)
5	All SLO output is directed to the terminal (T.UTSLO)
6	H.VOMM file control block reinitialization is required (T.FCBINT)
7	Task cannot attach debugger (T.NODBG)

	T.BBUFN	Number of blocking buffers associated with task (1 byte)
	T.FILES	Number of FAT/FPT pairs associated with task (1 byte)
71C	T.DSOR	DSECT origin within T.MEML/T.MIDL (1 halfword); usually zero
	T.DSSZ	DSECT size in map blocks (1 halfword)
720	T.CSOR	CSECT origin within T.MEML/T.MIDL (1 halfword). If size is zero, T.CSOR is set equal to T.EAOR contents.
	T.CSSZ	CSECT size in map blocks (1 halfword)
724	T.MEML1	MEML overlay map one (1 halfword)
	T.MEML2	MEML overlay map two (1 halfword)
728	T.MIDL1	MIDL overlay map one (1 halfword)
	T.MIDL2	MIDL overlay map two (1 halfword)
72C	T.EAOR	Extended address origin in T.MEML/T.MIDL (1 halfword)
	T.EASZ	Extended address size in map blocks (1 halfword)
730	T.ACCESS	Privileged flags (2 words). Bit variables are assigned as follows:

Bits	Meaning if Set
0	EXIT directive disabled (T.EXIT)
1	ABORT directive disabled (T.ABORT)
2	ACTIVATE directive disabled (T.ACTVT)
3	BATCH directive disabled (T.BATCH)
4	BREAK directive disabled (T.BRK)
5	CONNECT directive disabled (T.CONNCT)
6	CONTINUE directive disabled (T.CONTNU)
7	DEPRINT directive disabled (T.DEPRNT)
8	DEPUNCH directive disabled (T.DEPNCH)
9	DISABLE directive disabled (T.DISABL)
10	DISCONNECT directive disabled (T.DISCON)
11	DUMP directive disabled (T.DUMP)

Bits	Meaning if Set
12	ENABLE directive disabled (T.ENABLE)
13	Reserved
14	HOLD directive disabled (T.HOLD)
15	KILL directive disabled (T.KILL)
16	LIST directive disabled (T.LIST)
17	MODE directive disabled (T.MODE)
18	MODIFY directive disabled (T.MODIFY)
19	OFFLINE directive disabled (T.OFFLNE)
20	ONLINE directive disabled (T.ONLINE)
21	PURGEAC directive disabled (T.PURGAC)
22	REDIRECT directive disabled (T.REDIR)
23	REMOVE directive disabled (T.REMOVE)
24	TSM REPRINT directive disabled (T.REPRNT)
25	REPUNCH directive disabled (T.REPNCH)
26	REQUEST directive disabled (T.REQEST)
27	DELETETIMER directive disabled (T.DTIMER)
28	Reserved
29	SEARCH directive disabled (T.SEARCH)
30	SEND directive disabled (T.SEND)
31	SETTIMER directive disabled (T.STIMER)
32	SNAP directive disabled (T.SNAP)
33	START directive disabled (T.START)
34	STATUS directive disabled (T.STATUS)
35	SYSASSIGN directive disabled (T.SYSASN)
36	TIME directive disabled (T.TIME)
37	TSM URGENT directive disabled (T.URGENT)
38	RESUME directive disabled (T.RESUME)
39	ESTABLISH directive disabled (T.ESTAB)
40	Access to other owners disabled (privileged) (T.OWNACC)
41	Activation of privileged tasks disabled (privileged) (T.APRIV)
42	Cataloging as privileged user disabled (privileged) (T.CPRIV)
43	TSM RESTART (privileged) disabled (T.RESTRT)
44	TSM URGENT directive disabled (privileged) (T.PRIOR)
45	MOUNT directive disabled (T.MOUNT)
46	DISMOUNT directive disabled (T.DMOUNT)
47	System Administrator attribute (privileged) enabled (T.SAA)
48	Commands are not echoed (privileged) (T.NOCOMM)
49	Cataloging tasks which set owner ID on access enabled (privileged) (T.OWNRID)
50	Changing defaults enabled (T.CHANGD)
51-55	Reserved for future use
56-63	Available for customer use

738	T.SYCS	SYC definition (2 words)
	T.LINBUF	Address of TSM's line buffer (1 word)

740	T.SGOS	SGO definition (8 words)
760	T.SLOS	SLO definition (8 words)
780	T.SBOS	SBO definition (8 words)
7A0	T.CDIR	Name of current working directory (4 words)
7B0	T.CVOL	Name of current working volume (8 words)
7D0	T.IPUAC	IPU real-time clock accounting
7D4	T.CURH	Current high address in map
7D8	T.CRHX	Current high address in extended space
7DC	T.SYCF	Address of SYC dedicated FAT
7E0	T.SGOF	Address of SGO dedicated FAT
7E4	T.SLOF	Address of SLO dedicated FAT
7E8	T.SBOF	Address of SBO dedicated FAT
7EC	T.CPUSH	Number of push levels used for compatibility (1 byte)
	T.SPARB	Reserved (1 byte)
	T.DBOR	Task debugger origin within T.MIDL/T.MEML (1 halfword). If debugger is not included with the task, T.DBOR is set equal to T.CSOR contents.
7F0	T.DPINFO	Dual-processor information (1 halfword). Bit variables 0 to 15 indicate the base priority.
	T.TASKK	Number of maps in the TSA and CALL stack (1 byte)
	T.BIT4	Bit variables (1 byte) assigned as follows:

Bit	Meaning if Set
0	Inhibit exception handler (T.NOSET)
1	Inhibit command line scan (T.INHSCN)
2	Debugger load in progress (T.DBLIP)
3	Arithmetic exception handling in progress (T.ARTECP)
4	Base mode task with shared CSECT (T.BRSCS)
5	No checksum on load (T.NCKSM)
6	Task is swappable (T.SWP)
7	Shadow memory table is present (T.SHAD)

7F4	T.RDBUFA	Address of dual-processor resource descriptor buffer

7F8	T.EXCPAD	Arithmetic exception handler address
7FC	T.RORG	Read-only section origin
800	T.RWORG	Read/write section origin
804	T.DBSTAT	Debugger status word
808	T.MIDLA	MIDL array address
80C	T.MEMLA	MEML array address
810	T.MEMLO	MIDL to MEML offset
814	T.STKSZ	Number of bytes in CALL stack
818	T.DBNAME	Base mode debugger name (4 words)
828	T.EXPSD	PSD at arithmetic exception (2 words)
830	T.IDXA	Overlay count and index table address. Bytes are assigned as follows:

Bytes	Definition
0	Number of entries in the overlay index table (each entry in the table is ten bytes in length)
1-3	Address of the overlay index table

If this word is zero, there are no overlays in single file format. If this word is minus one, there are overlays in separate file format.

834	T.WORK	Work space scratch area (1 halfword)
	T.NSI	Number of shared image descriptors (1 halfword)
838	T.SHIMDA	Shared image descriptor table address
83C	T.PREL	Prelocation delta
840	T.DBSTW2	Debugger status word 2
844	T.BBHCA	Address of task's blocking buffer head cell
848	T.BIT5	Bits defined as follows:

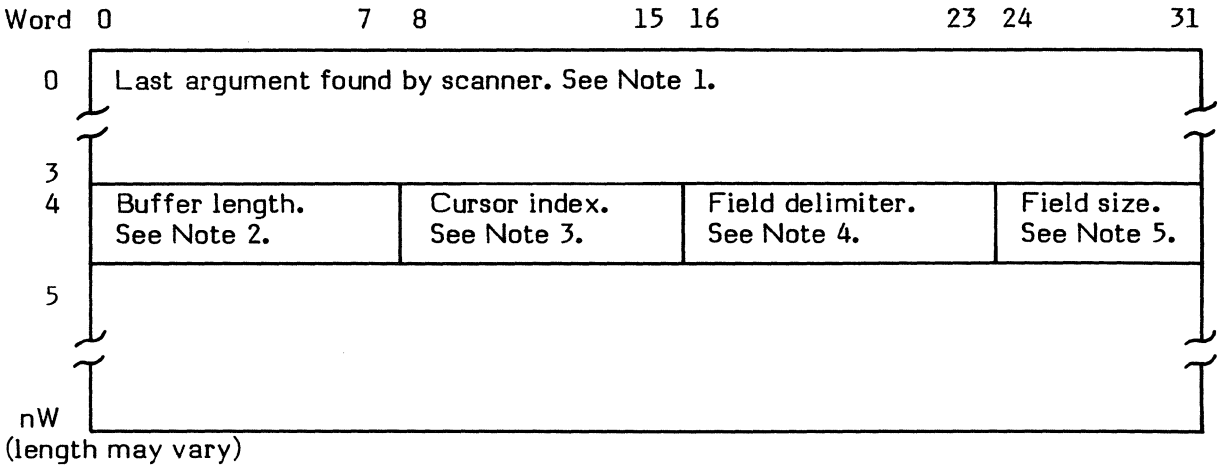
Bits	Meaning if Set
0	Task activation complete (T.TAC)
1	Reserved
2	Reserved
3	No arithmetic exception handler (T.NOEXCP)
4	Arithmetic exception occurred in CPU (T.CPUAE)
5-31	Reserved

84C	T.SHTBL	Shadow memory table address
850	T.SMTMLT	SMT index for multicopied shared image with no read only section
854	T.SIGSTK	ADA stack size
858	T.MPXBR	Base register save area (8 words)
878	T.OSREGS	Base code logical map address
87C	T.NSTAT	Number of static partition maps in use (1 halfword)
87E	T.SPARHW	Reserved (1 halfword)
880-8F8	T.SPARES	Reserved
8FC	T.ABPSD	Abort PSD
900	T.MIDL	Halfword map image descriptor list (128 to 1024 words depending on the logical space requirement of the task)

Bits	Meaning if Set
0	Map number is valid (MIDL.VAL)
1	First protection granule is write protected (MIDL.PRO)
2	Second protection granule is write protected (MIDL.PR2)
3	Third protection granule is write protected (MIDL.PR3)
4	Fourth protection granule is write protected (MIDL.PR4)
5-15	Physical map number

2.36 Terminal Line Buffer

The terminal line buffer buffers terminal input and output. It is allocated from memory pool for each on-line task when the terminal is opened. The size of the buffer is determined by the contents of UDT.CHAR. The buffer is pointed to by T.LINBUF and always begins on a doubleword boundary. It is deallocated when the terminal is closed.



Notes:

1. Two doublewords containing last argument found by syntax scanner. It is left justified and blank filled.

2. Number of words in line buffer (20 minimum, 59 maximum). This number does not reflect possible words required to maintain memory pool doubleword bounding.

3. Cursor index for next call to scanner. Relative to word zero of line buffer.

4. The delimiting character previously found by scanner.

5. Number of significant characters in previous argument found by scanner.

2.37 Timer Table

The timer table contains all necessary information for the time scheduling of the functions provided in the create timer entry service. The functions include activating a program, resuming a program, setting a bit, resetting a bit, and requesting an interrupt.

The table also contains a variable number of five word timer entries that are specified at SYSGEN. Entries in the table are identified by a two-character (ASCII) timer ID specified by the create timer entry service. Entries are deleted by the delete timer entry service. SYSGEN forces three entries for the exclusive use of J.SSIN1, J.SSIN2, and J.SOUT. The format of the timer table entries follows:

Word	0 3	4 7	8 15	16 31
0	Timer Status. See Note 1.	Function code. See Note 2.	Reserved	Timer ID - two unique ASCII characters
1	Function parameter one. See Note 3.			
2	Function parameter two. See Note 3.			
3	Current time value in negative time units. See Note 4.			
4	Reset time value in negative time units. See Note 5.			

Notes:

1. The time status bits have the following meanings:

Bits	Meaning if Set
0	Timer in hold state
1	Timer entry is taken
2	Reissue activation request
3	Reserved

2. The function code (4-bit numeric value) is assigned as follows:

Value	Meaning
1	Activate program
2	Resume program
3	Set bit in static memory partition or operating system
4	Reset bit in static memory partition or operating system
5	Request interrupt between X'12' to X'7F'

3. The function code, bits 4 through 7 in word zero, determines the contents of the function parameters (words one and two).

Function Code	Word	Contents
1	1	Dispatch queue address of task to be activated at time out. M.SETT preactivates the task, then acquires the dispatch queue address. The task remains suspended until time out.
	2	Reserved
2	1	Dispatch queue address of task to be resumed at time out. M.SETT acquires the dispatch queue address from the user-supplied task name or task number.
	2	Reserved
3	1	Address of word where the bit is to be set.
	2	The bit configuration to be ORed at time out with the data at the address specified in word one.
4	1	Address of word where the bit is to be reset.
	2	The bit configuration to be ANDed at time out with the data at the address specified in word one.
5	1	A request interrupt (RI) instruction, for the user-specified priority level, to be executed upon time out.
	2	Reserved

4. The current time value is the negative timer units to elapse before the selected function is completed. This value is incremented until it equals zero. At that time, the selected function is complete.

   If word four is zero when time out occurs, the entry is deleted. If word four is nonzero when time out occurs, the value in word four is loaded into word three and incremented.

5. The reset time value is the negative timer units to elapse before the selected function is repeated. When the timer expires, this value is loaded into word three and incremented.

   Word four is not changed until the timer is deleted or the sytem is rebooted.

2.38 Type Control Parameter Block (TCPB)

The Type Control Parameter Block (TCPB) allows I/O to and from the system console by setting up task buffer areas for messages output by a task and optional reads back from the console. If no input is desired, word one of the TCPB must be zero.

See the MPX-32 Reference Manual Volume I, Chapter 5 for further details on the TCPB.

Word	0	1 11	12 13	30 31
0	Output quantity (TCP.OQ)	See Note 1	Output data address (TCP.OTCW)	
1	Input quantity (TCP.IQ)	See Note 1	Input data address (TCP.ITCW)	
2	Console Teletype Flags (TCP.FLGS). See Note 2.			

Notes:

1. This bit is set to one.

2. Bits in TCP.FLGS are assigned as follows:

Bit	Meaning if Set
0	No-wait I/O
1	Output and input quantities are contained in bits 0-7. Output and input addresses are contained in bits 8-31 of words zero and one, respectively. (TCP.LAD). If 24-bit addressing is used, the output and input data addresses must be word bounded.
31	Operation in progress. This bit is reset after post-I/O processing completes.

2.39 Unit Definition Table (UDT)

The Unit Definition Table (UDT) is a system resident structure that identifies device-dependent information required by a handler for a specific device. The UDT is built by the SYSGEN process, one for each device configured in the system. During SYSGEN, each UDT is linked to its corresponding Controller Definition Table (CDT) and its associated controller and handler.

Word	0 7	8 15	16 23	24 31
0	UDT index (UDT.UDTI)		CDT index (UDT.CDTI)	
1	Unit status (UDT.STAT). See Note 1.	Device type code (UDT.DTC). See Note 2.	Logical channel number (UDT.CHAN)	Logical subaddress (UDT.SUBA)
2	Reserved	Address of dispatch queue entry of task which has device allocated if device is not shared (UDT.DQEA)		
3	Physical channel Physical sub-number (UDT.PCHN)	Sectors per address (UDT.PSUB)	Sectors per block (UDT.SPB) or number of characters per line (UDT.CHAR). See Note 3.	Sectors per allocation unit (UDT.SPAU) or number of lines per screen (UDT.LINE) See Note 4.
4	Flags (UDT.FLGS). See Note 5.	Number of sectors per track on disc or global line counter if a terminal (UDT.SPT)	Maximum byte transfer (UDT.MBX)	
5	Number of sectors on disc or tab setting if a terminal (UDT.SECS)			
6	Sector size, on disc or a tab setting if a terminal(UDT.SSIZ)		Number of heads on disc or a tab setting if a terminal (UDT.NHDS)	
7	Serial number if tape or removable disc (UDT.SERN). See Note 6.			
8	Peripheral time-out value (UDT.PTOV)			
9	Reserved	Address of device context area (UDT.DCAA) or handler name at initialization (UDT.HNAM)		
10	Bit flags (UDT.BIT2). See Note 7.		Associated Allocated Resource Table index if assigned (UDT.ARTI)	
11	Service interrupt handler address (UDT.SIHA)			
12	Reserved for future de-velopment use (UDT.PRIF). See Note 8.	Reserved for future de-velopment use (UDT.SECF).	Reserved for future de-velopment use (UDT.SHFL).	Reserved for future de-velopment use (UDT.DQEN).
13	Address of first IOQ linked to this device (UDT.FIOQ)			
14	Address of last IOQ linked to this device (UDT.BIOQ)			
15	Link Priority (UDT.LPR1)	Link Count (UDT.IOCT)	Unit Status byte 2 (UDT.STA2). See Note 9.	

Notes:

1. Bits in UDT.STAT are assigned as follows:

Bit	Meaning if Set
0	On-line (UDT.ONLI)
1	Dual-ported XIO disc (UDT.DPDC)
2	Allocated (UDT.ALOC)
3	In use (UDT.USE)
4	System output unable to allocate (UDT.NOAL)
5	Shared device (UDT.SHR)
6	Premounted (UDT.PREM)
7	Terminal (TSM) device (UDT.TSM)

2. For example, 01 for any disc, 04 for any tape, etc. Valid device type codes are listed in Appendix A of the MPX-32 reference manual.

3. For discs, contains the number of sectors per block (UDT.SPB). For terminals, contains the number of characters per line (UDT.CHAR).

4. For discs, contains the number of sectors per allocation unit (UDT.SPAU). For SLO or terminals, contains the number of lines per page or screen (UDT.LINE).

5. Bits in UDT.FLGS are assigned as follows:

Bit	Meaning if Set
0	Extended I/O device (UDT.FCLS)
1	I/O outstanding (UDT.IOUT)
2	Removable disc pack (UDT.RMDV)
3	Terminal user logged on (UDT.LOGO)
4	Autoselectable for batch SLO (UDT.BSLO)
5	Autoselectable for batch SBO (UDT.BSBO)
6	Autoselectable for real-time SLO (UDT.RSLO)
7	Autoselectable for real-time SBO (UDT.RSBO)

6. If the device is a terminal or console, the first halfword is the current terminal type for TERMDEF (UDT.CTDF) and the second halfword is the default terminal type (UDT.DTDF).

7. Bits in UDT.BIT2 are assigned as follows:

Bit	Meaning if Set
0	Port is private; else switched (UDT.DIAL)
1	Port is connected to modem (UDT.MODM)
2	Port has graphic capability (UDT.GRFC)
3	Port is full duplex (UDT.FDUX)
4	Port is configured multidrop (UDT.MDRA)
5	Volume mounted on device (UDT.VOL)
6	Echo by computer (UDT.ECHO)
7	Device has failed. Log off TSM (UDT.DEAD)
8	Cache device (UDT.CAC)
9	Inhibit automatic line wrap (UDT.NRAP)
10	Spool device requires form feed after printing rather than before; initial form feed is inhibited (UDT.FEOP)

11	Quarter inch cartridge tape drive (UDT.QITD)
12	Software read flow control required (UDT.RXON)
13	Software write flow control required (UDT.WXON)
14	Hardware read flow control required (UDT.RHWF)
15	Hardware write flow control required (UDT.WHWF)

8. For switched port, contains the value specified in the LOGONFLE CXR = option (UDT.CXR)

9. Bits in UDT.STA2 are assigned as follows:

Bit	Meaning if Set
0	IOQ linked from UDT (UDT.IOQ)
1	IOP device (initialized by SYSGEN) (UDT.IOP)
2	Device malfunction (UDT.MALF)
3	Operator intervention applicable (UDT.INTV)
4	Use standard XIO interface
5	Floppy disc
6	Cartridge module drive
7	Moving head disc with fixed head option
8	If software read flow control enabled, use DTR line; otherwise, use RTS line. (UDT.RDTR)
9	Memory disc (UDT.MD)
10	Memory allocated for memory disc (UDT.MDAL)
11	Start address of memory disc specified at SYSGEN (UDT.MDST)
12	Multiport device is shared with an MPX-32 Release 3.2C or earlier version (UDT.PPV)
13	Device is exclusive ANSI (UDT.ANSI)
14	Serial printer (UDT.SLPR)
15	Port is switched and CXR=N option has been specified (UDT.DCXR)

2.40 Volume Assignment Table (VAT)

The Volume Assignment Table (VAT) is used to identify a nonpublic volume associated with a particular task. The table is located in the task's service area (TSA). The VAT points to the information necessary to process access to the volume for the specific task and required to be memory resident (frequently required information).

A volume assignment table entry is created when a nonpublic volume mount is invoked. A mount must be issued by a task before being allowed to reference the volume or its contents. A linkage in the VAT is then connected to a corresponding entry in the Mounted Volume Table (MVT) for the requested volume. If an entry does not exist in the MVT for the referenced volume (i.e., the volume is not physically mounted), a mount message is issued and an entry is made in the MVT when the volume is physically mounted. When this is completed, the linkage in the VAT is connected to the MVT as previously described.

Word	0 7	8 15	16 23	24 31
0	Assign count (VA.ASSNS)	Mounted Volume Table address (VA.MVTA)		
1	Assigned access restrictions (VA.ACCS). See Note.			

Note: Bits in VA.ACCS are assigned as follows:

Bits	Meaning if Set
0-15	Reserved
16	Entry is available (VA.AVAIL)
17-31	Reserved

2.41 Disc Resident Resource Descriptors (RD)

The following chart shows the MPX-32 disc resident resource descriptors most commonly used and their correlation to each other.

As shown in the chart, the first 86 words of each descriptor are identical. For example, words 0 through 63 are the common parameters and words 64 through 85 are the space parameters.

Unique descriptor types begin at word 86 and end at word 95, with the exception of the memory partition parameters that end at word 97 because of memory page parameters, and the volume descriptor parameters that end at word 159 because they do not contain a segment definition area.

The descriptors, with the exception of the two noted above, contain a segment definition area that begins at word 96 and ends at word 159. The segment definition area for memory partitions begins at word 98 and ends at word 159.

The descriptors contain a free usage area that begins at word 160 and ends at word 190.

Each descriptor is described following the chart.

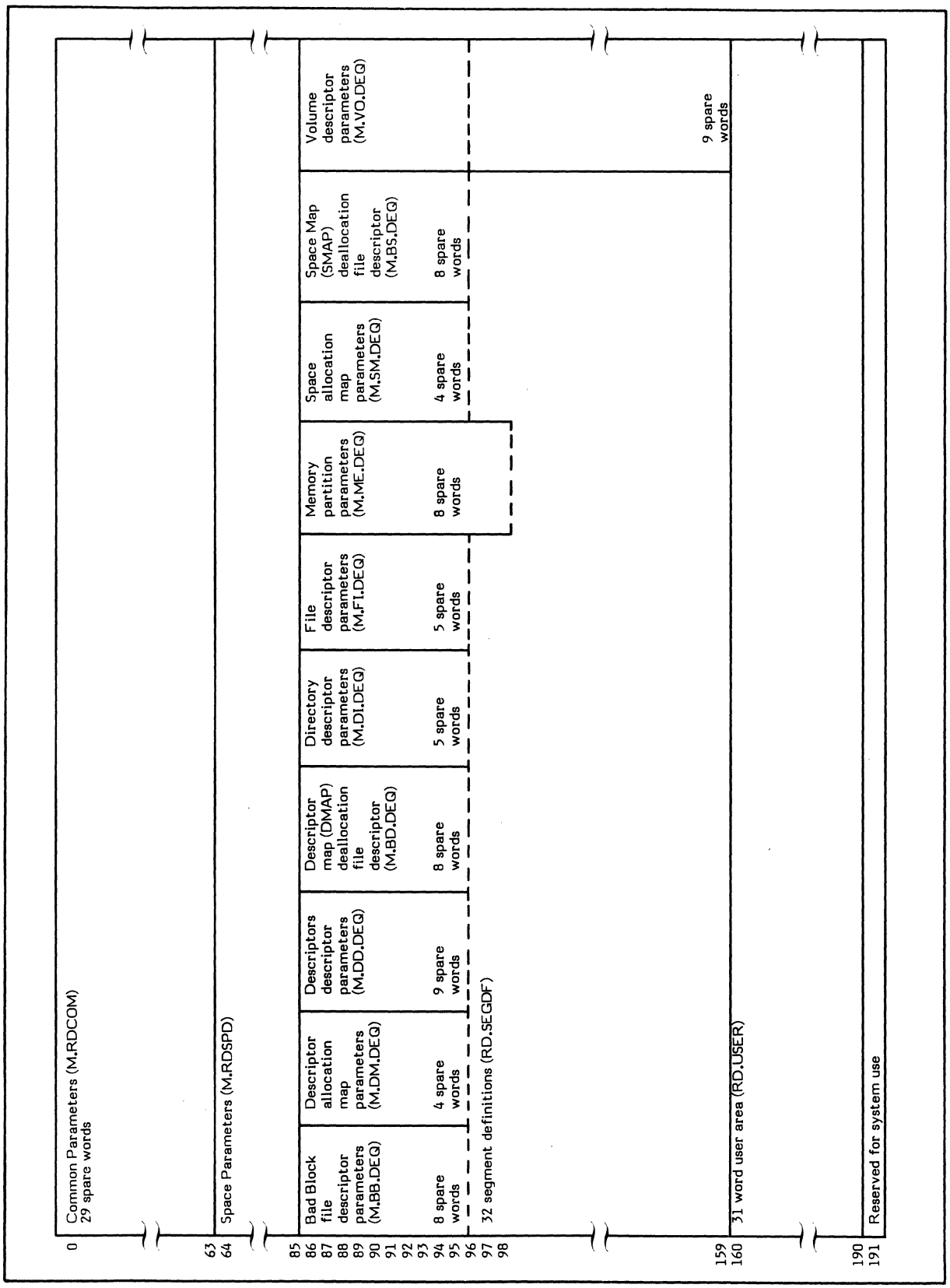


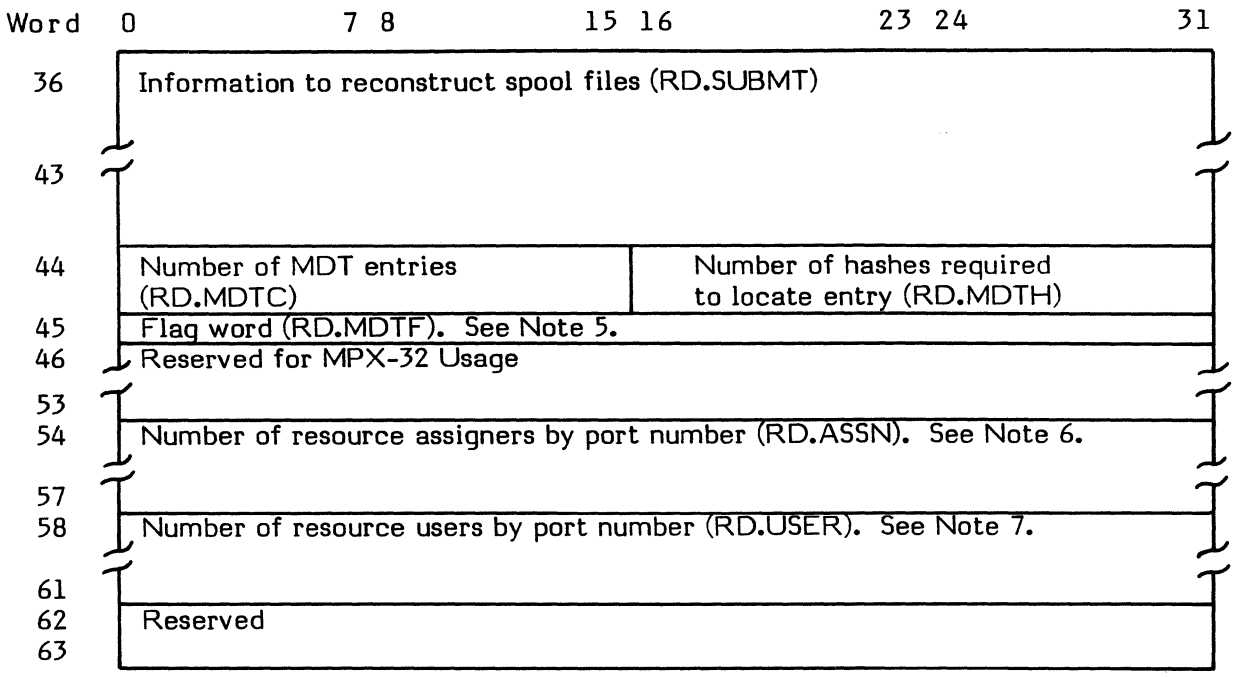
Figure 2-7. Disc Resident Resource Descriptor Table

2.41.1 Resource Descriptor (M.RDCOM)

The Resource Descriptor (M.RDCOM) defines the system common portion of a resource descriptor. The common area ends at 64W, immediately followed by the information specific to the resource descriptor type.

Words 0 to 7 correspond to M.RDID, words 8 to 25 correspond to M.RDACT, words 26 to 32 correspond to M.RDACC, and words 33 to 64 define the balance of the common area.

Word	0 7 8 15 16 23 24 31
0	Volume name (RD.IDNAM)
3	
4	Binary creation date (RD.DATE)
5	Binary creation time (RD.TIME)
6	Absolute block number of resource descriptor (RD.DOFF)
7	Resource ID flags (RD.RDFLG). See Note 1. / Resource type (numeric value) (RD.RTYPE). See Note 2.
8	Binary date of creation/deletion (RD.CRDAT)
9	Binary time of creation/deletion (RD.CRTIM)
10	Binary date of expiration (RD.XPDAT)
11	Binary time of expiration (RD.XPTIM)
12	Binary date of last read access (RD.RDDAT)
13	Binary time of last read access (RD.RDTIM)
14	Binary date file last changed (RD.CHDAT)
15	Binary time file last changed (RD.CHTIM)
16	Binary date of last save (RD.SVDAT)
17	Binary time of last save (RD.SVTIM)
18	Binary date of last restore (RD.RSDAT)
19	Binary time of last restore (RD.RSTIM)
20	Owner name of last changer (RD.CHOWN)
21	
22	Owner name of creator (RD.CROWN)
23	
24	Count of opens in read mode (RD.RDCNT)
25	Accounting flags (RD.AFLGS). See Note 3.
26	Name of resource owner (RD.OWNR)
27	
28	Name of resource project group (RD.UGRP)
29	
30	Owner access/privileges (RD.AOWNR). See Note 4.
31	Project group access/privileges (RD.AUGRP). See Note 4.
32	Others access/privileges (RD.AOTHR). See Note 4.
33	Reserved
34	Resource link count (RD.LNKCT)
35	Port number (1 through 16) of task that opened resource for write access (RD.WRID) / Port number (1 through 16) of task that opened resource for modify access (RD.MDID) / Port number (1 through 16) of task that opened resource for update access (RD.UPID) / Port number (1 through 16) of task that opened resource for append access (RD.APID)



Notes:

1. Internal flags reserved for MPX-32.

2. Values for RD.RTYPE are as follows:

Value	Meaning
1	Volume type (RD.VOL)
2	Resource descriptor description (RD.RESRC)
3	Descriptor map descriptor (RD.DMAP)
4	Space map descriptor (RD.SMAP)
5	Root directory descriptor (RD.ROOT)
6	System image descriptor (RD.IMAGE)
7	Bad block descriptor (RD.BDBLK)
8	Value for spool file (RD.SPOOL)
9	Reserved
10	Permanent file or shared image (RD.FILE)
11	Permanent directory (RD.DIR)
12	Temporary file (RD.TFILE)
13	Temporary directory (RD.TDIR)
14	Static memory partition (RD.MEM)
15	Dynamic memory partition (RD.TMEM)
16	Device descriptor (RD.DEVC)
17	Resource descriptor for the DMAP bad block deallocation file (RD.BDMAP)
18	Resource descriptor for the SMAP bad block deallocation file (RD.BSMAP)

3. Bits in RD.AFLGS are assigned as follows:

Bits	Meaning if Set
0	Record last read information (RD.AREAD)
1	RD is being deleted (RD.DEL)
2-31	Reserved

4. Bits in RD.AOWNR, RD.AUGRP and RD.AOTHR are assigned as follows:

Bits	Meaning if Set
0	Read access allowed (RD.READ)
1	Write access allowed (RD.WRITE)
2	Modify access allowed (RD.MODFY)
3	Update access allowed (RD.UPDAT)
4	Append access allowed (RD.APPND)
5-7	Reserved
8	Traverse directory access allowed (RD.TRAVR)
9	Delete resource access allowed (RD.DELET)
10	Delete directory entry access allowed (RD.DEENT)
11	Add directory entry access allowed (RD.ADENT)
12-31	Reserved

5. Bits in RD.MDTF are assigned as follows:

Bits	Meaning if Set
0	MDT entry is in use
1-31	Reserved

6. Bytes in RD.ASSNS are as follows:

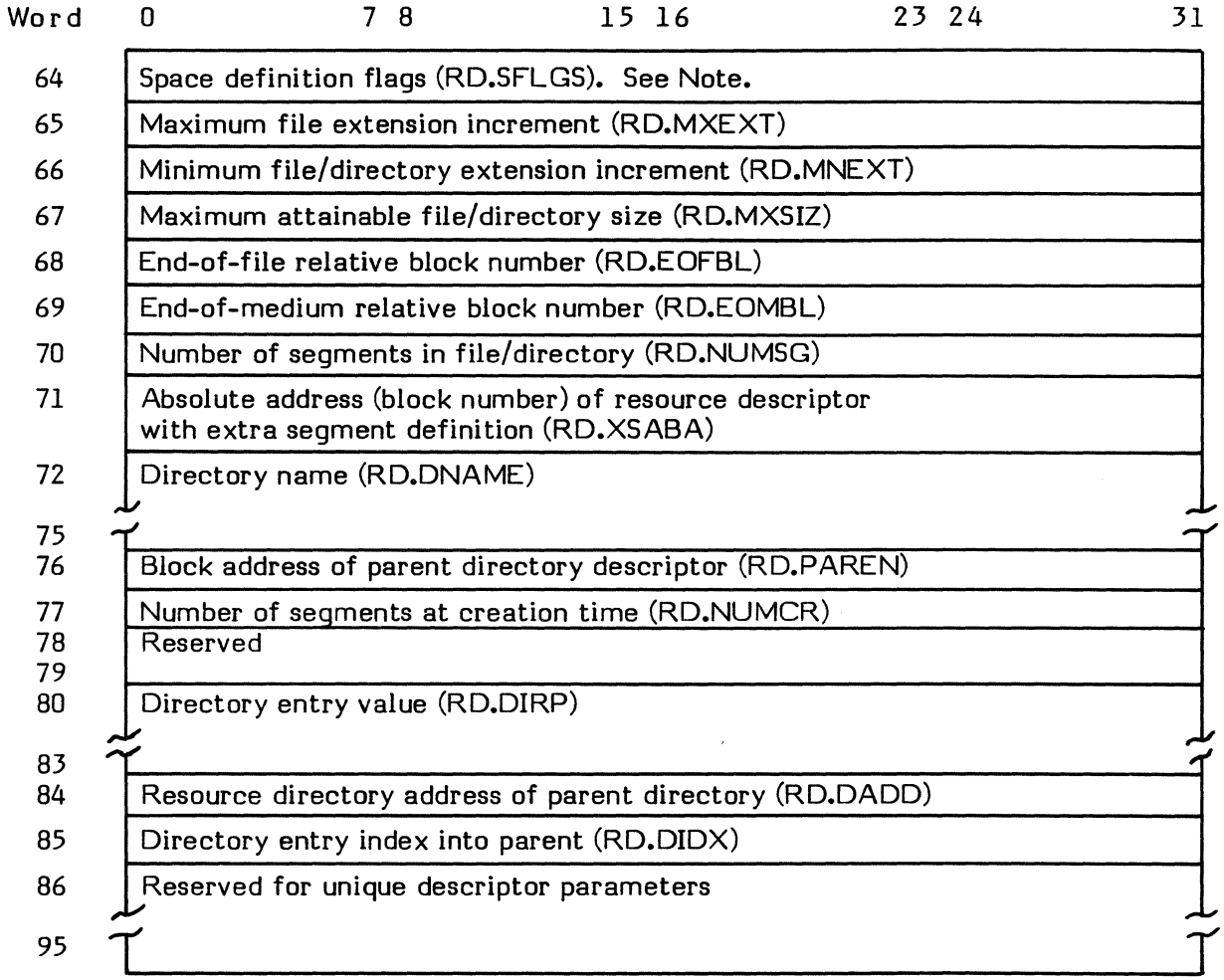
Byte	Meaning
0	Number of assigners on port 0
1	Number of assigners on port 1
2	Number of assigners on port 2
3	Number of assigners on port 3
4	Number of assigners on port 4
5	Number of assigners on port 5
6	Number of assigners on port 6
7	Number of assigners on port 7
8	Number of assigners on port 8
9	Number of assigners on port 9
10	Number of assigners on port 10
11	Number of assigners on port 11
12	Number of assigners on port 12
13	Number of assigners on port 13
14	Number of assigners on port 14
15	Number of assigners on port 15

7. Bytes in RD.USERS are as follows:

Byte	Meaning
0	Number of users on port 0
1	Number of users on port 1
2	Number of users on port 2
3	Number of users on port 3
4	Number of users on port 4
5	Number of users on port 5
6	Number of users on port 6
7	Number of users on port 7
8	Number of users on port 8
9	Number of users on port 9
10	Number of users on port 10
11	Number of users on port 11
12	Number of users on port 12
13	Number of users on port 13
14	Number of users on port 14
15	Number of users on port 15

2.41.2 Resource Descriptor Space Definition (M.RDSPD)

The Resource Descriptor Space Definition (M.RDSPD) descriptor defines a resource descriptor space definition area.



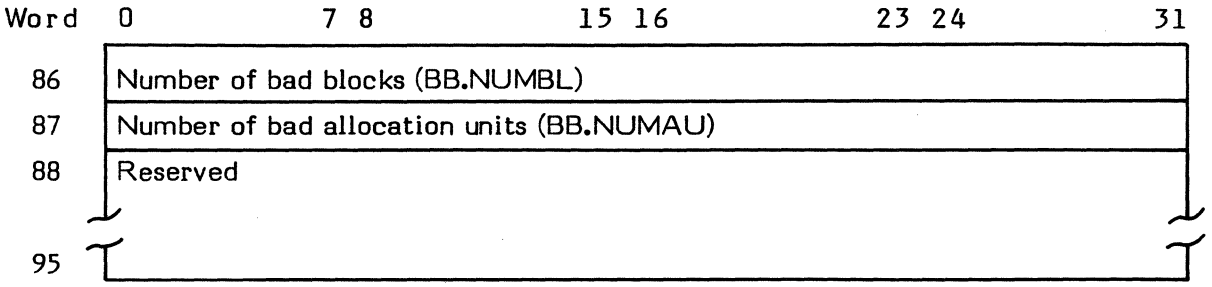
Note: Bits in RD.SFLGS are assigned as follows:

Bits	Meaning if Set
0-7	Resource type, equivalent to file type code, interpreted as two hexadecimal digits, 0-FF (RD.FTYPE)
8-10	Reserved
11	EOF management required (RD.EOFM)
12	Fast access file (RD.FAST)
13	No-save file (RD.NSAVE)
14	File contains default image (RD.DEFI)
15	Reserved
16	Execute access (RD.EXEC)

Bits	Meaning if Set
17	Set owner ID on access (RD.OWNID)
18	Set project group ID on access (RD.USRID)
19	Restoring this file (RD.PREFR)
20-22	Reserved
23	Zero file on creation/expansion (RD.ZERO)
24	Automatically extendible (RD.AUTO)
25	Manually extendible (RD.MANUL)
26	Contiguity desired (RD.CONTG)
27	Shareable access allowed (RD.SHRBL)
28	Link access (RD.LINK)
29	File physically/logically contiguous (RD.NCNTG)
30	File written to DFI (RD.DFI)
31	File data is recorded as blocked (RD.BLOCK)

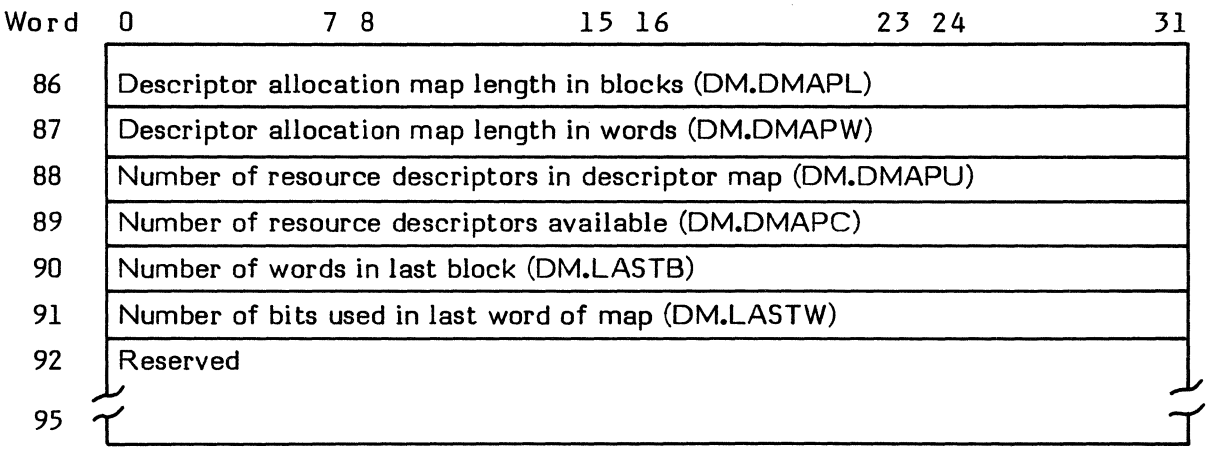
2.41.3 Bad Block Descriptor (M.BB.DEQ)

The Bad Block Descriptor (M.BB.DEQ) defines the bad block file descriptor.



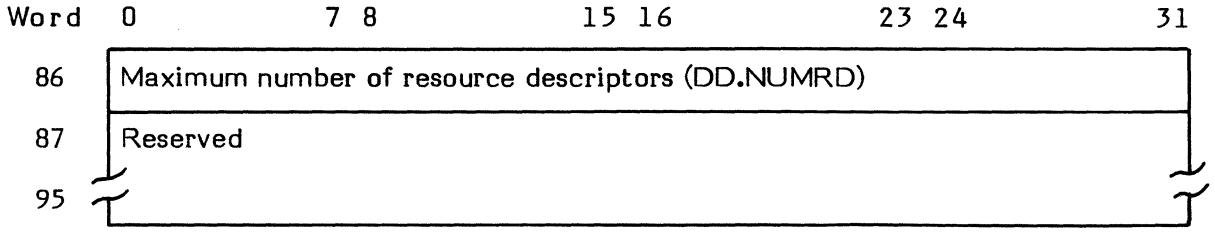
2.41.4 Descriptor Allocation Map Descriptor (M.DM.DEQ)

The Descriptor Allocation Map Descriptor (M.DM.DEQ) defines the descriptor map descriptor block.



2.41.5 Descriptors Descriptor (M.DD.DEQ)

The Descriptors Descriptor (M.DD.DEQ) defines the descriptor for the resource descriptors.



2.41.6 Descriptor Map (DMAP) Deallocation File Descriptor (M.BD.DEQ)

The DMAP Deallocation File Descriptor (M.BD.DEQ) defines the DMAP deallocation file descriptor. Each entry in the DMAP deallocation file is 4 bytes in length (BD.ESIZE) and contains the disc block number of the block (resource descriptor) to be deallocated.

Word	0 7 8 15 16 23 24 31			
86	Reserved			
87	Total number of entries that can be specified in the bad DMAP deallocation file (BD.NENTR)			
88	Total number of entries available in the bad DMAP deallocation file (BD.AVAIL)			
89	Reserved			
91				
92	Number of resource assigners by the dual-port number (RD.ASSNS or RD.ART). See Note 1.		Number of resource users by the dual-port number (RD.USERS). See Note 2.	
93	Current multi-port access mode (RD.CACM)	Port number of resource lock owner (RD.MPID)	Reserved	
94	AR.FLAGS. See Note 3.		AR.XRL. See Note 3.	AR.SRL. See Note 3.
95	AR.ASSNS. See Note 3.	AR.USERS. See Note 3.	Reserved	AR.RDRS. See Note 3.

Notes:

1. Bytes in RD.ASSNS are as follows:

Byte	Description
0	Number of assigners on port 0
1	Number of assigners on port 1

RD.ART is equivalent to RD.ASSNS, and indexes the ART information stored in words 94 and 95.

2. Bytes in RD.USERS are as follows:

Byte	Description
2	Number of users on port 0
3	Number of users on port 1

3. For detailed description of field contents, refer to the corresponding field within the Allocated Resource Table (ART) in this chapter.

2.41.7 Directory Descriptor (M.DI.DEQ)

The Directory Descriptor (M.DI.DEQ) defines the directory descriptor block (also pertains to the root directory).

Word	0 7 8 15 16 23 24 31			
86	Number of entries per block (DI.ENTSG)			
87	Current number of active entries (DI.ACTIV)			
88	Total entry capacity (DI.TOTEN)			
89	Current number of available entries (DI.AVLEN)			
90	Directory descriptor flags (DI.FLAGS)			
91	Reserved			
92	Number of resource assigners by the dual-port number (RD.ASSNS or RD.ART). See Note 1.		Number of resource users by the dual-port number (RD.USERS). See Note 2.	
93	Current multi-port access mode (RD.CACM)	Port number of resource lock owner (RD.MPID)	Reserved	
94	AR.FLAGS. See Note 3.		AR.XRL. See Note 3.	AR.SRL. See Note 3.
95	AR.ASSNS. See Note 3.	AR.USERS. See Note 3.	Reserved	AR.RDRS. See Note 3.

Notes:

1. Bytes in RD.ASSNS are as follows:

Byte	Description
0	Number of assigners on port 0
1	Number of assigners on port 1

   RD.ART is equivalent to RD.ASSNS, and indexes the ART information stored in words 94 and 95.

2. Bytes in RD.USERS are as follows:

Byte	Description
2	Number of users on port 0
3	Number of users on port 1

3. For detailed description of field contents, refer to the corresponding field within the Allocated Resource Table (ART) in this chapter.

2.41.8 File Descriptor (M.FI.DEQ)

The File Descriptor (M.FI.DEQ) defines the file descriptor block (also pertains to descriptors for the system image).

Word	0 – 7	8 – 15	16 – 23	24 – 31
86 – 90	Reserved			
91	Assign count Port 0. See Note 1.	Assign count Port 1. See Note 1.	Use count Port 0. See Note 1.	Use count Port 1. See Note 1.
92	Number of resource assigners by the dual-port number (RD.ASSNS or RD.ART). See Note 2.		Number of resource users by the dual-port number (RD.USERS). See Note 3.	
93	Current multi-port access mode (RD.CACM)	Port number of resource lock owner (RD.MPID)	Reserved	
94	AR.FLAGS. See Note 4.		AR.XRL. See Note 4.	AR.SRL. See Note 4.
95	AR.ASSNS. See Note 4.	AR.USERS. See Note 4.	Reserved	AR.RDRS. See Note 4.

Notes:

1. Interpreted for explicit shared use only.

2. Bytes in RD.ASSNS are as follows:

Byte	Description
0	Number of assigners on port 0
1	Number of assigners on port 1

   RD.ART is equivalent to RD.ASSNS, and indexes the ART information stored in words 94 and 95.

3. Bytes in RD.USERS are as follows:

Byte	Description
2	Number of users on port 0
3	Number of users on port 1

4. For detailed description of field contents, refer to the corresponding field within the Allocated Resource Table (ART) in this chapter.

2.41.9 Memory Partition Descriptor (M.ME.DEQ)

The Memory Partition Descriptor (M.ME.DEQ) defines the memory partition descriptor.

Word	0 7	8 15	16 23	24 31
86	Starting physical page number (ME.PPAGE)			
87	Memory class (ME.MCLAS)			
88	Starting logical page number (RD.LPAGE)			
89	Total number of pages (RD.PGLEN)			
90	Owner access rights (ME.AOWNR)	Project group access rights (ME.AUGRP). See Note.	Others access rights (ME.AOTHR). See Note.	Reserved
91	Reserved			
97				

Note: Access rights apply to the shared image loaded into memory. Bits in ME.AOWNR, ME.AUGRP, and ME.AOTHR are assigned as follows:

Bits	Meaning if Set
0	Read access
1	Write access
2-7	Reserved

2.41.10 Space Allocation Map Descriptor (M.SM.DEQ)

The Space Allocation Map Descriptor (M.SM.DEQ) defines the space map descriptor block.

Word	0 7	8 15	16 23	24 31
86	Space allocation map length in blocks (SM.SMAPL)			
87	Space allocation map length in words (SM.SMAPW)			
88	Number of allocation units in space map (SM.SMAPU)			
89	Number of allocation units available (SM.SMAPC)			
90	Number of words in last block (SM.LASTB)			
91	Number of bits used in last word of map (SM.LASTW)			
92	Reserved			
95				

2.41.11 Space Map (SMAP) Deallocation File Descriptor (M.BS.DEQ)

The SMAP Deallocation File Descriptor (M.BS.DEQ) defines the SMAP deallocation file descriptor. Each entry in the SMAP deallocation file is eight bytes in length (BS.ESIZE) and contains the segment definition of the disc space to be deallocated.

Word	0 7 8 15 16 23 24 31			
86	Reserved			
87	Total number of entries that can be specified in the bad SMAP deallocation file (BS.NENTR)			
88	Total number of entries available in the bad SMAP deallocation file (BS.AVAIL)			
89	Reserved			
91				
92	Number of resource assigners by the dual-port number (RD.ASSNS or RD.ART). See Note 1.		Number of resource users by the dual-port number (RD.USERS). See Note 2.	
93	Current multi-port access mode (RD.CACM)	Port number of resource lock owner (RD.MPID)	Reserved	
94	AR.FLAGS. See Note 3.		AR.XRL. See Note 3.	AR.SRL. See Note 3.
95	AR.ASSNS. See Note 3.	AR.USERS. See Note 3.	Reserved	AR.RDRS. See Note 3.

Notes:

1. Bytes in RD.ASSNS are as follows:

Byte	Description
0	Number of assigners on port 0
1	Number of assigners on port 1

RD.ART is equivalent to RD.ASSNS, and indexes the ART information stored in words 94 and 95.

2. Bytes in RD.USERS are as follows:

Byte	Description
2	Number of users on port 0
3	Number of users on port 1

3. For detailed description of field contents, refer to the corresponding field within the Allocated Resource Table (ART) in this chapter.

2.41.12 Volume Descriptor (M.VO.DEQ)

The Volume Descriptor (M.VO. DEQ) defines the volume descriptor block.

Word	0 7 8 15 16 23 24 31
86	Binary date of last mount (VO.MTDAT)
87	Binary time of last mount (VO.MTTIM)
88	Binary date of last dismount (VO.DTDAT)
89	Binary time of last dismount (VO.DTTIM)
90	Total number of blocks on volume (VO.TOTBL)
91	Total number of allocation units on volume (VO.TOTAU)
92	Number of allocation units for user allocation (VO.USRAU)
93	Number of blocks available for user allocation (VO.USRBL)
94	Space map start address (VO.SMAPS)
95	Space allocation map length in blocks (VO.SMAPL)
96	Number of allocation units reflected in space map (VO.SMAPU)
97	Number of allocation units currently available (VO.SMAPC). See Note 1.
98	Descriptor allocation map start address (VO.DMAPS)
99	Descriptor allocation map length in blocks (VO.DMAPL)
100	Number of resource descriptors in descriptor map (VO.DMAPU)
101	Number of resource descriptors currently available (VO.DMAPC)
102	Root directory segment definition start (VO.ROOTS)
103	Root directory segment definition size (VO.ROOTL)
104	Blocks per allocation unit (VO.BLKAU)
105	Number of blocks in system area (VO.SYSBL)
106	Number of allocation units occupied by system (VO.SYSAU)
107	Number of blocks of resource descriptors (VO.RESBL)
108	Volume flags (VO.FLAGS). See Note 2.
109	Reserved
110	
111	
112	Number of IOCDs to boot maximum size system (VO.IOCD1 or VO.BTDSC)
	≈ ≈
121	
122	Number of blocks in restart code (VO.RNRST) / Number of blocks in restart image (VO.SGNOS)
123	Restart/ bootstrap flags (VO.RFLGS). See Note 3. / Index to selected image for rest (VO.IMGSL) / Reserved
124	SYSGEN (cold start) image name (VO.PRIMG)
125	
126	SYSGEN (cold start) starting block (VO.PRSTB)
127	SYSGEN (cold start) number of blocks (VO.PRNBK)
128	SYSGEN (cold start) image checks (VO.PRCKS)
129	SYSGEN (cold start) number of bytes (VO.PRBCT)
130	Current selected image name (VO.CUIMG)
131	
132	Current selected starting block number (VO.CUSTB)
133	Current selected number of blocks (VO.CUNBK)
134	Current selected image checksum (VO.CUCKS)
135	Current selected number of bytes (VO.CUBCT)

Word	0 7	8 15	16 23	24 31
136	Sectors per block (VO.SPB)	Sectors per track (VO.SPT)	Sectors per cylinder (VO.SPC)	
137	Device type (VO.DEVT). See Note 4.	Number of heads (VO.NHDS)	Sector size (VO.SSIZ)	
138	Device class (VO.CLASS)	Reserved	Channel/subaddress for boot/restart (VO.CHNSA)	
139	Boot device (VO.IPLDV)			
140	Reserved for future development use (VO.IOCD2)			
141				
142	Drive attribute information (VO.DATR)			
150				
151	Seek information for boot of primary image (VO.SEEK)			
152	Long resource identifier of current default system image (doubleword bounded) (VO.LRID)			
159				

Notes:

1. This number includes some blocks allocated by the operating system.
2. Bits in VO.FLAGS are assigned as follows:

Bits	Meaning if Set
0	Volume currently mounted (VO.MONTD)
1	Volume is mounted under MP(DP)0 (VO.PORT0)
2	Volume is mounted under MP(DP)1 (VO.PORT1)
3	Volume is mounted under MP2
4	Volume is mounted under MP3
5	Volume is mounted under MP4
6	Volume is mounted under MP5
7	Volume is mounted under MP6
8	Volume is mounted under MP7
9	Volume is mounted under MP8
10	Volume is mounted under MP9
11	Volume is mounted under MPA
12	Volume is mounted under MPB
13	Volume is mounted under MPC
14	Volume is mounted under MPD
15	Volume is mounted under MPE
16	Volume is mounted under MPF
17-31	Reserved

3. Bits in VO.RFLGS are assigned as follows:

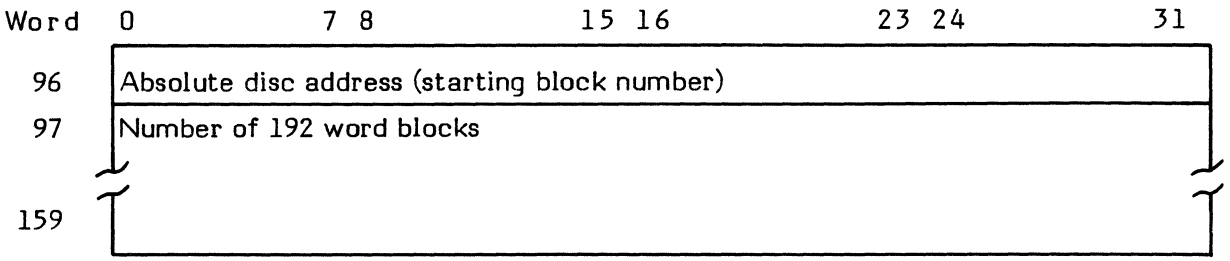
Bits	Meaning if Set
0	Program restart flag (VO.PGMRS)
1	Default image established flag (VO.DEFLT)
2	Bootstrap in retry mode (VO.BRTRY)
3	Automatic default flag (VO.AUTO)
4	Debugger requested flag (VO.DEBUG)
5-7	Reserved

4. Bits in VO.DEVT are assigned as follows:

Bits	Meaning if Set
0	Moving head disc (VO.MHDDT)
1	Fixed head disc (VO.FHDDT)
2	Cartridge module drive (VO.CMDDT)
3	Reserved
4	Device not present (VO.NPRES)
5	Multi-/dual-port disc (VO.DUAL)
6-7	Reserved

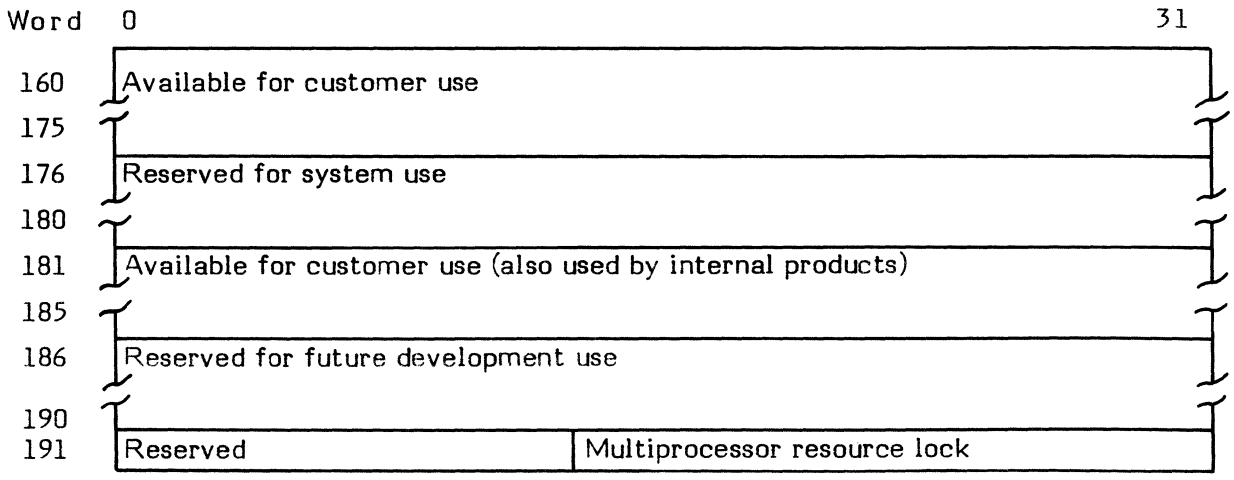
2.41.13 Segment Definitions (RD.SEGDF)

The Segment Definition (RD.SEGDF) total area covers words 96 to 159 of the M.RDSPD. Each entry in RD.SEGDF is a doubleword entry, with a maximum of 32 entries.



2.41.14 User Area (RD.USER)

The User Area (RD.USER) covers words 160 to 191 of M.RDSPD. Words 160 to 190 can be modified by users; however, words 176 to 190 are used by some internal products. Therefore, users are advised to use only words 160 to 175.



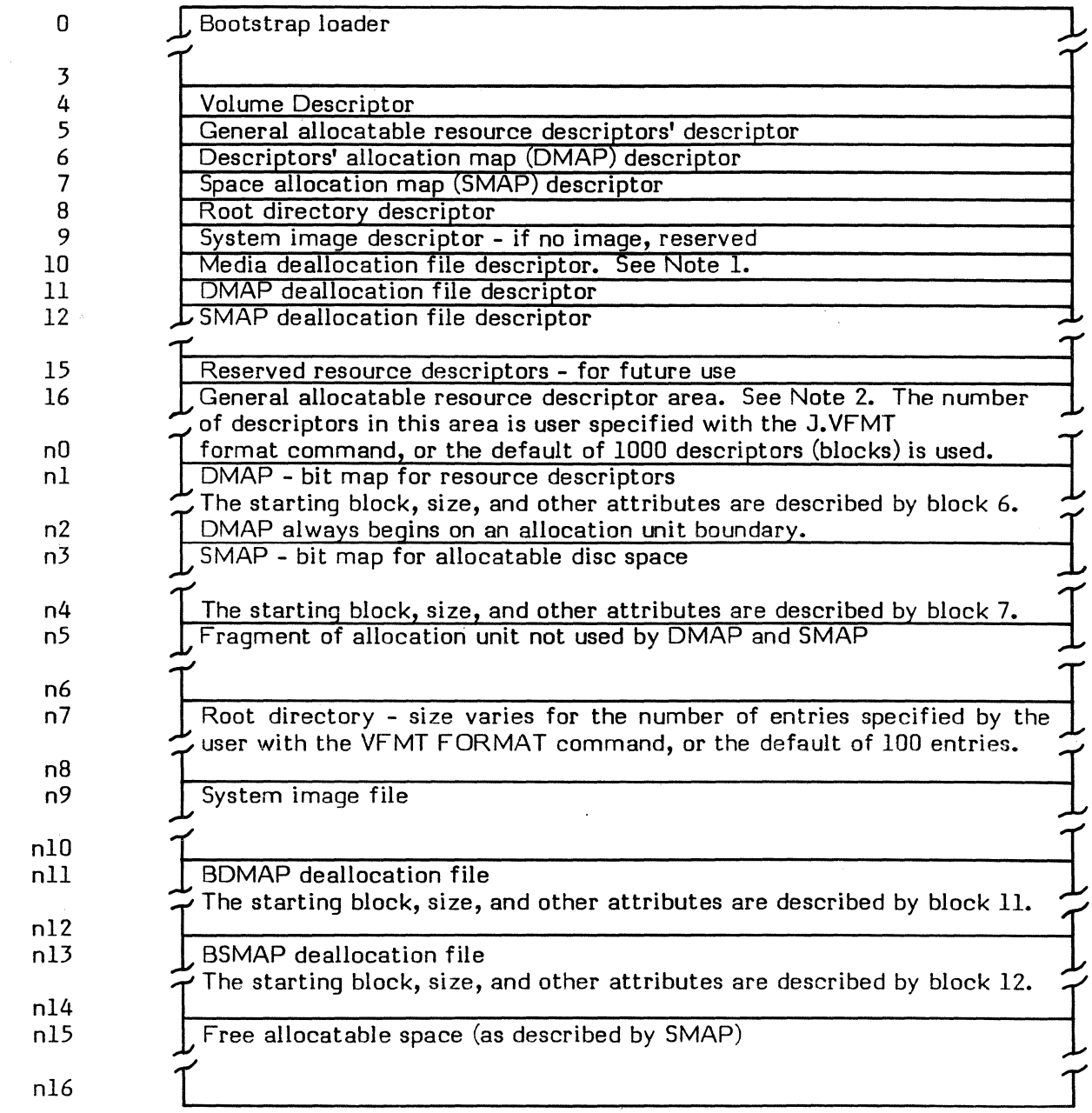
2.42 Disc Resident Structures

The following are disc resident structures in the order they appear in this section.

- Volume Format
- Load Module Structure
- Load Module Preamble
- Executable Image Structure
- Executable Image Preamble
- Shared Executable Image Structure
- Shared Executable Image Preamble
- Shared Image Descriptor

2.42.1 Volume Format

After a volume is formatted by J.VFMT (volume formatter), the volume has the following format:

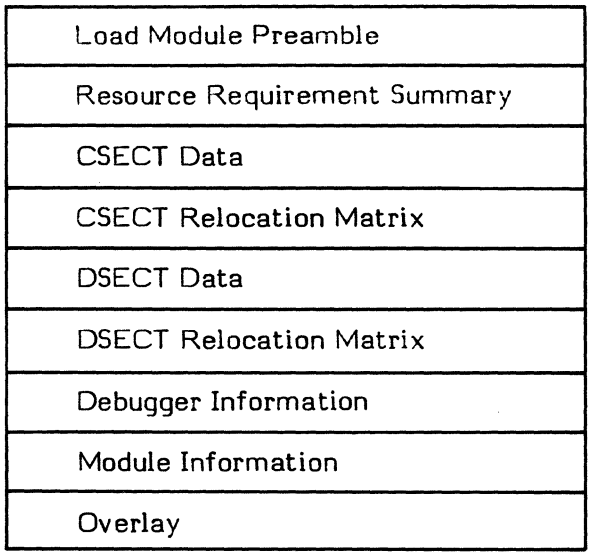


Notes:

1. The cylinder closest to the spindle of all disc packs is reserved for the disc vendor's use.
2. This area is described by the DMAP area (blocks n1 through n2). Block 16 is reserved for the system directory resource descriptor.

2.42.2 Load Module Structure

A load module is a permanent file of cataloged nonbase mode object code. The following is the load module structure.

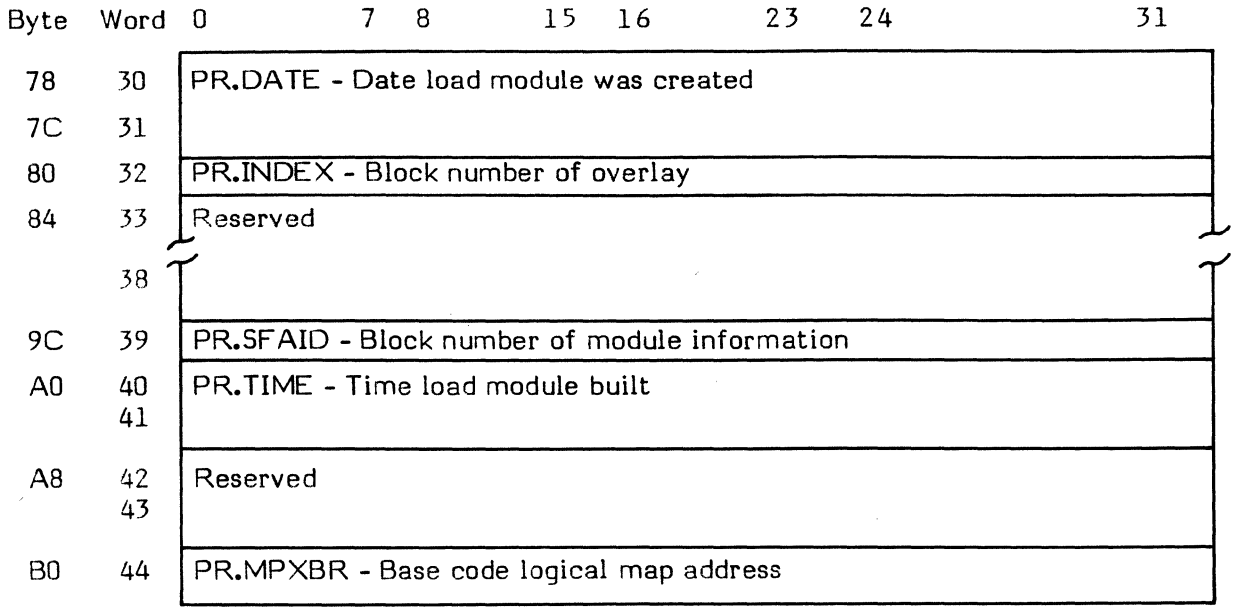


If either the CSECT or DSECT segments are empty, the empty segments are omitted.

2.42.3 Load Module Preamble

The load module preamble contains load module information in the following format:

Byte	Word	0 7	8 15	16 23	24 31
0	0	PR.NAME - Load module name			
4	1				
8	2	PR.USER - User name			
0C	3				
10	4	PR.MOUNT	Reserved	PR.KEY	
14	5	PR.TRAN - Module transfer address			
18	6	PR.CNTL	PR.FLAG	PR.NRRS	PR.PRIOR
1C	7	PR.PAGEC	PR.PAGED	PR.PGSIZ	PR.MEMS
20	8	PR.PAGEG	PR.FILE	PR.BUFR	PR.SEGS
24	9	PR.OPTN - Program option word			
28	10	PR.ORGC - Starting byte address of code section			
2C	11	PR.COMC	Reserved for big blocking buffers	Reserved for big blocking buffers	Reserved
30	12	PR.ENDC - Ending byte address of code section			
34	13	PR.SFAC - Relative file address of code section			
38	14	PR.BYTEC - Number of bytes in code section			
3C	15	PR.CHKC - Code section checksum			
40	16	PR.SFACR - Address of code section relocation matrix			
44	17	PR.BYTCR - Number of bytes in matrix			
48	18	PR.CHKCR - Matrix checksum			
4C	19	PR.ORGD - Starting byte address of data section			
50	20	PR.COMD - Common delta			
54	21	PR.ENDD - Ending byte address of data section			
58	22	PR.SFAD - Relative file address of data section			
5C	23	PR.BYTED - Number of bytes in data section			
60	24	PR.CHKD - Data section checksum			
64	25	PR.SFADR - Address of data section relocation matrix			
68	26	PR.BYTDR - Number of bytes in matrix			
6C	27	PR.CHKDR - Matrix checksum			
70	28	PR.SYMG - Global symbol block number			
74	29	PR.SYMP - Local symbol block number			


Byte
Word
0
7
8
15
16
23
24
31
78
30
PR.DATE - Date load module was created
7C
31
80
32
PR.INDEX - Block number of overlay
84
33
Reserved
38
9C
39
PR.SFAID - Block number of module information
A0
40
PR.TIME - Time load module built
41
A8
42
Reserved
43
B0
44
PR.MPXBR - Base code logical map address

Byte (Hex)	Name	Description
0	PR.NAME	Load module name; Field length = 2W; Left-justified and blank-filled eight-character ASCII; This name must match the file name.
8	PR.USER	User name; Field length = 2W; Left-justified and blank-filled eight-character ASCII; Zero if the USERNAME directive was not used.
10	PR.MOUNT	Mounts; Field length = 1B; Contains the number from the VOLUMES directive; Default value is zero.
	RESERVED	Field length = 1B.
	PR.KEY	User key; Field length = 1H; Compressed ASCII; Zero if the USERNAME directive was not used.
14	PR.TRAN	Module transfer address; Field length = 1W; Module relative address where the module is to start execution when loaded; Bit 7 set indicates the transfer address is absolute.
18	PR.CNTL	Control; Field length = 1B;

Bits	Description
0	Overlay is in separate file format (PR.OVLS)
1	No overlays (PR.NOVL)
2	Reserved
3	Privileged task (PR.PRIV)
4	System administrator attribute (PR.SAM)
5-7	Reserved

PR.FLAG — Flags;
Field length = 1B;

Bit	Description
0	Absolute CSECT load addresses (PR.ABSC)
1	Resident (program cannot be swapped out when in execution) (PR.RES)
2	Shared (a single copy can be shared by two or more users) (PR.SHR)
3	Absolute DSECT load addresses (PR.ABSD)
4	Do not attach debugger (PR.NODBG)
5	Multicopy (PR.MULTI)
6	Load base mode debugger
7	MPX-32 Release 2.0+ load module (PR.MPX20)

PR.NRRS — RRS count;
Field length = 1B;
Number of entries in the Resource Requirement Summary Table.

PR.PRIOR — Priority;
Field length = 1B;
Base execution priority of the load module.

1C PR.PAGEC — CSECT pages;
Field length = 1B;
Number of 512-word pages in CSECT.

PR.PAGED — DSECT pages;
Field length = 1B;
Number of 512-word pages in DSECT;
Derived from the ending address of the DSECT (PR.ENDD).

PR.PGSIZ — Block size;
Field length = 1B;
Defines the map block granularity required as specified in the environment directive. It is the number of 512-word protection granules in a map block (four hexadecimal).

PR.MEMS — Memory class;
Field length = 1B;
The value indicates memory class as follows: one for E-class memory, two for H-class memory, and three for S-class memory. The default value is three.

20 PR.PAGEG — Common pages;
Field length = 1B;
Number of 512-word pages in Global Common/Datapool.

PR.FILE — Files;
Field length = 1B;
Number from the FILES directive;
Default value is five (the requirement for the debugger).

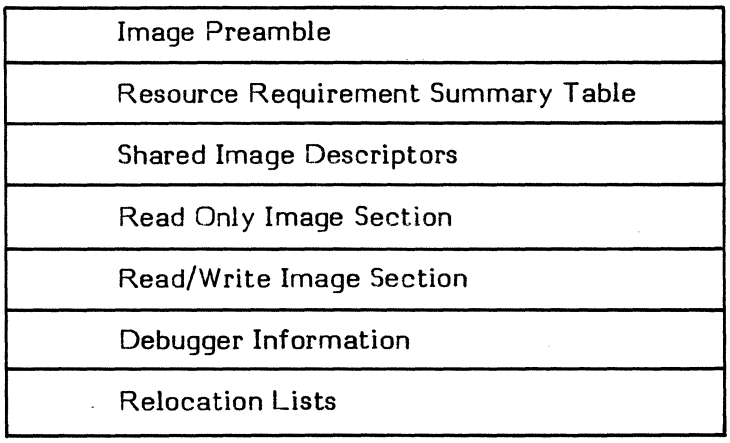
	PR.BUFR	Buffers; Field length = 1B; Number from the BUFFERS directive; Default value is three (the requirement for the debugger).
	PR.SEGS	Segmented files; Field length = 1B; Number from the SEGFILES directive; Default is PR.FILE.
24	PR.OPTN	Program option word; Field length = 1W;
28	PR.ORGC	CSECT origin; Field length = 1W; Address at which to begin loading the CSECT; Bit 7 set indicates absolute origin.
2C	PR.COMC	CSECT common delta; Field length = 1B; Static buffer count; Field length = 1B. Head cell count; Field length = 1B.
	Reserved	Field length = 1B.
30	PR.ENDC	CSECT ending address; Field length = 1W; Address of the first free word after the CSECT.
34	PR.SFAC	CSECT data block number; Field length = 1W; File relative block number of the CSECT data.
38	PR.BYTEC	CSECT data byte count; Field length = 1W; Number of bytes of CSECT data.
3C	PR.CHKC	CSECT data checksum; Field length = 1W; Checksum for the CSECT data; All halfwords of CSECT data are summed in a register.
40	PR.SFACR	CSECT relocation matrix block number; Field length = 1W; File relative block number of the CSECT relocation matrix.
44	PR.BYTCR	CSECT relocation matrix byte count; Field length = 1W; Number of bytes, rounded up to a multiple of four, of CSECT relocation matrix.

48	PR.CHKCR	CSECT relocation matrix checksum; Field length = 1W; Checksum for the CSECT relocation matrix; All halfwords of CSECT relocation matrix are summed in a register.
4C	PR.ORGD	DSECT origin; Field length = 1W; Address at which to begin loading the DSECT; Bit 7 set indicates absolute origin.
50	PR.COMD	DSECT common delta; Field length = 1W; Not used, contains zero.
54	PR.ENDD	DSECT ending address; Field length = 1W; The address of the first free word after the DSECT.
58	PR.SFAD	DSECT data block number; Field length = 1W; File relative block number of the DSECT data.
5C	PR.BYTED	DSECT data byte count; Field length = 1W; Number of bytes of DSECT data.
60	PR.CHKD	DSECT data checksum; Field length = 1W; Checksum for the DSCECT data. All halfwords of DSECT data are summed in a register.
64	PR.SFADR	DSECT relocation matrix block number; Field length = 1W; File relative block number of the DSECT relocation matrix.
68	PR.BYTDR	DSECT relocation matrix byte count; Field length = 1W; Number of bytes, rounded up to a multiple of four, of DSECT relocation matrix.
6C	PR.CHKDR	DSECT relocation matrix checksum; Field length = 1W; Checksum for the DSECT relocation matrix. All halfwords of DSECT relocation matrix are summed in a register.
70	PR.SYMG	Global symbol block number; Field length = 1W; Relative sector number of the global symbol table; Zero indicates no symbols.
74	PR.SYMP	Local symbol block number; Field length = 1W; Relative sector number of the local symbol table; Zero indicates no symbols.

78	PR.DATE	Date of creation; Field length = 2W; Date load module was created; Format identical to C.DATE.
80	PR.INDEX	Index block; Field length = 1W; Relative block number of the overlay index for single file load modules; Zero indicates no overlays.
84	Reserved	Field length = 6W.
9C	PR.SFAID	Block number of load module information; Field length = 1W.
A0	PR.TIME	Time load module built; Field length = 2W.
A8	PR.MPXBR	Base code logical map address; Field length = 1W.

2.42.4 Executable Image Structure

An executable image is a permanent file of base mode object code that was built by the Linker/X32. The following is the nonshared executable image structure.



2.42.5 Executable Image Preamble

The nonshared executable image preamble contains image information in the following format:

Byte	Word	0 7	8 15	16 23	24 31
0	0	PR.LVER - Linker version number			
4	1	PR.IVER - Image version number			
8	2	PR.ROSIZ - Number of bytes in read only section			
0C	3	PR.RWSIZ - Number of bytes in read/write section			
10	4	PR.MNT	PR.FLAG2	PR.NSI	
14	5	PR.TRAN - Transfer address			
18	6	PR.CNTL	PR.FLAG	PR.NNRS	PR.BPRI
1C	7	PR.NOOL		PR.IPRI	PR.RRSSZ
20	8	PR.RRS	PR.FILE	PR.BUFR	PR.SEGS
24	9	PR.OPTN - Program option word			
28	10	PR.RESVD	PR.SSIZ - Stack size in bytes		
2C	11	PR.OLD - File address of overlay descriptors			
30	12	PR.GST - File address of global symbol table			
34	13	PR.DBG - File address of debugger information			
38	14	PR.ROSD - Read only image section descriptor			
4C	19				
50	20	PR.RWSD - Read/write image section descriptor			
64	25				
68	26	PR.DBGNM - Debugger name			
74	29				
78	30	Reserved			
88	34				
8C	35	PR.LAS	Reserved		
90	36	PR.REGD - Reserved for Linker			
94	37	PR.OIT - Reserved for Linker			
98	38	PR.SHIMG - File address of shared image descriptors			
9C	39	Reserved			
A0	40	PR.TIME - Time image linked			
	41				
A8	42	PR.DATEB - Date image linked			
	43				
B0	44	Reserved to end of sector			
	n				

Byte (Hex)	Name	
0	PR.LVER	Linker version number; Field length = 1W; Indicates which hash mechanism is used for hash accesses and the size of the global symbol table entries for use by the Symbolic Debugger/X32.
4	PR.IVER	Image version number; Field length = 1W; Task version number of the nonshared image.
8	PR.ROSIZ	Bytes in read only section; Field length = 1W; Number of bytes in the read only section.
C	PR.RWSIZ	Bytes in read/write section; Field length = 1W; Number of bytes in the read/write section.
10	PR.MNT	Dynamic mount count; Field length = 1B; Number of nonpublic volumes required for dynamic mounts; Default is zero.
	PR.FLAG2	Second flag; Field length = 1B;

Bit	Meaning if Set
0	Debugger symbol support not present (PR2.NOSY)
1	Debugger speed support present (PR2.DBSS)
2	Task is an Ada task (PR2.ADA)
3	Reserved for shared image use by owner (PR2.SHBO)
4	Special arithmetic function (PR2.AF)

	PR.NSI	Number of shared images included; Field length = 1H.
14	PR.TRAN	Transfer address; Field length = 1W.
18	PR.CNTL	Control; Field length = 1B;

Bit	Meaning if Set	Meaning if Unset
0	Load module file status is segmented	Load module status is continuous
1	Link operation status (PR.LOS) is executable	Link operation (PR.LOS) is nonexecutable
2	Loader (PR.NOCKS) is no checksum	Loader (PR.NOCKS) is checksum
3	Task status (PR.PRIV) is privileged	Task status (PR.PRIV) is unprivileged
4	System Administrator (PR.SAM) attribute is on	System Administrator (PR.SAM) attribute is off
5	Overlay status is overlaid	Overlay status is nonoverlaid

Bit	Meaning if Set	Meaning if Unset
6	Image status is shared	Image status is nonshared
7	Overwrite status is verified	Overwrite status is not verified

PR.FLAG Flags;
Field length = 1B;

Bit	Meaning if Set	Meaning if Unset
0	Address mode (PR.ABSC) is absolute	Address mode (PR.ANSC) is relocatable
1	Residency (PR.RES) is unswappable	Residency (PR.RES) is swappable
2	Read only section (PR.SHR) is shared	Read only section (PR.SHR) is unshared
3	Prelocation status (PR.PREL) is prelocated	Prelocation status (PR.PREL) is not located
4	Debugger (PR.NODBG) is not allowed. See Note.	Debugger (PR.NODBG) is allowed
5	Task configuration (PR.MULTI) is multicopied	Task configuration(PR.MULTI) is unique
6	Base mode executable image PR.BASE is always set	
7	PR.MPX20 always set	

Note: If bit 4 is set, the file and buffer assignment totals in PR.FILE and PR.BUFR are incremented for debugger support.

PR.NNRS Resource requirement summary entry;
Field length = 1B;
Number of entries in the resource requirement summary table.

PR.BPRI Execution priority;
Field length = 1B;
Base execution priority.

1C PR.NOOL Overlay;
Field length = 1H;
Number of overlays;
Should be zeroed.

PR.IPRI I/O priority;
Field length = 1B;
Base I/O priority level;
Default is equal to PR.BPRI.

PR.RRSSZ Resource requirement summary size;
Field length = 1B;
Size in blocks of the resource requirement summary table.

20 PR.RRS Resource requirement summary address;
Field length = 1B;
File address of the resource requirement summary table.

PR.FILE Files;
Field length = 1B;
Total number of files that can be concurrently open during task execution. This number is the sum of the file allocations specified in the LINKER/X32 FILES directive plus the number of files specified in any shared images included in the task.

PR.BUFR Buffers;
Field length = 1B;
Total number of blocking buffers required by the task. This number is the sum of the buffers specified in the LINKER/X32 BUFFERS directive plus the number of buffers specified in any shared images included in the task.

PR.SEGS Segment definition;
Field length = 1B;
Segment definition area count. This number is the sum of the segmented files specified in the LINKER/X32 SEGFILES directive plus the number of files specified in any shared images included in the task.

24 PR.OPTN Program option word;
Field length = 1W.

28 PR.RESVD Reserved;
Field length = 1B.

PR.SSIZ Stack size;
Field length = 3B;
Program stack size in bytes. This number is the sum of the STACKSIZE specified in the LINKER/X32 STACKSIZE directive plus the number of files specified in any shared images included in the task.

2C PR.OLD Overlay descriptors;
Field length = 1W;
Should be zeroed.

30 PR.GST Global symbol table;
Field length = 1W;
File address of the Global Symbol Table.

34 PR.DBG Debugger information;
Field length = 1W;
File address of debugger information.

38 PR.ROSD Read only image section descriptor;
Field length = 6W;

Word	Definition
0	Load address (RO.LADD)
1	Image section file address (RO.FADD)
2	Image section length (in bytes) (RO.SIZE)
3	Image section checksum (RO.CKSM)
4	Relocation list file address (RO.RLAD)
5	Number of relocation entries (RP.NRE)

50	PR.RWSD	Read/write image section descriptor; Field length = 6W; Word Definition 0 Load address (RW.LADD) 1 Image section file address (RW.FADD) 2 Image section length (in bytes) (RW.SIZE) 3 Image section checksum (RW.CKSM) 4 Relocation list file address (RW.RLAD) 5 Number of relocation entries (RW.NRE)
68	PR.DBGNM	Debugger name; Field length = 4W; Debugger name to be used if different from the default debugger name.
78-88	Reserved	
8C	PR.LAS	Logical address size; Field length = 1H; The size in map blocks of the logical address space.
	Reserved	Field length = 1H.
90	PR.REGD	Region descriptor; Field length = 1W; Should be zeroed.
94	PR.OIT	Overlay information table; Field length = 1W; Should be zeroed.
98	PR.SHIMG	Shared image descriptors; Field length = 1W; File address of shared image descriptors.
9C	Reserved	Field length = 1W.
A0	PR.TIME	Time image linked or relinked; Field length = 2W.
A8	PR.DATEB	Date image linked or relinked; Field length = 2W.
B0	Reserved	Field length = to end of sector.

2.42.6 Shared Executable Image Structure

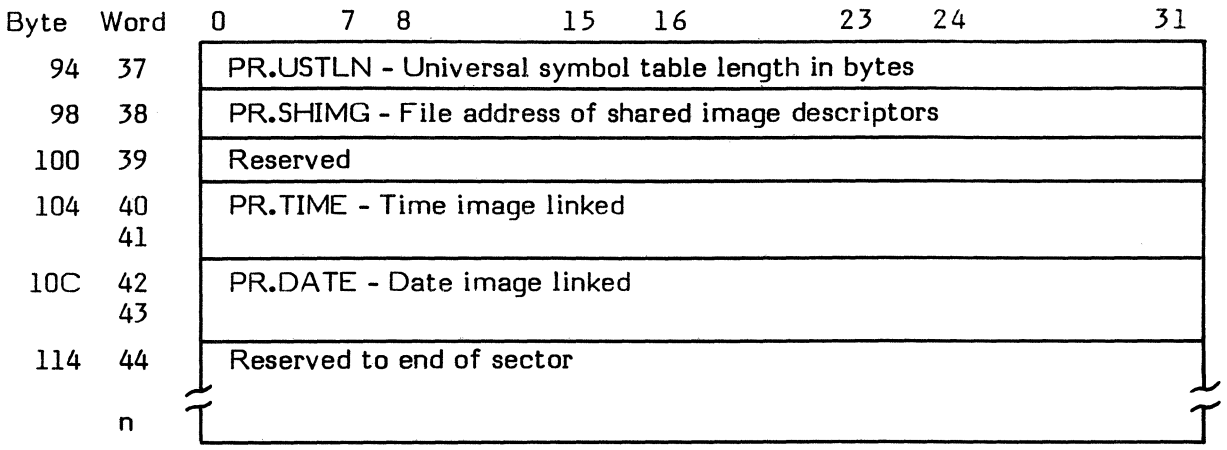
An executable image is a permanent file of base mode object code that was built by the Linker/X32. The following is the shared executable image structure.

Shared Image Preamble
Shared Image Descriptors
Read Only Image Section
Read/Write Image Section
Read/Write Writeback Image Section
Universal Symbol Table
Global Common Program Image Section Information

2.42.7 Shared Executable Image Preamble

The shared image preamble contains image information in the following format:

Byte	Word	0 7 8 15 16 23 24 31
0	0	PR.LVER - Linker version number
4	1	PR.IVER - Shared image version number
8	2	PR.COMP - Compatibility level
0C	3	PR.PHADS - Physical load address
10	4	PR.MNT | PR.FLAG2 | PR.NSI
14	5	PR.TRAN - Should be zeroed
18	6	PR.CNTL | PR.FLAG | Reserved for Linker
1C	7	Reserved for Linker
20	8	Reserved | PR.FILE | PR.BUFR | PR.SEGS
24	9	Reserved for Linker
28	10	PR.SSIZE - Stack size in bytes (first byte reserved)
2C	11	PR.GCMFA - File address of global common definitions
30	12	PR.GCMNE - Number of global common definitions
34	13	PR.DBG - File address of debugger information
38	14	PR.ROSD - Read only image section descriptor
4C	19	
50	20	PR.RWSD - Read/write image section descriptor
64	25	
68	26	PR.RWWB - Read/write writeback image section descriptor
7C	31	
80	32	PR.UST - File address of universal symbol table
84	33	Reserved
88	34	Reserved
8C	35	PR.USTSO - Sector offset for universal symbol table
90	36	PR.USHLN - Universal symbol hash table length in bytes



Some of the above reserved areas are used by the Linker/X32. They are not used by the MPX-32 operating system.

Byte (Hex)	Name	
0	PR.LVER	Linker version number; Field length = 1W; Indicates that hash mechanism is used for hash accesses and the size of the global symbol table entries for use by the Symbol Debugger/X32.
4	PR.IVER	Shared image version number; Field length = 1W; Task version number of the shared image.
8	PR.COMP	Compatibility level; Field length = 1W; The lowest version number that is compatible with this copy of the shared image.
0C	PR.PHADS	Physical load address; Field length = 1W; The physical address where the shared image is loaded.
10	PR.MNT	Dynamic mount count; Field length = 1B; Number of nonpublic volumes required for dynamic mounts; Default is zero.
	PR.FLAG2	Second flag; Field length = 1B;

Bit	Meaning if Set
0	Debugger symbol support not present (PR2.NOSY)
1	Debugger speed support present (PR2.DBSS)
2	Task is an ADA task (PR2.ADA)
3	Share image by owner (PR2.SHBO)
4	Reserved for executable image use

	PR.NSI	Number of shared images included; Field length = 1H.

Byte (Hex)	Name	
14	PR.TRAN	Should be zeroed; Field length = 1W.
18	PR.CNTL	Control; Field length = 1B;

Bit	Meaning if Set	Meaning if Unset
0	Load module file status is segmented	Load module status is continuous
1	Link operation status (PR.LOS) is executable	Link operation status (PR.LOS) is nonexecutable
2	Loader (PR.NOCKS) is no checksum	Loader (PR.NOCKS) is checksum
3	Task status (PR.PRIV) privileged	Task status (PR.PRIV) is unprivileged
4	System Administrator (PR.SAM) attribute is on	System Administrator (PR.SAM) attribute is off
5	Overlay status is overlaid	Overlay status is nonoverlaid
6	Image status is shared	Image status is nonshared
7	Overwrite status is verified	Overwrite status is not verified

PR.FLAG Flags;
Field length = 1B;

Bit	Meaning if Set	Meaning if Unset
0	Address mode (PR.ABSC) is absolute	Address mode (PR.ABSC) is relocatable
1	Residency (PR.RES) is unswappable	Residency (PR.RES) is swappable
2	Read only section (PR.SHR) is shared	Read only section (PR.SHR) is unshared
3	Prelocation status (PR.PREL) is prelocated	Prelocation status (PR.PREL) is not prelocated
4	Debugger (PR.NODBG) is not allowed. See Note.	Debugger (PR.NODBG) is allowed
5	Task configuration (PR.MULTI) is multicopied	Task configuration (PR.MULTI) is unique
6	Base register load module (PR.BASE) is always set	
7	PR.MPX20 is always set	

Note: If bit 4 is set, the file and buffer assignment totals in PRE.FLE and PR.BUFR are incremented for debugger support.

	Reserved	Field length = 1H.
1C	Reserved	Field length = 1W.
20	Reserved	Field length = 1B.
	PR.FILE	Files; Field length = 1B; Number of dynamic file allocations specified in the Linker/X32 FILES directive.
	PR.BUFR	Buffers; Field length = 1B; Number of dynamic blocked file allocations specified in the Linker/X32 BUFFERS directive.
	PR.SEGS	Segment definition; Field length = 1B; Segment definition area counts specified in LINKER/X32 SEGFILES directive.
24	Reserved	Field length = 1W.
28	PR.SSIZE	Reserved; Field length = 1B.
		Stack size; Field length = 3B; Program stack size in bytes specified in LINKER/X32 STACKSIZE directive.
2C	PR.GCMFA	File address of global common definitions; Field length = 1W.
30	PR.GCMNE	Number of global common definitions; Field length = 1W.
34	PR.DBG	Debugger information; Field length = 1W; Should be zeroed.
38	PR.ROSD	Real only image section descriptor; Field length = 6W;

Word	Definition
0	Load address (RO.LADD)
1	Image section file address (RO.FADD)
2	Image section length in bytes (RO.SIZE)
3	Image section checksum (RO.CKSM)
4	Relocation list file address (RO.RLAD)
5	Number of relocation entries (RO.NRE)

50 PR.RWSD Read/write image section descriptor;
Field length = 6W;

Word	Definition
0	Load address (RW.LADD)
1	Image section file addres (RW.FADD)
2	Image section length in bytes (RW.SIZE)
3	Image section checksum (RW.CKSM)
4	Relocation list file address (RW.RLAD)
5	Number of relocation entries (RW.NRE)

68 PR.RWWB Read/write image section descriptor for writeback;
Field length = 6W;

Word	Definition
0	Load address (WB.LADD)
1	Image section file address (WB.FADD)
2	Image section length in bytes (WB.SIZE)
3	Image section checksum (WB.CKSM)
4	Relocation list file address (WB.RLAD)
5	Number of relocation entries (WB.NRE)

80 PR.UST Universal symbol table;
Field length = 1W;
File address of the universal symbol table.

84 Reserved Field length = 1W.

88 Reserved Field length = 1W.

8C PR.USTSO Universal symbol table offset;
Field length = 1W;
Sector offset for the universal symbol table.

90 PR.USHLN Universal symbol hash table;
Field length = 1W;
Length in bytes of the universal symbol hash table.

94 PR.USTLN Universal symbol table;
Field length = 1W;
Length in bytes of the universal symbol table.

98 PR.SHIMG File Address of shared image descriptors;
Field length = 1W.

100 Reserved Field length = 1W.

104 PR.TIME Time image linked or relinked;
Field length = 2W.

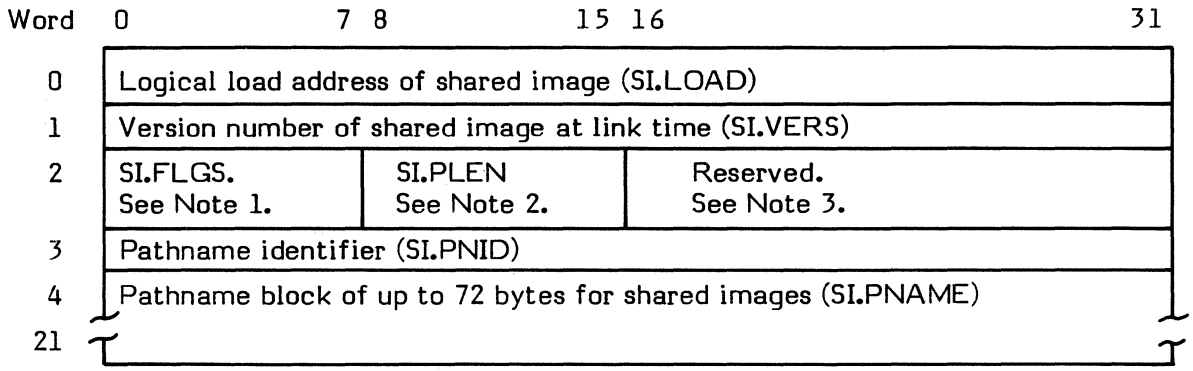
10C PR.DATE Date image linked or relinked;
Field length = 2W.

114 Reserved Field length = to end of sector.

2.42.8 Shared Image Descriptors

Every shared image has a shared image descriptor that is doubleword bounded. Shared image descriptors are used by the loader to verify that a task can access the required shared images.

Preassigned shared images are loaded at task activation. Other shared images are loaded by the task through run-time shared image calls.



Note:

1. SI.FLGS Bits in SI.FLGS are assigned as follows:

Bits	Meaning if Set
0	Preassigned shared image
1	Read/write access requested (SI.RWAC)
2	Modified read/write image section request
3	Position dependent shared image (SI.PDEP)
4	Writeback mode requested (SI.WRBK)
5	Writeback section present
6-7	Reserved

2. Byte one is the shared image pathname length (SI.PLEN).

3. Byte two is reserved for LINKX32 (should be zeroed). Byte three is reserved for LINKX32-inclusion level of the shared image.

CHAPTER 3

SYSTEM TASK DESCRIPTIONS

3.1 Non Operating System Resident Swap Scheduler Task (J.SWAPR)

The swap scheduler (J.SWAPR) is a nonswappable memory management task. J.SWAPR provides memory allocation for tasks that require memory, and services memory requests when memory is not available. J.SWAPR is not embedded in the operating system, and does not occupy logical address space. J.SWAPR is not mapped into the address space of every task in the system. J.SWAPR is a memory resident, privileged task. See Figure 3-1.

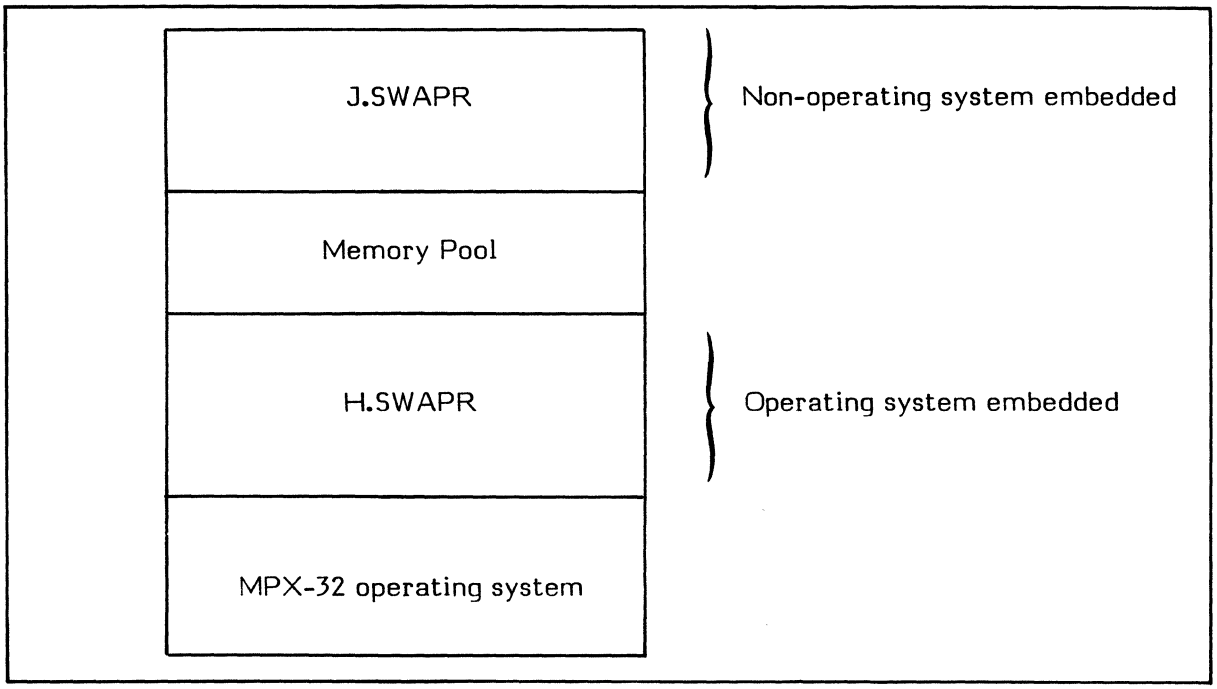


Figure 3-1. System Swap Scheduler

J.SWAPR is activated by the system initializer task (SYSINIT). Once activated, J.SWAPR remains in a suspended state until it is needed. When resumed, J.SWAPR executes at the priority of the highest priority task in the memory request queue (MRQ). When all swap activity is completed, J.SWAPR returns to the suspended state. J.SWAPR remains suspended until it is resumed by H.EXEC,9 in response to a memory scheduler event.

To free memory, J.SWAPR selects a task for outswap and proceeds to write all or parts of the task to a secondary storage area called the swap file. When memory becomes available, outswapped tasks that are requesting memory are inswapped in priority order, and are allowed to execute normally.

J.SWAPR uses two sets of MIDL and MEML arrays for task map manipulation. J.SWAPR's TSA MIDL/MEML arrays are used to map the candidate task's TSA on top of J.SWAPR. See Figure 3-2. If the task is to be outswapped, the MIDL entries for maps to be rolled out are copied to J.SWAPR's working MIDL. See Figure 3-3. J.SWAPR then calls internal subroutine S.SWAP1A to remap to the working MIDL. This gives J.SWAPR addressability to the outswap candidate's memory in order to write the maps to the swap file.

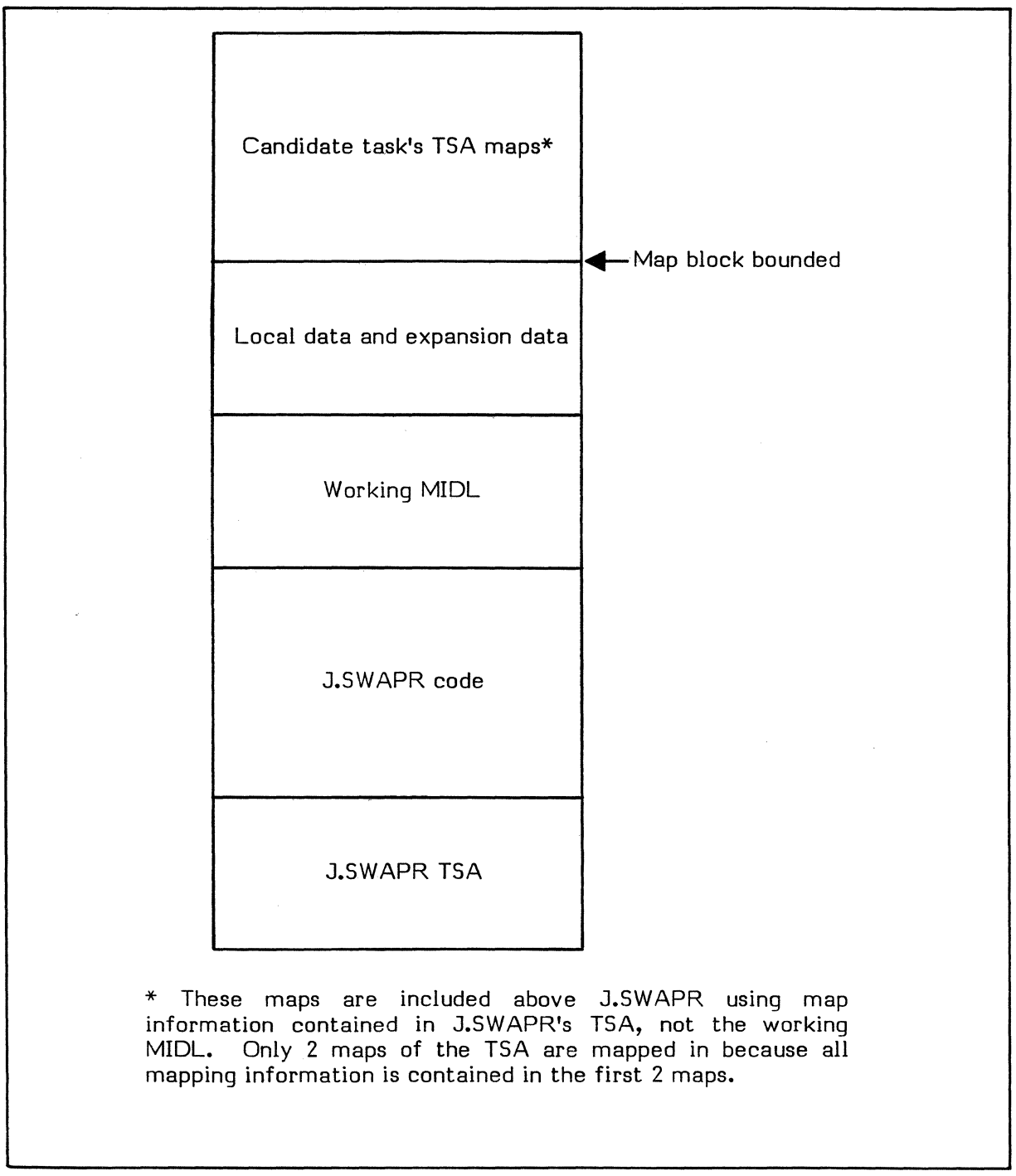


Figure 3-2. Mapping of Candidate TASK's TSA (an overview)

J.SWAPR has an operating system resident counterpart called H.SWAPR. H.SWAPR performs MIDL entry copying, and holds data structures that are required by other areas of the operating system. This ensures that the system trace and system debugger function normally. Base R in the system debugger references H.SWAPR, not J.SWAPR.

J.SWAPR also provides for selective partial outswapping. Partial outswapping allows a small portion of a large task to be outswapped when only a small portion of memory is required. Partial outswapping saves the time it would take to outswap and eventually inswap the unneeded memory.

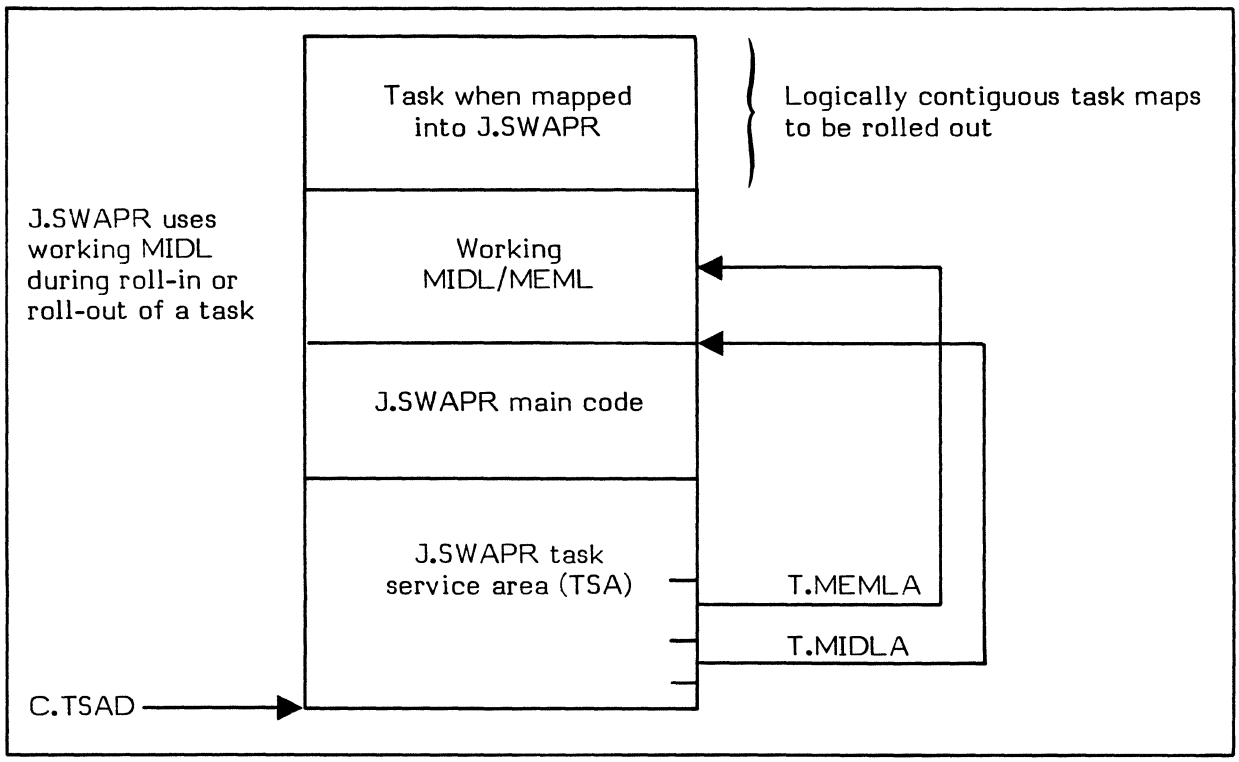


Figure 3-3. Mapping of a Candidate Task During Roll-out

The SYSGEN SWAPLIM directive allows the user to specify a minimum partial outswap quantum. When heavy swapping is anticipated, a value between 7 and 15 is recommended. A SWAPLIM within this range reduces the number of swaps necessary to obtain the required memory. A SWAPLIM value of 16 or greater can reduce the effectiveness of partial swapping.

When memory is requested by a task or tasks, and there is insufficient memory to satisfy the request, J.SWAPR is resumed by using H.EXEC,37. J.SWAPR allocates and deallocates memory for the memory requests listed in Section 3.1.3 and 3.1.4.

The following illustrates J.SWAPR's task layout.

Remap area: used to map task's TSA or memory that is being swapped
J.SWAPR swap file entries pool (Q-entries): expands as the system runs to meet the system's requirements. See Note 1.
Dedicated system buffer space
J.SWAPR's shadow tables: used to allocate shadow memory. See Note 2.
J.SWAPR's outswap shadow tables: one for each DQE. See Note 2.
J.SWAPR's working MIDL/MEML arrays (WORKMIDL and WORKMEML): used for managing memory that is being swapped. See Note 3.
Swap context area: one entry for each DQE
J.SWAPR's DSECL
J.SWAPR's TSA (SWPMIDL and SWPMEML)
Memory pool
H.SWAPR
MPX-32

Notes:

1. Q-entries are also built from the initialization code after it is executed.

2. Shadow tables are present only if shadow memory is configured.

3. WORKMIDL and WORKMEML point to the mapping information for J.SWAPR while MIDLA and MEMLA point to the mapping information for the logical address space called the remap area.

3.1.1 J.SWAPR Processing

J.SWAPR can perform the following types of processing:

1) Dispatch processing
2) Inswap processing
3) Shared memory request processing
4) No memory available processing
5) Outswap processing
6) I/O error handling processing
7) Initialization processing

When sufficient memory is unavailable, inswap and outswap are both serial processes, and are completed before the MRQ is reexamined. When sufficient memory is available to load an outswapped task into memory from the swap volume, the inswap task is initiated. The difference between memory requests and inswap requests is that there is no associated disc file to read. Tasks linked to the MRQ can be queued for both expansion and inswap.

1) Dispatch Processing

Dispatch processing begins when swap activity is resumed. Dispatch processing performs the following functions:

- Calls the inswap routine if the task is outswapped
- Allocates memory as required by calling H.MEMM,1
- Outswaps a task if no memory is available
- Links the task's DQE to the ready queue
- Suspends itself if the MRQ (Memory Request Queue) is empty
- Redispatches if entries still exist in the MRQ
- Calls H.EXEC,7 to report memory requests event complete
- Changes J.SWAPR's priority to that of the requesting task.

The dispatcher examines the memory request queue (MRQ). If entries are not present, J.SWAPR suspends until another memory request event occurs. If entries are present, dispatch processes memory requests, and counts unprocessed memory (MRQ.CNT) and memory release events (C.RRUN). J.SWAPR attempts to allocate memory to tasks waiting for memory.

2) Inswap Processing

When sufficient memory is available, J.SWAPR allocates the memory to the highest priority task on the MRQ. If the request is for inswap, J.SWAPR reads the swapped image into the newly allocated memory. This process is completed in two passes.

On the first pass, the outswapped TSA and DSECT are read off the swap space, and the swap space entries are released.

On the second pass, the outswapped shared regions are interrogated to find out if they are outswapped. If they are outswapped, memory must be allocated before they can be loaded into the task's TSA tables. If sufficient memory cannot be allocated, the task is put on the MRQ requesting dynamic expansion to include a shared region. After the shared memory is inswapped, the used swap space entries are released if the shared memory is not a CSECT.

3) Shared Memory Request (SISHR) Processing

Shared memory request (SISHR) process begins when there is currently outswapped shared memory that needs to be inswapped, or when a specified memory partition needs to be included into the address space of a task.

4) No Memory Available (NOMEM) Processing

No memory available (NOMEM) processing begins when there is insufficient memory to fulfill the memory requestor's requirements. NOMEM determines if enough memory can be freed by outswapping. If so, NOMEM calls the outswap process to free the required memory, then calls the inswap or SISHR process to allow the memory requestor to get the free memory.

5) Outswap Processing

Outswap processing occurs when the outswapping of a particular task or tasks will free sufficient memory for the memory requestor.

In order to complete the outswap process, the TSA of the outswap candidate is mapped into J.SWAPR. The protocol of outswap requires two (2) complete passes of the user's TSA tables (MIDL and MEML).

On the first pass, only nonshared maps are logically built into J.SWAPR's work are in their original TSA format. A second swap space is allocated and the outswap I/O process is performed. A used swap space entry is then added to DQE.SRID.

On the second pass, all swappable regions (if any) are built into a work area inside J.SWAPR in logically contiguous format. A swap space is allocated and the outswap I/O process is performed. A used swap entry is then added to SMT.SRID. If there are no shared regions, I/O is not performed in this pass.

At the end of each pass, memory is returned to the free list. When both passes of the outswap are complete, the MRQ is reexamined to find the highest priority candidate to receive the memory from the free list.

Outswap candidates are first chosen from the wait states, then from the run state. The default order is shown below:

3.1.2 Selection of Outswap Candidates

The swapper initially attempts to process the memory requirement for the highest priority task on the MRQ. If insufficient memory is available, the swapper examines the state queues on a priority basis searching for the memory class and number of map blocks required by the requesting task. The first task found with resources satisfying any of the requirements of the requester is partially or totally outswapped. When the outswap process is complete, the swapper re-examines the MRQ and continues to process memory requests.

Outswap candidates are first chosen from wait states and then from run states. The order is shown below:

```
NWS       DATAW   15          NUMBER OF WAIT STATES
WAITSTAT  DATAW   C.HOLD      WAITSTATE Q POINTERS IN OUTSWAP ORDER
          DATAW   C.SUSP
          DATAW   C.RUNW
          DATAW   C.SWDV
          DATAW   C.SWDC
          DATAW   C.SWSR
          DATAW   C.SWSM
          DATAW   C.SWLO
          DATAW   C.SWFI
          DATAW   C.MRQ
          DATAW   C.ANYW
          DATAW   C.SWGQ
          DATAW   C.SWTI
          DATAW   C.SWIO
          DATAW   C.SWMP
NRS       DATAW   12          NUMBER OF RUN STATE Q ENTRIES
RUNSTAT   DATAW   C.SQ64      RUN STATE Q POINTERS IN OUTSWAP ORDER
          DATAW   C.SQ63
          DATAW   C.SQ62
          DATAW   C.SQ61
          DATAW   C.SQ60
          DATAW   C.SQ59
          DATAW   C.SQ58
          DATAW   C.SQ57
          DATAW   C.SQ56
          DATAW   C.SQ55
          DATAW   C.RIPU
          DATAW   C.SQRT
```

Once an outswap candidate is selected by J.SWAPR, DQE.SOPO (swap-on priority only) and DQE.SWIF are examined. Both locations are one byte variables containing several swap limitations.

Swap inhibit conditions are checked first. If any DQE.SWIF bit other than DQE.TLVS (task leaving system) is set, the task is unswappable. If DQE.SOPO bits DQE.BMAP or DQE.MDTA are set, the task is unswappable. J.SWAPR then searches for a new outswap candidate.

DQE.SOPO is examined only if DQE.SWIF equals zero or if DQE.TLVS is set. If the DQE.USPO bit in DQE.SOPO is set, the priority of the memory requestor is compared to the outswap task priority. If the user settable swap-on priority only feature is enabled, and if the DQE.USPO bit in DQE.SOPO is set, the priority of the memory requestor is compared to the outswap task priority. J.SWAPR searches for another outswap candidate if the memory requestor does not have a higher priority if the outswap task is ready to run, if the memory requestor does not have a higher or equal priority or if the outswap task is in a wait state. If DQE.SOPO equals zero, the outswap candidate is outswapped.

6) I/O Error Handling Processing

The I/O error handling portion of J.SWAPRs code helps the system recover from I/O errors. If a write error is encountered during the outswap of a task, J.SWAPR assumes that the error is a bad block on the disc. J.SWAPR discards the current Q-entry that is defining the bad swap space, allocates another Q-entry for new swap space, and retries the write request. If succeeding writes fail and the end of the swap file is reached, J.SWAPR releases the bad Q-entries and suspends for ten seconds. After the suspension, J.SWAPR retries the write. J.SWAPR repeats the process until the write is successfully completed.

On a read error during the inswap of an outswapped task, J.SWAPR places the inswap candidate in the hold state, outputs a message to the console, and continues the swapping activities. In order to resume the task in the hold state, the operator must issue the OPCOM CONTINUE directive for that task.

7) Initialization Processing

The system initialization program (SYSINIT) creates a swap file and activates J.SWAPR. J.SWAPR then activiates its secondary initialization routine S.SWAP99. S.SWAP99 then does the following:

- Initializes the swap activity table
- Allocates memory for Q-entries dynamically
- Allocates the working MIDL/MEML dynamically
- Allocates the shadow memory tables dynamically, if applicable
- Allocates the swap files space dynamically
- Assigns and opens the swap file
- Sets the MIDL/MEML pointers for use by the remap rouitnes
- Computes the location where task's TSA will reside in J.SWAPR
- Sets up swap inhibit and swap-on priority only (SOPO) masks
- Builds Q-entries from J.SWAPR's secondary initialization code space

After the initialization code is processed, its space is used as a buffer area for I/O operations or as an extension of linked list structures. This ensures a compact J.SWAPR.

3.1.3 J.SWAPR Internal Subroutines

The following are internal subroutines of J.SWAPR:

Subroutine	Description
S.SWAP1	Remaps task into J.SWAPR not using the working array
S.SWAP1A	Remaps J.SWAPR using the working MIDL/MEML array
S.SWAP2	Updates the SMT use count when a task using shared image is rolled back into memory. S.SWAP2 also updates tasks MIDL/MEML to correctly reference the shared image.
S.SWAP3	Determines the outswap candidate. S.SWAP3 also checks for sufficient available outswap maps. For example, do the requested maps equal the maps available if the candidate is outswapped.
S.SWAP4	Reads in a task from the swap file with the initial 128KB read. It is followed by a single disc sector read until the task is restored to memory. Memory roll-in is a random access operation.
S.SWAP5	Writes a task to the swap file with the initial 128KB write. It is followed by single disc sector writes until the task is rolled out of memory. Memory roll-out is a random access operation.
S.SWAP6	Allocates the swap space on the swap file; if none is available, it aborts the task.
S.SWAP7	Releases swap spaces. Where applicable, it will coalesce two small free swap spaces into one large free swap space. This is a garbage collection routine.
S.SWAP8	Allocates a free space entry
S.SWAP9	Releases a free space entry
S.SWAP10	Builds a linked list of free entries
S.SWAP11	Links an entry to a given queue by priority
S.SWAP12	Unlinks an entry from a given queue
S.SWAP13	Increases the priority of a target task. The target tasks DQE address and the amount of increase are supplied by the caller.
S.SWAP14	Stores information needed to inswap a task requesting shadow memory
S.SWAP15	Is a swap file extension subroutine

S.SWAP16 Adds an entry into the shared memory include list (SMIL)

S.SWAP17 Removes an entry from the shared memory include list (SMIL)

S.SWAP18 Allocates a map block for more queue entry space

S.SWAP19 Determines the total number of map blocks needed to inswap an outswapped task

S.SWAP20 Counts the number of swappable map blocks belonging to an outswap candidate

S.SWAP21 Determines if an inswap of a task requiring multiple memory types is possible

S.SWAP22 Determines how many map blocks of a shared memory partition to outswap

S.SWAP23 Determines if a task is swap-on priority only (SOPO)

S.SWAP24 Is the thrash control subroutine

S.SWAP99 Performs J.SWAPR self initialization. S.SWAP99 is executed once when SYSINIT activates J.SWAPR. The space occupied is then reclaimed for swap space entries by S.SWAP10 if the initial allocation of swap entries is used up.

3.1.4 J.SWAPR Memory Request Functions

Memory request functions performed by J.SWAPR are:

1) Memory expansion request
2) Memory deallocation request
3) Inswap request
4) Change in task status request
5) Shared memory include request
6) Exit conditions

See the following sections for descriptions of the memory request functions.

There are memory request function codes for J.SWAPR. These function codes determine the action that J.SWAPR performs for a requesting task. See Table 3-1.

1) Memory Expansion Request

A memory exapnsion request occurs when there is insufficient memory to satisfy a task's dynamic memory request. The task is linked to the memory request queue (MRQ) with the number of maps needed, the type of memory needed, the address where the memory is loaded, and the expansion request code. J.SWAPR is then resumed to process the request.

2) Memory Deallocation Request

When a task deallocates all or some of its memory and there are tasks linked to the MRQ, J.SWAPR is resumed so it can reallocate the memory to a task in the MRQ.

3) Inswap Request (Memory Roll-in)

When a currently outswapped task is ready for execution, J.SWAPR is resumed. The task is located in the MRQ, and the inswap request is processed.

4) Change in Task Status Request

When a task which has previously been ineligible for swapping due to unbuffered I/O in progress, release of a lock in memory flag, expiration of a stage 1 time quantum, etc., J.SWAPR is resumed.

5) Shared Memory Include Request

When there is sufficient memory to satisfy the inclusion of a task's dynamic shared memory partitions, J.SWAPR, having been resumed prior to receiving the request, links the tasks to the MRQ to process the request.

6) Exit Conditions

When J.SWAPR finds the MRQ empty, or when there are no outstanding requests to process, J.SWAPR suspends itself by unlinking from the ready-to-run queue and relinking to the wait-for-memory-event queue. This is accomplished by using H.EXEC,8.

Table 3-1
Memory Request Function Codes for J.SWAPR

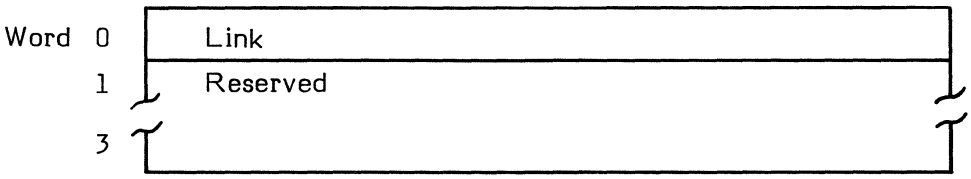
Function Code	Description
0	Inswap request
1	Pre-activation request
2	Task activation request
3	Memory expansion request
4	IOCS buffer request
5	Shared memory include request
6	System buffer space request
7	Release swap space request
8	Exclude shared memory request
9	Reserved

3.1.5 Managing Swap Space Entries

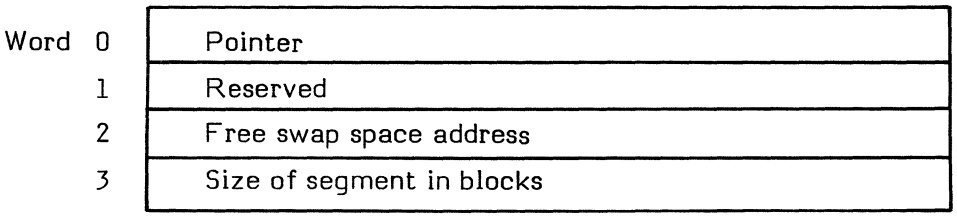
J.SWAPR manages swap space entries by using four one-way linked entry lists:

List	Description
H.ENTRY	Points to free queue entry lists
Q.SWP	Lists all free queue entries
DQE.SRID	Contains queue entries for DSECT memory
SMT.SRID	Contains queue entries for outswapped shared memory

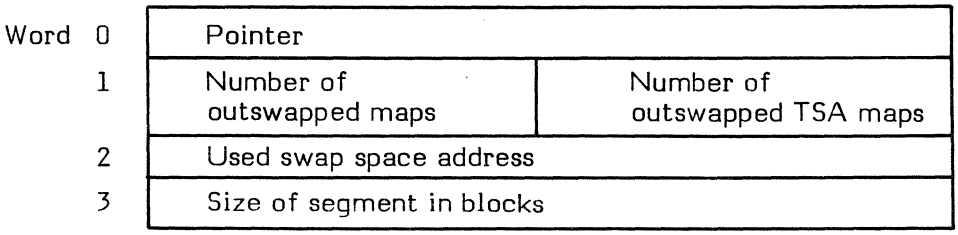
H.ENTRY is an internal linked headcell list that monitors the free swap file entries. H.ENTRY is located in H.SWAPR. The first word is the link, and the last three words are reserved. The addresses of the swap file free entries are maintained by Q.SWP.



Q.SWP is an internal linked list head cell that maintains the addresses of all swap file free entries. Q.SWP has a four word entry for each linked list entry. The swap space entry, Q.SWP, is as follows:



DQE.SRID is a two-word linked list headcell. DQE.SRID functions as a queue and is located in the task's DQE. Each swap space entry is four words long. The first word points to the first used swap space entry. The second word points to the last entry. All entries except the headcell are located internal to J.SWAPR. When a task's nonshared memory is outswapped, the swap space is allocated and DQE.SRID is updated. When the memory is inswapped, the swap space is freed, and DQE.SRID is emptied. The swap space entry (headcell = DQE.SRID) is as follows:



SMT.SRID is a linked list headcell located in the SMT. When memory is a CSECT, the swap space is not released, and the SMT.SRID remains the same as before the inswap. SMT.SRID empties when the task exists and the user count is zero. A SMT.SRID entry has the same format as a DQE.SRID except the right half of the second word is always zero. The swap space entry (headcell = DMT.SRID) is as follows:

Word	0	Pointer	
	1	Number of outswapped maps	Zero
	2	Used swap space address	
	3	Size of segment in blocks	

3.1.6 Swap Context Area

The swap context area (SCA) is a table of fixed size entries. There is one entry for each DQE in the system. The SCA entries are indexed by the DQE index number which is the first byte of the task activation number (DQE.TAN).

		0	16	31
Word	0	SCA.TAN. See Note 1.		
	1	SCA.SMIL. See Note 2.		
	2	SCA.MIDL. See Note 3.		
	3	SCA.MAPN. See Note 4.	SCA.FLGS. See Note 5.	

Notes:

1. SCA.TAN contains the task activation number most recently associated with the entry.

2. SCA.SMIL contains the address of the first Shared Memory Include (SMI) Q-entry of the Shared Memory Include List (SMIL).

3. SCA.MIDL can contain the beginning address with J.SWAPR's address space where the last partial outswap MIDL scan started

4. SCA.MAPN is used with SCA.MIDL. SCA.MAPN is the negative number of MIDs left to scan when outswapping a partially swapped task.

5. SCA.FLGS indicates the swap state of a task. The bit setiing are as follows:

Bit	Meaning if Set
0	DSECT is partially outswapped
1	DSECT is completely outswapped
2-15	Reserved

3.1.7 Swap Activity Table

The swap activity table is maintained and updated by the J.SWAPR subroutine S.SWAP24. This subroutine determines how active J.SWAPR is and sets the global swap-on priority only (SOPO) flag if the actual swap activity exceeds the desired maximum swap activity. S.SWAP24 is called at the end of an inswap, at the end of an outswap, and just prior to searching for an outswap candidate.

3.1.8 Shadow Memory Outswap Tables

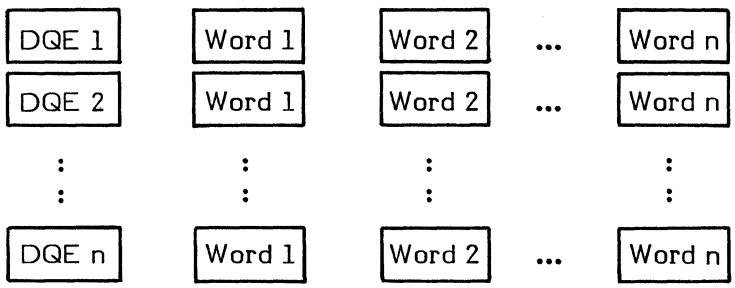
J.SWAPR initializes outswap shadow tables within its logical address space. If needed, J.SWAPR dynamically allocates more memory for these tables. The number of words of memory needed for the outswap tables is equal to the number of dispatch queue entries in the system times the sum of the greater number of physical map blocks shadowed by either processor 1 or processor 2 plus one. For example:

1. 48 dispatch queue entries
2. One 128KB shadow memory board on processor 1 (16 map blocks)
3. Two 128 KB shadow memory board on processor 2 (32 map blocks)

Outswap shadow tables needed equals 48 X (32 + 1) words.

J.SWAPR uses the outswap shadow memory tables to remember which map blocks an outswapped task may need to be shadowed when inswapped.

Outswap Shadow Table



*The bits in each word have the following meaning:

Bit	Label	Meaning if Set
0	SH.REQST	Shadow memory requested
1	SH.REQRD	Shadow memory required
2	SH.IPU	IPU shadow memory
3-31	N/A	Contains the MIDL of the map image descriptor. This points to the map block number to be shadowed.

3.2 Terminal Service Manager Task (J.TSM)

3.2.1 Functional Description

The Terminal Services Manager (J.TSM) is a nonresident privileged system task which provides job control processing in both the batch and interactive environments. Its functions include validating logon requests, processing command files, initiating batch jobs, and activating tasks.

3.2.2 Operational Design

J.TSM is designed to operate in five logical interrupt levels. An activation request at a given level may interrupt processing of any lower level. The context of an interrupted process is saved in the TSA stack to enable resumption after completion of the interrupting level. The following are the five logical interrupt levels in order from lowest to highest:

- Base (Initial task activation level)
- Message
- End Action
- Break
- Abort

3.2.2.1 Base Level

J.TSM is activated at the base level by SYSINIT following a system restart. SYSINIT computes the maximum number of files, segment tables, and mounted volume assignments that J.TSM might need based on communication region variables. These parameters are used to construct J.TSM's TSA via the M.PTSK system service. At J.TSM's initial activation, a one time initialization is performed by J.TSM, at which time the abort, break, and message level entry points are established. The base level then enters a scan of all terminal and batch context searching for any outstanding base level service requests. After all base level services are performed, the base level requests an indefinite suspension. The base level will be removed from the any wait state whenever J.TSM exits from a break, message or end action level. Base level service requests include:

- Context clean-up if device failure
- Logon end action chain initiation
- Task status request
- Message sending
- Batch job initiation

3.2.2.2 Message Level

All messages sent to J.TSM have a message type passed in a byte field of the message. The message type determines the type of process to be performed by J.TSM. If the message level is active, further message requests are queued from the message head cell in J.TSM's DQE. The message types recognized by J.TSM are as follows:

Type	Processing
0	Echo messages to screen
1	Set SYC record
2	Task exit processing
3	Validate owner name
4	Task hold processing
5	Purge account file
6	Validate project group
7	Update key file
8	Update project file
9	Output message on system console

3.2.2.3 End Action Level

End action is the transfer of control upon completion of no-wait I/O to the end action address specified in the FCB. No-wait I/O with end action is used by J.TSM to read/write terminals and batch SYC. Therefore, each terminal session is a chain of end action activations as illustrated below:

```
Base level scan
Write-------------ENTER YOUR OWNERNAME:
Request indefinite suspension
                .
                .
Enter end action from write
Read--------------SYSTEM <CR>
Process owner name entered
Report end action complete
                .
                .
Enter end action from read
Write-------------TSM>
Report end action complete
                .
                .
Enter end action from write
Read--------------SHOW <CR>
Report end action complete
                .
                .
Enter end action from read
Process show command at end action level
Write-------------(show output to TTY)
Report end action complete
                .
                .
                etc.
```

Services performed at the end action level are as follows:

- Terminal I/O
- SYC I/O for interactive and batch
- Interactive command processing
- Interactive and batch command file processing
- TSM and batch accounting
- Abort and error message output

Since the bulk of command processing is performed at the end action level, J.TSM performs only one command at a time. If the end action level is active, further end action requests are queued from the task interrupt head cell in J.TSM's DQE.

3.2.2.4 Break Level

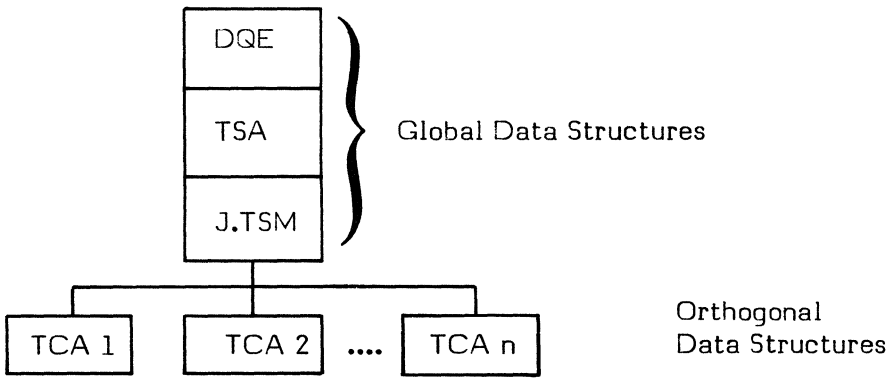
Break level processing sets the break received flag in J.TSM and exits the break level. The break level exit removes J.TSM base level from the any wait state to resume the scan for base level service requests. This level is entered when a user depresses the wake up character (not the break key) on his terminal.

3.2.2.5 Abort Level

J.TSM should never be entered at the abort level. If it is, a fatal system error message is sent to the system console requesting a system reboot. J.TSM then attempts to run by restarting the base level scan.

3.2.3 Data Structures

J.TSM retains a Terminal Context Area (TCA) for each terminal and batch job configured at SYSGEN time. Each end action level activation of J.TSM generally operates on a single TCA. In addition to the TCA, some TSM commands use global data structures such as flags local to J.TSM, the project group in the TSA, or the owner name in the DQE. The global data structures are redefined by J.TSM upon entry to each end action message, and certain base level operations. Processes which depend on these global data structures, but can be logically interrupted by some higher logical activation of J.TSM, must logically block higher requests by setting the synchronous task interrupt bit in J.TSM's DQE. A symbolic diagram of J.TSM's data structures is illustrated below:



3.2.3.1 Terminal Context Area (TCA) Table

Word No. (Decimal)	Byte (Hex)	0 7 8 15 16 23 24 31
000-015	000	TCA.FCB
016-031	040	TCA.EFCB
032-041	080	TCA.PSB and TCA.SPAD
042-047	0A8	TCA.CNP and TCA.SPAD
048-079	0C0	TCA.MPAR
080-081	140	TCA.MS1Z
082-113	148	TCA.CPAR
114-115	1C8	TCA.CS1Z
116-117	1D0	TCA.LABL
118-122	1D8	TCA.MHDR
123-142	1EC	TCA.MBUF
143	23C	TCA.DUMY
144	240	TCA.STS2 | TCA.MNUM | TCA.CNUM
145	244	TCA.ERR
146	248	TCA.LOGN
147	24C	TCA.TERM | RESERVED
148	250	TCA.STAT
149	254	TCA.LBFA
150	258	TCA.UDTA
151	25C	TCA.TNUM
152	260	TCA.JFLG
153	264	TCA.MCEA
154	268	TCA.RRSF
155	26C	TCA.RRSN
156	270	TCA.CPU
157	274	TCA.IPU
158-159	278	TCA.ACES
160-167	280	TCA.SGOS
168-175	2A0	TCA.SLOS
176-177	2C0	TCA.SLOD
178-179	2C8	TCA.NAME
180	2D0	TCA.JOB#
181	2D4	TCA.LFLG
182-189	2D8	TCA.SBOS
190-191	2F8	TCA.SBOD
192-193	300	TCA.NAMB
194	308	TCA.JBB#
195	30C	TCA.BFLG
196-199	310	TCA.CDIR
200-207	320	TCA.CVOL
208-211	340	TCA.MSGB
212-214	350	TCA.ONR1
215-217	35C	TCA.DAT1
218-219	368	TCA.TIM1
220-240	370	TCA.MBX1
241-244	3C4	TCA.HDR2
245-247	3D4	TCA.ONR2
248-250	3E0	TCA.DAT2

Word No. (Decimal)	Byte (Hex)	0 7 8 15 16 23 24 31			
251-252	3EC	TCA.TIM2			
253-273	3F4	TCA.MBX2			
274-287	448	RESERVED			
288	480	TCA.MODE	TCA.NRRS	TCA.ALLO	TCA.MEMC
289	484	TCA.NBUF	TCA.NFIL	TCA.PRIO	TCA.IOER
290-291	488	TCA.LMN			
292-293	490	TCA.SUDO			
294-295	498	TCA.ONRN			
296-297	4A0	TCA.PROJ			
298	4A8	TCA.USRK			
299	4AC	TCA.PGOW			
300	430	TCA.USW			
301-492	4B4	TCA.FRRS			

Byte (HEX)	Name	Description
000	TCA.FCB	J.TSM terminal I/O file control block (sixteen words).
040	TCA.EFCB	External file I/O file control block (sixteen words).
080	TCA.SPAD	Command scratch area (sixteen words).
080	TCA.PSB	Mount request parameter send block (ten words). This area is overlaid by TCA.SPAD.
0A8	TCA.CNP	Caller notification packet. This area is overlaid by TCA.SPAD (six words).
0C0	TCA.MPAR	Macro/command file parameter area (32 words). Contains eight blocks of sixteen bytes for parameter storage. J.TSM uses this area for SYC, command file, and macro processing.
140	TCA.MSIZ	Macro/command file parameter length save area (eight bytes).
148	TCA.CPAR	Macro/command file argument area (32 words). Contains eight blocks of sixteen bytes for argument storage.
1C8	TCA.CSIZ	Macro/command file argument length save area (eight bytes).
1D0	TCA.LABL	Macro/command file target label set by a $GOTO directive.
1D8	TCA.MHDR	Macro/command file buffer header (five words).
IEC	TCA.MBUF	Macro/command file buffer (twenty words).
23C	TCA.DUMY	Reserved (one word).
240	TCA.STS2	Context status area two (one halfword). Bit variables assigned as follows:

Bit	Meaning if Set
0	First line of a macro/command file has been read
1	Macro is being processed
2	Target label from $GOTO is being searched for
3	Unrecoverable I/O occurred
4	Next record already read
5	Message receiver is being read
6	"$" is required
7	In command file mode
8	Macro end action requested
9	Next card must be a job card
10-15	Reserved

242	TCA.MNUM	Number of parameters in macro/command file (one byte).
243	TCA.CNUM	Number of arguments in macro/command file (one byte).
244	TCA.ERR	Last encountered abort code (one word).
248	TCA.LOGN	Binary logon time used to calculate connect time (one word).
24C	TCA.TERM	Initial terminal line and page size (one halfword).
24E	Reserved	One halfword.
250	TCA.STAT	Context area status flags (one word). Bit variables assigned as follows:

Bit	Meaning if Set
0	Terminal is in task mode
1	Terminal is in command mode
2	Message area one has a message
3	Message area two has a message
4	Next message will overwrite the previous message area
5	Terminal has a valid owner name (logged on)
6	Context is borrowed by another context to echo a message
7	Control file M.CNTRL is being processed

		8 Abort code is not to be expanded (for example, OPTION NOERROR) 9 Message reception is inhibited 10 Batch job 11 $JOB card required 12 Spool file requires deassignment 13 Sequential job 14 SLO is assigned to UT (for example, $SYSOUT UT) 15 Job is in effect 16 Remove request is outstanding 17 Command echo is inhibited (for example, OPTION NOCOMMAND) 18 Previous task aborted 19 Indicates this context to the SHOWUSERS command 20 Waiting for memory pool 21 Asynchronous messages inhibited 22 Asynchronous messages allowed 23 Synchronous messages requested (for example, OPTION QUIET) 24 Asynchronous messages requested (for example, OPTION UNQUIET) 25 Exit is deferred by busy FCB 26 Dead modem wind down is in effect 27 Echoplex inhibit during logon in effect 28 Current job is a submit 29 SLO is full (IO98 on SLO) 30 Synch mode set on call to gate 31 Reserved
254	TCA.LBFA	Current M.TSCAN line buffer address in memory pool (one word).
258	TCA.UDTA	UDT address of terminal for this context (one word).
25C	TCA.TNUM	Task number of currently active task for this context (one word).
260	TCA.JFLG	Conditional flags set by $SETF and $RESETF commands (one word).
264	TCA.MCEA	Macro/command file end action address (one word).

268	TCA.RRSF	Number of free words available in the RRS area (one word).
26C	TCA.RRSN	Address of next available RRS entry (one word).
270	TCA.CPU	Accumulated CPU time for accounting (one word).
274	TCA.IPU	Accumulated IPU time for accounting (one word).
278	TCA.ACES	Privilege access bits defined in M.KEY (one doubleword).
280	TCA.SGOS	Long RID for SGO file (eight words).
2A0	TCA.SLOS	Long RID for SLO file. This area is used by J.TSM to build the SLO run request for the J.SOEX spooler task (eight words).
2C0	TCA.SLOD	System listed output device (one doubleword).
2C8	TCA.NAME	SLO project/job name (one doubleword).
2D0	TCA.JOB#	SLO binary job number (one word).
2D4	TCA.LFLG	SLO listed output flags (one word). Bit variables assigned as follows:

Bit	Meaning if Set
0	Do not delete file on deassignment
1	Copy/reprint request
2	Output is unformatted
3-23	Reserved

2D7		0 = print, 1 = punch, 2 = plot, 3-255 = reserved.
2D8	TCA.SBOS	Long RID for SBO file. This area is used by J.TSM to build the SBO run request for the J.SOEX spooler task (eight words).
2F8	TCA.SBOD	System binary output device (one doubleword).
300	TCA.NAMB	SBO project/job name (one doubleword).
308	TCA.JBB#	SBO binary job number (one word).
30C	TCA.BFLG	SBO listed output flags (one word). Bit variables assigned as follows:

Bit	Meaning if Set
0	Do not delete file on deassignment
1	Copy/print request
2	Output is unformatted
3-23	Reserved

30F		0 = print, 1 = punch, 2 = plot, 3-255 = reserved.
310	TCA.CDIR	Current working directory (four words).

320	TCA.CVOL	Long RID for current working directory (eight words).
340	TCA.HDR1	Header for first message. This starts the area J.TSM uses for $SIGNAL and system task messages to terminals (four words).
350	TCA.ONR1	Owner name of first message (three words).
35C	TCA.DAT1	Date of first message (three words).
368	TCA.TIM1	Time of first message (two words).
370	TCA.MBX1	First message buffer, mailbox one (21 words).
3C4	TCA.HDR2	Header for second message (four words).
3D4	TCA.ONR2	Owner name of second message (three words).
3E0	TCA.DAT2	Date of second message (three words).
3EC	TCA.TIM2	Time of second message (two words).
3F4	TCA.MBX2	Second message buffer, mailbox two (21 words).
448	TCA.SP01	Gate/ungate context register three save area.
44C	TCA.SP02	Gate/ungate context register one save area.
450	TCA.SP03	Gate/ungate call from subroutine R0 save area.
454	Reserved	Fourteen words.
480	TCA.MODE	Task mode control byte. This starts the area of the TCA that is used by J.TSM as the parameter task activation block (one byte). Bit variables assigned as follows:

Bit	Meaning if Set
0	Reserved
1	Job oriented
2	Terminal task
3	Batch task
4	Load debugger with task
5	RTM resident, Establish
6	Command file active
7	SLO assigned to SYC

481	TCA.NRRS	Number of RRS entries (one byte).
482	TCA.ALLO	Number of 512 word pages of memory that this task will require as specified by $ALLOCATE (one byte).
483	TCA.MEMC	Memory class (one byte).
484	TCA.NBUF	Number of blocking buffers to reserve (one byte).
485	TCA.NFIL	Number of files, FAT/FPT pairs, to allocate (one byte).
486	TCA.PRIO	Task priority of task (one byte).

487	TCA.IOER	I/O error counter, not used by M.PTSK (one byte).
488	TCA.LMN	Load module name (one doubleword).
490	TCA.SUDO	Task pseudonym (one doubleword).
498	TCA.ONRN	Task owner name (one doubleword).
4A0	TCA.PROJ	Task project (one doubleword).
4A8	TCA.USRK	Number of VAT entries to reserve (one byte).
4A9	Reserved	Three bytes.
4AC	TCA.PGOW	Task option word, $OPTION (one word).
4B0	TCA.USW	User status word, initial value (one word).
4B4	TCA.FRRS	RRS entries (192 words). The first RRS entry is always the definition for UT.

3.2.4 Intertask Communications

J.TSM interacts with three system tasks: J.SSIN, J.SOEX, and J.MOUNT. J.SSIN spools input job streams and queues the job requests to the run request head cell in J.TSM's DQE. J.SOEX is sent run requests by J.TSM to service the PRINT command. J.MOUNT is sent run requests by J.TSM to service the MOUNT command.

3.3 System Mount Task (J.MOUNT)

The system nonresident media mounting task, J.MOUNT, is executed with a run request from the Resource Management Module (H.REMM). J.MOUNT performs the following functions:

- Mounts a formatted volume. For example, disc or floppy disc.
- Mounts an unformatted medium. For example, tape, disc or floppy disc.
- Dismounts a formatted volume
- Dismounts an unformatted medium

Information in the run request from H.REMM determines which function J.MOUNT performs.

3.3.1 Run Request Interface

H.REMM passes a block of information with each run request to J.MOUNT. This block can be one of two lengths, two words or sixteen words. If it is two words, a formatted mount is being requested. If it is sixteen words, any of the other three functions may be requested, depending on the information contained in the block.

3.3.1.1 Formatted Mount Requests

The formatted mount request from H.REMM carries with it a two word message. This message contains the following information:

Word 0	Mount device specification
Word 1	Flag mounted volume table address

The mount device specification contains the device type in byte zero and the device channel address, if present, in byte two with bit zero set to indicate its presence. The device subchannel address, if present, is supplied in byte three with bit sixteen set to indicate its presence.

The flag field has the following bit significance when set:

Bit	
0	Mount message inhibited
1-7	Reserved

The mounted volume table address contains the address of the Mounted Volume Table Entry (MVTE) to be completed for this volume.

3.3.1.2 Unformatted Mount Requests

When requesting an unformatted mount, H.REMM passes a sixteen word message. This message contains the File Assignment Table (FAT) entry for the device on which the volume is to be mounted.

3.3.1.3 Volume Dismount Requests

H.REMM passes a sixteen word message with both formatted and unformatted dismount requests. This message, like the unformatted mount, contains the FAT entry for the device requiring the dismount. J.MOUNT distinguishes between mount and dismount requests by testing bit two of DFT.ACF (mount message output) in the H.REMM supplied FAT. If this bit is set, a dismount is being requested. J.MOUNT distinguishes between a formatted or an unformatted dismount by testing bit four of DFT.STB (unformatted medium). If this bit is set, unformatted is indicated.

3.3.2 Mount Messages

For formatted volumes, the mount message is as follows:

```
MOUNT DISC VOLUME volname ON DRIVE ddccss
TASK taskname,taskno REPLY R,H,A OR DEVICE:
```

volname	is the one- to sixteen-character volume name
ddccss	is the device mnemonic, channel, and subaddress
taskname	is the eight-character task name
taskno	is the eight-digit hexadecimal task number

For unformatted volumes, the mount message is:

```
MOUNT name VOL nnn ON ddccss
TASK taskname,taskno REPLY R,H,A OR DEVICE:
```

name	is the one- to four-character reel ID
nnn	is the volume sequence number if multivolume tape
ddccss	is the device mnemonic, channel, and subaddress
taskname	is the eight-character task name
taskno	is the eight-digit hexadecimal task number

Reply R to indicate the volume is ready and proceed with processing. Reply A to abort the requesting task. Reply H to hold the requesting task and reprocess the mount at a later time. If desired, a device other than the one being requested can be specified for the volume to be mounted upon by specifying the desired device in response to the mount message. The format used is similar to the one displayed in the mount message (e.g., DM0804, M91001). If this option is specified, another mount message is issued to confirm the user's chioce. The device for a formatted mount may be changed several times before replying R, H, or A. The device specification for an unformatted mount may only be given once. If a second device specification is given, a warning message and a repeat of the mount message are displayed.

An additional option, available only on unformatted mount requests, is the specification of recording density. If the requested device allows software control of the density, the user may specify it following the R reply to the mount message. The available options are listed below:

Reply	Meaning
R	Use default density specification
RN	NRZI mode, 800 bpi
RP	PE mode, 1600 bpi
RG	GCR mode, 6250 bpi
R800	NRZI mode, 800 bpi
R1600	PE mode, 1600 bpi
R6250	GCR mode, 6250 bpi

The density specification, if given, must follow the R with no intervening comma or space as shown above.

3.3.3 Formatted Volume Clean-up

J.MOUNT verifies the volume name for the current volume by comparing the name in the MVTE supplied by H.REMM with the name in the volume descriptor of the volume currently on the requested device. If the names do not match, J.MOUNT issues a message on the operator's console indicating this fact, and provides the actual name of the volume. The exception to this is when the system initializer, SYSINIT, is running. In this case, J.MOUNT does not check, but uses the volume name contained in the volume descriptor since SYSINIT does not know the name of the system volume or the swap volume, only the device address. Normally, for these cases, a mount message is not displayed. If a user modification to SYSINIT turns on the mount message for the system or swap volume, the volume name, SYSTEM VOLUME, is displayed.

In addition to checking the name, J.MOUNT checks the last dismount date and time against the current date and time. If the current value is before the dismount value, a warning message is displayed and the user may continue or abort. This check is supplied to help detect incorrect entry of date and time that may cause problems in the operating system, which uses date and time marking on files.

When mounting a formatted volume, J.MOUNT checks to determine if the volume had been previously dismounted. If this is the case, volume clean-up is not needed since H.REMM will have updated the volume descriptor at dismount time to reflect the proper values for available and allocated space. Additionally, the space and descriptor maps should also reflect the current condition of the disc as H.VOMM maintains these during normal system operation.

If the volume has not been dismounted, it is assumed the volume descriptor and the maps do not contain valid information, and the possibility exists there are temporary files that must be deleted and spool files that have not been processed (this would be a possibility following a system crash). In this case, J.MOUNT performs volume clean-up. For multiport disc mount, volume cleanup is an option for the user.

During volume clean-up, J.MOUNT reads all resource descriptors on the disc being mounted starting with the volume descriptor, rebuilds the space map, the descriptor map, and the pertinent counts contained in the volume descriptor. Each type of resource descriptor is processed according to its particular needs. For example, temporary files are deleted, permanent files are marked as allocated in the SMAP and DMAP, and spool files are resubmitted.

If an invalid resource descriptor is detected, J.MOUNT zeroes words 7 and 22 of the resource descriptor (the resource descriptor type and the link count), and places the reason for the problem, in ASCII, in the free section of the resource descriptor. Currently, an invalid resource descriptor type field and a bad segment definition are the only reasons for marking a resource descriptor invalid. If control switch six is set during mount and the system debugger is configured into the current system, J.MOUNT enters the debugger when it encounters an invalid resource descriptor, thereby allowing the user to discover the nature of the problem (register two contains the address of the invalid resource descriptor). For multiport resources, all access information and resource descriptor locks are reinitialized.

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

```
FILE OVERLAP HAS OCCURRED IN RDnum
RD TYPE num
FILENAME is name
SECTORS num THROUGH num
```

num is a hexadecimal number

name is the one- to sixteen-character file name

To mount the disc so that data can be recovered, set control switch seven.

The following control switches are applicable to the clean-up of formatted volumes:

Switch	Function if Set
0	Inhibits volume clean-up by J.MOUNT
6	If J.MOUNT encounters an invalid resource descriptor due to an invalid resource descriptor type field or space definition, it branches and links to the system debugger (if present) with register two pointing to the resource descriptor
7	J.MOUNT prereads the file space bit map (SMAP) or the resource descriptor allocation bit map (DMAP); J.MOUNT does not perform file overlap detection
8	Delete spooled output files instead of resubmitting them for processing

3.3.4 Volume Dismounting

J.MOUNT is called by H.REMM to issue a dismount message for all volumes being dismounted. In addition, for formatted dismounts, J.MOUNT deallocates the mount device for the caller.

For formatted volumes, the dismount message is as follows:

DISMOUNT volname FROM ddccss

volname is the one- to sixteen-character volume name

ddccss is the device mnemonic, channel, and subaddress

For unformatted volumes, the dismount message is as follows:

DISMOUNT reel VOL nnn FROM ddccss

reel is the one- to four-character reel ID

nnn is the volume sequence number if multivolume tape

ddccss is the device mnemonic, channel, and subaddress

3.3.5 Error Status Return

J.MOUNT returns error status through the user status field of the sender's Parameter Send Block (PSB). A summary of these codes and their meaning follows:

Code	Explanation
0	Normal return, no error
1	Requested volume name does not match name in the disc's volume descriptor
8	Unrecoverable I/O error on requested volume
20	Unable to initialize the requested volume
37	Invalid request parameter length
254	User requested hold
255	User requested abort

3.4 Multiprocessor Recovery Task (J.UNLOCK)

J.UNLOCK is the multiprocessor recovery task. In a multiprocessor system, J.UNLOCK allows any processor to recover and continue processing when one of the port processors goes off-line. Resource locks, assign counts, and user counts owned by the off-line processor are removed from the shared volumes by an on-line processor.

3.4.1 Structure

J.UNLOCK is a resident, privileged task which resides in the any wait queue. It is activated by an OPCOM UNLOCK directive or by J.MOUNT when a multiprocessor shared volume is mounted. When the last multiprocessor shared volume is dismounted from the system, J.UNLOCK deactivates.

3.4.2 Entry Conditions

When activated by J.MOUNT, J.UNLOCK is linked to the any wait queue. J.UNLOCK is resumed when a run request is issued from J.MOUNT or when an OPCOM UNLOCK directive is issued. J.UNLOCK executes at priority 58.

3.4.3 Exit Conditions

Each time a multiprocessor shared volume is dismounted, J.MOUNT sends J.UNLOCK a run request. If no shared volumes are mounted when the request is received, J.UNLOCK exits.

J.UNLOCK has an abort receiver, and can be aborted by the OPCOM ABORT directive. When aborted, J.UNLOCK completes its current activity before exiting.

3.4.4 Multiprocessor Recovery

When one processor goes off-line, J.UNLOCK scans the Resource Descriptors (RDs) on the multiprocessor shared volumes. The resource allocation status of each RD is changed to reflect the on-line processors.

After the status is changed, each RD is allocated by space definition. Space definition allows access to a resource which was locked by an off-line processor. A time-out value prevents contention between RDs and other tasks in the same environment. The RDs are then recorded and processed.

J.UNLOCK compares the allocation information in each RD with the allocation information in the memory resident Allocated Resource Table (ART) for the RD. If an ART does not exist for an RD, the RD's allocation information is reinitialized. If an RD contains status for an off-line processor, the RD's information is reinitialized.

The allocation information in each RD is documented in Chapter 2.

Words 93, 94, and 95 correspond to words one, two, and three of the memory resident ART entry.

The processing of each Resource Descriptor (RD) contains four steps:

- Multiprocessor RD lock processing.
- Assign and user count processing.
- Resource exclusive and inclusive lock processing.
- Reader count and access processing.

Multiprocessor lock processing stores the RD locks in bytes two and three of word 191. If a lock in effect belongs to the port to be unlocked, word 191 is reinitialized.

Assign and user count processing reinitializes the RD's allocation information if the total assign count is zero. If the count is greater than zero, the specified processor's assign and user counts are reinitialized. The memory resident ART counts for the specified port processor are modified to reflect the new assign and user counts.

Resource exclusive and inclusive lock processing removes locks owned by an off-line processor. The locks are removed from AR.PORT within AR.FLAGS in the ART.

Reader count and access processing corrects the reader count and access modes.

There are eight combinations of resource descriptor (RD) access modes:

- Readers
- Readers plus one writer
- Readers plus one appender
- One writer
- One modifier
- One updater
- One appender
- One modifier plus one appender

Only one combination is allowed at a time.

The resource allocation flags field of the Allocated Resource Table (ART) is checked for set flags. To determine the access mode, J.UNLOCK uses the RD's access mode plus AR.WOWN and AR.WOWN2 from the on-line processor's ART.

For the readers group (combination one), the read access mode is set and the new reader count is set to equal the on-line processor's assign count. The access field is initialized if the reader count is zero.

For the reader plus one writer or appender groups, combinations two and three, if AR.WOWN or AR.WOWN2 is set, the read count is set to be one less than the on-line processor's assign count. If the resulting read count is zero, read access is removed. The read count plus the writer or appender equals the on-line processor's assign count.

For one writer, modifier, updater, or appender, combinations four, five, six, and seven, the reader count is zero. If the writer is not from this environment, the access field is initialized.

For the one modifier plus one appender group (combination eight), the reader count is zero. J.UNLOCK uses A.WOWN and A.WOWN2 to determine if the processor has modify, append, or both access modes set. The access field is set accordingly.

3.4.5 Error Status Return

J.UNLOCK sends a message to the operator console if an error occurs.

3.5 System Spooled Output Tasks (J.SOUT and J.SOEX)

3.5.1 Functional Description

The spooled output executive (J.SOEX) schedules activations of the spooled output task, J.SOUT. The activations are based on device availability.

If spooled output is requested on a device in use, the request is queued by J.SOEX on the SOEX Run Request Queue (SRRQ) until the device becomes available.

Spooled output--printing and punching--is controlled by J.SOUT. An activation of J.SOUT can exist for every output device configured into the system.

Upon completion of output spooling, J.SOUT sends a break to J.SOEX to indicate the end of spooling and exits the system.

3.5.2 Operational Design

J.SOEX is activated by a run request from a user task or one of the following tasks or services:

- J.TSM
- J.MOUNT
- M.DASN
- SYSINIT

When J.SOUT scheduling is complete, J.SOEX goes into a wait state. J.SOEX is resumed by new run requests or by devices becoming available. Once activated, J.SOEX never exits the system.

3.5.2.1 J.SOEX Message Receiver

The J.SOEX message receiver handles messages sent to it by OPCOM. All messages are sent in the wait mode with call back enabled. The messages have the following format:

		0 7	8 15	16 23	24 31
Word	0	Type. See Note 1.	Subtype. See Note 2.	Reserved	
	1	Variable message information. See Note 3.			
	2	Variable message information. See Note 4.			
	3	Variable message information. See Note 5.			
	4	Job or task number			
	5 6	Job or task name			
	7 8	Device mnemonic			

Notes:

1. This byte indicates the type of message as follows:

Type	Description
0	List print directive
1	List punch directive
2	Deprint directive
3	Depunch directive
4	Reprint directive
5	Repunch directive
6	Redirect directive

2. This byte indicates the message subtype to be used by J.SOEX as follows:

Type	Description
0	Default subtype for particular directive
1	Job/task
2	Task name
3	Device mnemonic

3. This word contains variable message information depending on the message type as follows:

Message Type	Message Information
0	Current SRRQ entry address
1	Current SRRQ entry address
2	Not used
3	Not used
4	Reprint/repunch count
5	Reprint/repunch count
6	Not used

4. This word contains variable message information depending on the message type as follows:

Message Type	Message Information
0	String forward address of current SRRQ entry
1	String forward address of current SRRQ entry
2	Not used
3	Not used
4	Reprint/repunch starting/stopping pages
5	Reprint/repunch starting/stopping pages
6	First word of the REDIRECT destination device mnemonic. This is zero if the default output device is used.

5. This word contains variable message information depending on the message type as follows:

Message Type	Message Information
0	String backward address of current SRRQ entry
1	String backward address of current SRRQ entry
2-5	Not used
6	Second word of the REDIRECT destination device mnemonic. This is zero if the default output device is used.

Message types zero and one retrieve information from the SRRQ for the LIST PRINT and LIST PUNCH directives. The information is returned to OPCOM as part of the call back.

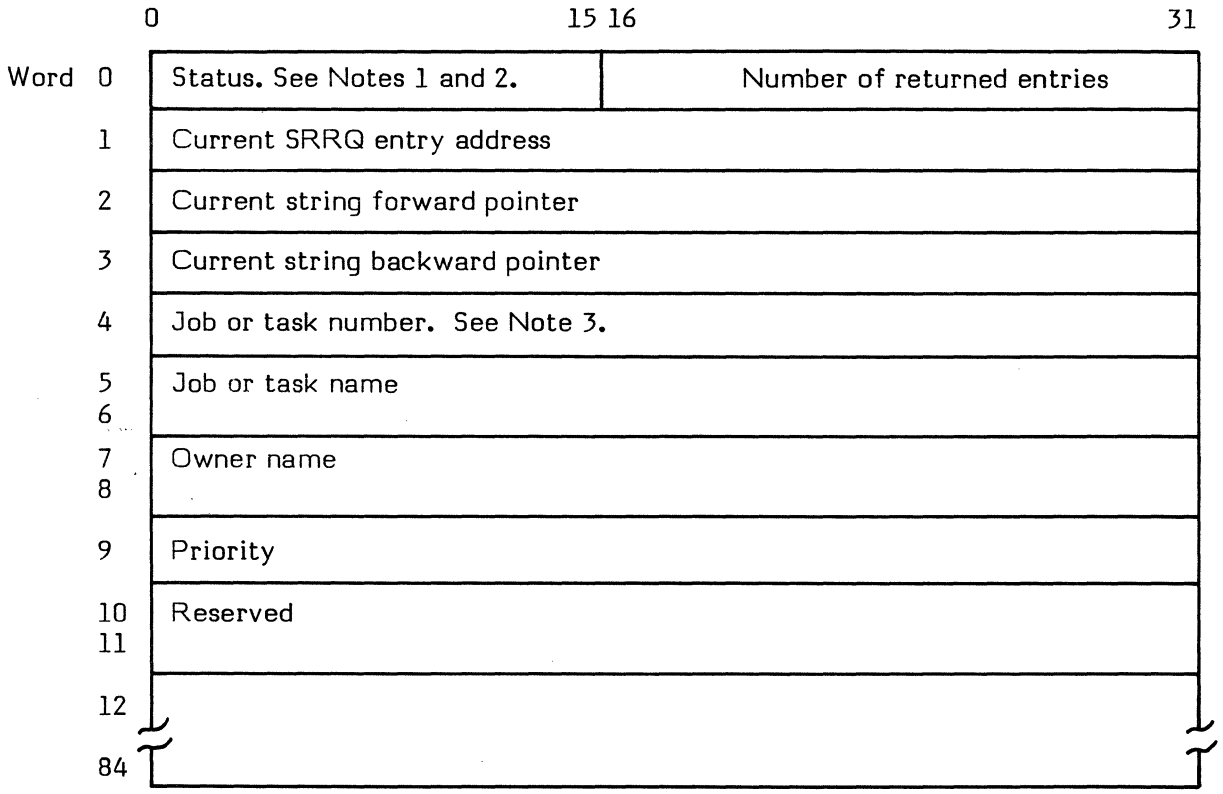
Message types two and three initiate SRRQ processing by J.SOEX for DEPRINT and DEPUNCH directives. No information is returned from these message types.

Message types four and five initiate SRRQ modifications by J.SOEX for REPRINT and REPUNCH directives. No information is returned from these message types.

Message type six initiates SRRQ modifications by J.SOEX for the REDIRECT directive. No information is returned from this message type.

3.5.2.2 Call Back Information

The call back can contain information on up to ten SRRQ entries. This information is formatted and displayed by OPCOM. The call back information is contained in a buffer with the following format:



Notes:

1. The first four words contain a header.

2. This halfword indicates the status as follows:

Status	Description
0	SRRQ not completely processed
1	SRRQ processed

If the status is zero, another message is sent to J.SOEX. J.SOEX then completes the SRRQ processing.

3. Words four through eleven make up one SRRQ entry description. These eight words can be repeated until the call back buffer contains ten SRRQ entries.

   The initial message for J.SOEX to process a LIST PRINT or LIST PUNCH directive contains zero values in words one through three. If another message is sent to J.SOEX, words one through three in the return buffer are used as words one through three in the new message. This lets J.SOEX continue directive processing where it left off in the SRRQ.

3.5.2.3 Return Status

When the message receiver completes processing and exits the message receiver level, return status is posted in the user status byte of the parameter send block. Returned status is defined as follows:

Status	Description
0	Operation successful
1	No entries in SRRQ
2	SRRQ pointers incorrectly linked
3	Error condition encountered
4	Invalid message type
5	Attempting to deprint SBO output
6	Attempting to depunch SLO output
7	Error encountered while attempting to delete resource
8	Invalid or missing default SLO device
9	Invalid or missing default POD device
10-255	Reserved

3.5.2.4 Break Receiver

J.SOEX has a break receiver that is entered when an interrupt is sent by a user task or any of the following tasks or services:

- J.SOUT
- J.TSM
- J.MOUNT
- M.DASN
- SYSINIT

J.TSM, M.DASN, and real-time tasks send interrupts to J.SOEX after a run request is sent. If a task does not send an interrupt, J.SOEX checks for queued run requests. If there are any, control is transferred to the break receiver that processes the run request.

When the break receiver is entered, information is transferred from the Memory Run Request Queue (MRRQ) to the J.SOEX Run Request Queue (SRRQ). Memory for the SRRQ entries is allocated from the logical address space of J.SOEX. The format of an SRRQ is the same as the format of a J.SOEX run request. See Section 2.33.3. After the transfer, the MRRQ entry is unlinked from the run request and the associated memory pool is deallocated.

CHAPTER 4

SYSTEM GENERATION TASK DESCRIPTION

4.1 Task Structure and Functional Organization

The System Generation Task, SYSGEN, is a privileged system task that operates within the framework of a standard MPX-32 system and can be executed in batch or interactive mode. It consists of an executive segment and five overlays. Figure 4-1 shows the loading sequence and gives a description of each phase.

System generation for an MPX-32 system involves supplying a set of configuration directives to the SYSGEN task. Section 4.1.2 shows the functional breakdown of directive processing for each overlay. The end result is the creation of a permanent file containing the installation specific MPX-32 system in memory image absolute format. This file may be subsequently restarted or utilized on a System Distribution Tape (SDT).

System generation by the SYSGEN utility is described in the MPX-32 Reference Manual Volume III. This chapter provides a functional description.

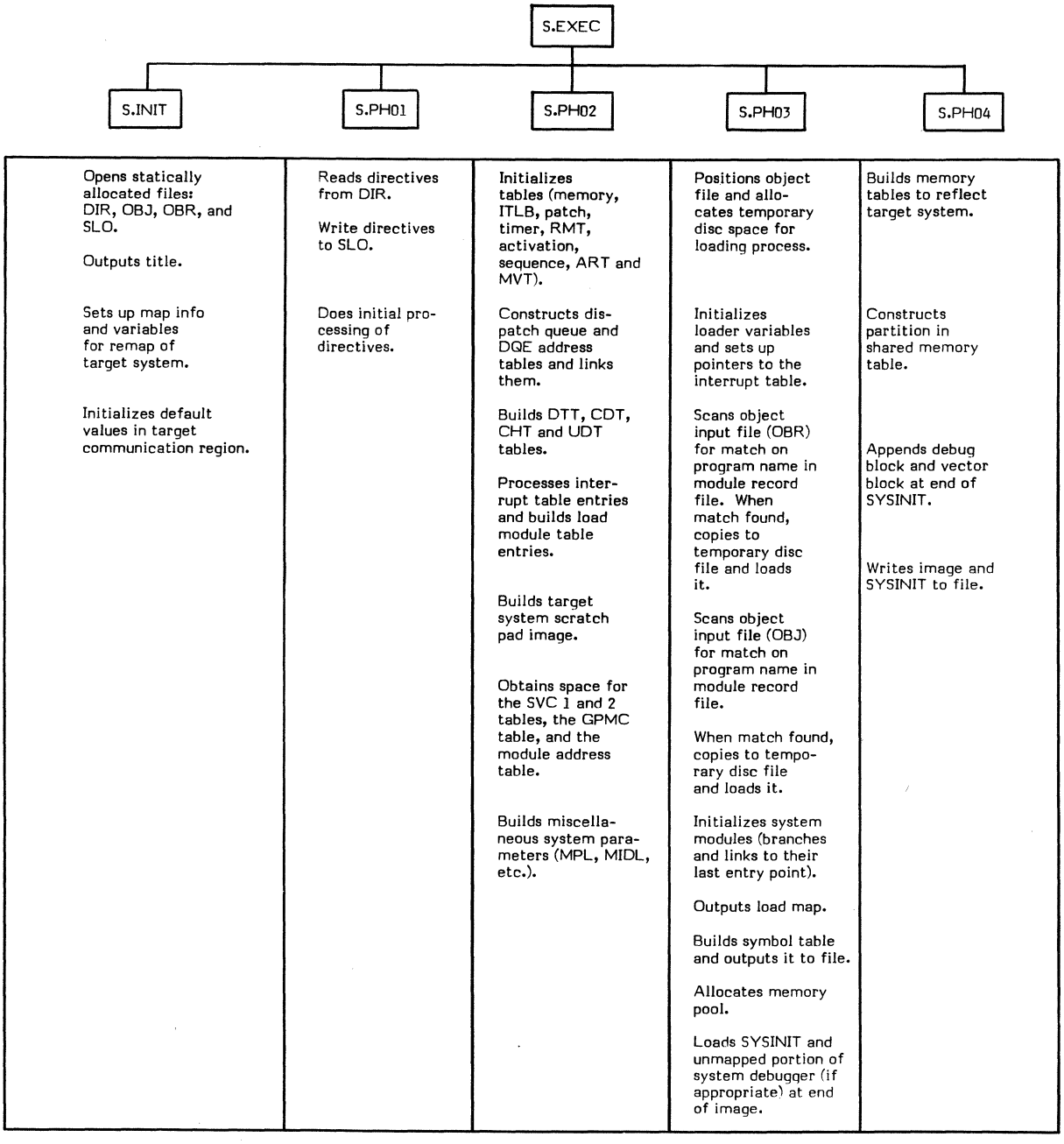


Figure 4-1. SYSGEN Overlay Structure and Functions (Page 1 of 6)

	S.INIT	S.PH01	S.PH02	S.PH03	S.PH04
//HARDWARE /PARAMETERS MACHINE = type		Sets C.MACH.			
IPU		Sets C.IPU in C.CONF.			
/MEMORY SIZE = nn, TYPE = c, CLASS = x, MULTI		Builds memory table prototype in scratch space. Parses memory types.	Sets C.MATA. Builds memory allocation table from prototype.		Allocates memory in target system. Builds MIDL.
/CHANNELS CONTROLLER = ttcc, PRIORITY = intlev, CLASS = class, HANDLER = name, MUX = type, SUBCH = aa, CACHE DEVICE = aa, DISC = devcode, SHR,DTC = tt, LINESIZ = x, PAGE = y, SPOOL = code, HANDLER = name, PHYSA = ccaa,OFF IOQ = mode,CACHE, QITD,DEAL START = start		Builds preliminary DTT, CDT, CHT and UDT as linked lists in scratch space. Adds handler to interrupt list. Sets C.CDTN, C.CHTN, and C.UDTN. Builds internal partition table in scratch space for each memory disc.	Builds DTT, CHT, CDT and UDT for target system. Sets C.DTTA, C.CHTA, C.DTTN, C.CDTA and C.UDTA. Reserves space for the GPMC jump table. Sets C.MIOP. Builds scratch pad entires.	Loads handler's object and initializes.	String CDTs. Uses partition table to initialize a shared memory table for each memory disc.
/TRAPS PROGRAM = (name 1, ..., name7) USERPROG = (name 1, ..., name 7) SYSTRAP = name 1, RETRAP = name 2		Builds internal interrupt table in scratch space. Builds internal interrupt table in scratch space. Replaces default trap name in trap table.	Builds load table with interrupt table. Builds scratch-pad entries. Builds load table with interrupt table. Builds scratch-pad entries.	Loads program's object and initializes. Loads program's object and initializes. Loads program's object and initializes.	

Figure 4-1. SYSGEN Overlay Structure and Functions (Page 2 of 6)

	S.INIT	S.PH01	S.PH02	S.PH03	S.PH04
/INTERRUPTS PRIORITY = intlev, RTOM = (channel, subaddress), PROGRAM = name, INTV		Builds internal interrupt table in scratch space.	Builds load table with interrupt table. Builds scratch pad entries.	Loads program's object and initializes.	
/SYSDEVS SID = devmnc, DENSITY = density, PARITY = parity		Sets C.SIDV, C.SIDD and C.SIDP.	Verifies C.SIDV.		
LOD = devmnc,IBP		Sets C.LODC and C.SIBP.	Verifies C.LODC.		
POD = devmnc		Sets C.PODC.	Verifies C.PODC.		
SWP		This directive is ignored.			
/VP VP = (aa, number), PROGRAM = module 1		Builds preliminary VP UDT in linked list. Builds preliminary null device CDT if necessary.	Builds VP UDTs. Builds null device CDT if necessary.	Loads VP handler object and initializes.	Strings CDT, if necessary.
VPID = aa, VPTYPE = tt, STARTBLK = blk, PRIORITY = intlev, INTRPT = (cc,ss), PROGRAM = module 2, IPCA = ipsize, BUS0 = b0size, BUS1 = b1size, BUS2 = b2size, BUS3 = b3size		Adds VP internal handler to interrupt table in SCR space. Specifies VP device specific information to be stored in preliminary UDT. Builds internal partition table entries for VP in scratch space.	Builds load table with interrupt table. Builds scratchpad entries. Builds VP UDTs.		Uses partition table to initialize SMT and allocate more memory to the system.
//SOFTWARE /PARAMETERS					
EXTDMPX	Defaults C.MPXBRD to -2.	Sets C.MPXBRD.	Builds load module table with extended MPX-32 modules.	Loads EXTDMPX object and Non-EXTD object.	Verifies EXTDMPX end address.
DISP = entries	Defaults C.NQUE to ten.	Sets C.NQUE.	Constructs dispatch queue and DQE address table and links them. Sets C.ADAT and C.DQUE.		Sets C.SWAP.
POOL = words	Defaults C.POOL to 1000.	Sets C.POOL.		Builds memory pool. Sets C.SBUF.	
NTIM = number	Defaults C.NTIM to 60.	Sets C.NTIM.			
MTIM = number	Defaults C.MTIM to 60.	Sets C.MTIM.			
ITIM = microseconds	Defaults C.ITRS to 384 (38.4 microseconds).	Sets C.ITRS.	Used to recompute C.IDLC, C.TDQ1, C.TDQ2, and C.TDQ3.		

Figure 4-1. SYSGEN Overlay Structure and Functions (Page 3 of 6)

	S.INIT	S.PH01	S.PH02	S.PH03	S.PH04
ITLB = intlev		Builds entry in internal interrupt table for H.ICP. Sets C.NITI.	Initializes indirectly connected interrupt table. Sets C.ITLB. Builds scratch pad entries.	H.ICP loaded from object file and initialized.	
MMSG=n	Defaults C.MMSG to 5.	Sets C.MMSG.			
MRUN=n	Defaults C.MRUN to 5.	Sets C.MRUN.			
MNWI=n	Defaults C.MNWI to 5.	Sets C.MNWI.			
PASSWORD		This directive is ignored.			
SYSTEM = sysfile		Sets C.SYSTEM.			Uses C.SYSTEM for name of target system file.
SYMTAB = filename		Sets C.SYMTAB.		Uses C.SYMTAB for name of symbol table file.	
TQFULL = time	Defaults C.IDCL to 26042, C.TDQ3 to 600, and C.TDQ2 to 400.	Sets C.TDQ3.	Recomputes C.TDQ3, C.TDQ2 and C.IDLC.		
TQMIN = time	Defaults C.IDCL to 26042, C.TDQ1 to 200, and C.TDQ2 to 400.	Sets C.TDQ1.	Recomputes C.TDQ2, C.TDQ1 and C.IDLC.		
BATCHPRI = nn	Defaults C.BPRI to 61.	Sets C.BPRI.			
TERMPRI = nn	Defaults C.TSMPRI to 60.	Sets C.TSMPRI.			
PATCH = number		Sets C.PATCH.	Sets C.MPAA, C.MPAC, C.MPAH, and zeros patch area.		

Figure 4-1. SYSGEN Overlay Structure and Functions (Page 4 of 6)

	S.INIT	S.PH01	S.PH02	S.PH03	S.PH04
MODE = code		Sets C.SCBT, C.NOP, C.SPADOK, C.SIBP, C.SUFA and C.SIMM as specified.			
SVC = number	Defaults C.SVTN to X '7F'.	Sets C.SVTN.	Sets C.SVTA and C.SVTA2 and zeros SVC tables.		
RMTSIZE = number	Defaults C.RMTL to 32 and C.RMTM to 64.	Sets C.RMTM.	Sets C.RMTA and zeros resource-mark table.		
FLTSIZE		This directive is ignored.			
ACTIVATE = (name1,...,name7)		Builds prototype activation table as linked list in scratch space. Sets C.ACTN.	Builds activation table and sets C.ACTA.		
TRACE = num	Defaults C.TRACE to X'FFFFFFFE'.	Sets C.TRACE.			
DEBUGTLC = cc	Defaults C.DBTLC to X'7E'.	Sets C.DBTLC.			
PCHFILE = name		Sets C.PCHFLE.			
DBGFILE = name	Defaults C.DBGLM to 'MPXDB'.	Sets C.DBGLM.			
SEQUENCE = (name1,...,name7)		Builds prototype sequence table as linked list in scratch space. Sets C.SEQN.	Builds sequence table and sets C.SEQA.		
DPTIMO = number	Initializes C.DPTIMO to zero.	Sets C.DPTIMO.			
DPTRY = num	Initializes C.DPTRY to zero.	Sets C.DPTRY			
DTSAVE = time		Sets C.DTSAVE.			
SWAPSIZE = size		Sets C.SWAPSZ.			
SWAPLIM = n		Sets C.PDQE.			
/MODULES MODULE = (name, module, entpoints)	Defaults C.MODN to twelve.	Builds prototype module table in scratch space. Sets C.MODN.	Uses prototype module table to build load table. Sets C.MODD and zeros module address table.	Loads modules and initializes.	

Figure 4-1. SYSGEN Overlay Structure and Functions (Page 5 of 6)

	S.INIT	S.PH01	S.PH02	S.PH03	S.PH04
/OVERRIDE					
SYSMOD = name1, REPMOD = name2		Removes compatibility modules or replaces module with name given in the system module table defined in SYSGEN.			
NOBASE		Removes base mode support.			
NOCMS		Excludes H.ALOC, H.FISE, H.MONS, and H.CALM from system image.			
/PARTITION					
NAME = name, SIZE = np, STRTPG = sp, MAP = pm		Builds internal partition table in scratch space.			Uses partition table to initialize shared memory table and allocate more memory to the system.
OWNER = name PROJECT = name OTHERS = name		Builds internal partition table in scratch space.			Uses partition table to initialize shared memory table and allocate more memory to the system.
/TABLES					
CDOTS = number		Sets parameters for user communication area.	Allocates user communication area.		
JOBS = number	Defaults C.JOBN to one.	Sets C.JOBN.			
MDT = n, BLOCK = b		Sets C.MDTN. Sets C.MDTA.			
SHARE = number		Sets C.SMTN.	Zeros shared memory table. Sets C.SMTA and C.SMTS.		Initializes shared memory table.
TIMER = number		Sets C.TENT.	Zeros timer table. Sets C.TTAB.		
/RMSTABLS					
ARTSIZE = number	Defaults C.ARTN to 100.	Sets C.ARTN.	Allocates and zeros ART. Sets C.ARTA.		
/FILES					
SMD		This directive is ignored.			
SYCSIZE = blocks	Defaults C.SYCS to 32.	Sets C.SYCS to 32.			
SGOSIZE = blocks	Defaults C.SGOS to 32.	Sets C.SGOS to 32.			

Figure 4-1. SYSGEN Overlay Structure and Functions (Page 6 of 6)

4.2 SYSGEN Components

SYSGEN contains the Device Type Table (DTT) definition, the Device ID (DID) table definition, and a special scanner used to parse SYSGEN directives.

4.2.1 DID and DTT Definitions

The Device Identification Table (DID) and the Device Type Table (DTT) definitions are defined using the Macro Assembler FORM directive. SYSGEN fills in the DTT with information from the Controller Definition Table (CDT), and vice-versa. The information used by SYSGEN is shown in Section 4.2.1.1. This information is used by SYSGEN to build the table layout of the DTT described in Chapter 2.

The DID is an internal structure containing disc identification information. Its layout is shown in Section 4.2.1.2. This information is used by SYSGEN to build Unit Definition Table (UDT) entries during the SYSGEN process.

4.2.1.1 Device Type Table

```
*****************************************************************************
*                          DEVICE TYPE TABLE (DTT)                          *
*****************************************************************************
            SPACE
            BOUND       1D
DTT.TBL     EQU         $
*
*                . . . . . . . . . . . . . . . . . . . . DEV. TYPE CODE
*                :   . . . . . . . . . . . . . . . . . . CDT POINTER
*                :   :   . . . . . . . . . . . . . . . . # OF CDT'S
*                :   :   :   . . . . . . . . . . . . . . FLAGS
*                :   :   :   :   . . . . . . . . . . . . DEVICE NAME
*                :   :   :   :   :   . . . . . . . . . . MAX BYTES/XFER
*                :   :   :   :   :   :   . . . . . . . . ALIAS DTC
*              . . : . . : . . : . . : . . : . . : . . : . FREE BYTE
DTT  FORM        8,  24,   8,    8,   16, 16,  8, 8
             SPACE
   DTT  X'00',A(00),X'00',X'00',C'CT', 4096,X'00',0 DUMMY CT
   DTT  X'01',A(00),X'00',X'41',C'DC',16384,X'00',0 ANY DISC EXCEPT
                                                    MEMORY DISC
   DTT  X'02',A(00),X'00',X'40',C'DM',16384,X'01',0 MOVING HEAD OR
                                                    MEMORY DISC
   DTT  X'03',A(00),X'00',X'40',C'DF',16384,X'01',0 FIXED HEAD DISC
   DTT  X'04',A(00),X'00',X'61',C'MT', 8192,X'00',0 ANY MAG. TAPE
   DTT  X'05',A(00),X'00',X'60',C'M9', 8192,X'04',0 9-TRACK MAG. TAPE
   DTT  X'07',A(00),X'00',X'01',C'CD', 0120,X'00',0 CARD DEVICE
   DTT  X'08',A(00),X'00',X'00',C'CR', 0120,X'07',0 CARD READER
   DTT  X'0A',A(00),X'00',X'00',C'LP', 0133,X'00',0 LINE PRINTER
   DTT  X'0B',A(00),X'00',X'00',C'PT', 4096,X'00',0 PAPER TAPE
   DTT  X'0C',A(00),X'00',X'00',C'TY', 4096,X'00',0 TELETYPE
   DTT  X'0D',A(00),X'00',X'01',C'CT', 4096,X'00',0 OPERATOR'S CONSOLE
   DTT  X'0E',A(00),X'00',X'60',C'FL',16384,X'00',0 FLOPPY DISC
   DTT  X'0F',A(00),X'00',X'00',C'NU',16384,X'00',0 NULL DEV
   DTT  X'10',A(00),X'00',X'00',C'CA', 4096,X'00',0 CA DEVICE
   DTT  X'11',A(00),X'00',X'00',C'U0', 0000,X'00',0 U0 DEVICE
   DTT  X'12',A(00),X'00',X'00',C'U1', 0000,C'00',0 U1 DEVICE
   DTT  X'13',A(00),X'00',X'00',C'U2', 0000,X'00',0 U2 DEVICE
   DTT  X'14',A(00),X'00',X'00',C'U3', 0000,X'00',0 U3 DEVICE
   DTT  X'15',A(00),X'00',X'00',C'U4', 0000,X'00',0 U4 DEVICE
   DTT  X'16',A(00),X'00',X'00',C'U5', 0000,X'00',0 U5 DEVICE
   DTT  X'17',A(00),X'00',X'00',C'U6', 0000,X'00',0 U6 DEVICE
   DTT  X'18',A(00),X'00',X'00',C'U7', 0000,X'00',0 U7 DEVICE
   DTT  X'19',A(00),X'00',X'00',C'U8', 0000,X'00',0 U8 DEVICE
   DTT  X'1A',A(00),X'00',X'00',C'U9', 0000,X'00',0 U9 DEVICE
   DTT  X'1B',A(00),X'00',X'00',C'LF', 0000,X'00',0 PRINTER/FLOPPY
*
```

4.2.1.2 Device ID Table

```
*********************************************************************
*                          DEVICE ID TABLE                          *
*********************************************************************
          SPACE
          BOUND              1W
DID. TBL  EQU                $
*
*DEVICE ID NAME.............................................:
*TOTAL SECTORS ........................................     :
*BIT MAP SIZE      ..........................       :       :
*NO. OF HEADS      ...................     :        :       :
*SECTOR SIZE       ................  :     :        :       :
*SECTORS/TRACK     ..............  : :     :        :       :
*SECTORS/ALOC. UNIT...........  :  : :     :        :       :
*SECTORS/BLOCK     ........  :  :  : :     :        :       :
*OLD DEVICE ID NAME....   :  :  :  : :     :        :       :
*                    :    :  :  :  : :     :        :       :
*                 ...:....:..:..:..:.:.....:........:.......:
DID       FORM          32, 8, 8,  8,  8, 16,   16,      32,      64
          SPACE
*         CLASS   'F' EXTENDED I/O DISC DEVICES
          DID      C'DF01', 3, 3, 26,  64,  2,     ,    4004, C'FL001    '
          DID      C'DF02', 1, 2, 20, 192,  5,  642,   41100, C'MH040    '
          DID      C'DF03', 1, 2, 20, 192,  5, 1286,   82300, C'MH080    '
          DID      C'DF04', 1, 4, 20, 192, 19, 2444,  312740, C'MH300    '
          DID      C'DF05', 1, 1, 20, 192,  4,  160,    5120, C'FH005    '
          DID      C'DF06', 1, 2, 20, 192,  1,  258,   16460, C'CD032    '
          DID      C'DF06', 1, 2, 20, 192,  1,  258,   16460, C'CD032    '
          DID      C'DF07', 1, 2, 20, 192,  1,  258,   16460, C'CD064    '
          DID      C'DF07', 1, 2, 20, 192,  3,  772,   49380, C'CD064    '
          DID      C'DF08', 1, 2, 20, 192,  1,  258,   16460, C'CD096    '
          DID      C'DF08', 1, 2, 20, 192,  5, 1286,   82360, C'CD096    '
          DID      C'DF09', 1,10, 20, 192, 40, 2635,  674400, C'MH600    '
          DID      C'DF0A', 1,10, 20, 192, 40, 2635,  674400, C'FM600    '
          DID      C'DF0A', 1, 1, 20, 192, 96,   60,    1920, C'FM600    '
          DID      C'DF0B', 1, 4, 20, 192, 10, 1286,  164600, C'MH160    '
          DID      C'DF0C', 1, 2, 20, 192,  5, 1286,      00, C'ANY      '
          DID      C'DF0D', 1, 4, 20, 192, 24, 2670,  341280, C'MH340    '
*
```

Word	0 — 7	8 — 15	16 — 23	24 — 31
0	Old device ID name (DID.ONAM)			
1	Sectors per block (DID.SPB)	Sectors per allocation unit (DID.SPAU)	Sectors per track (DID.SPT)	Sector size (DID.SSIZ)
2	Number of heads (DID.NHDS)		Bit map size (DID.MSIZ)	
3	Total sectors (DID.SECS)			
4	Device ID name (5 bytes) (DID.NAME)			
5	5th byte of DID.NAME	Reserved		

4.2.2 SYSGEN Scanner

The SYSGEN scanner parses SYSGEN directives by utilizing linked information tables built by calling system macros. The system macros SECTION, SUBSECT, DIRTV, KEYWD, and PARAM set up tables as shown in Section 4.2.2.1. SECTION and SUBSECT correspond to SYSGEN directive sections and subsections. KEYWD and PARAM correspond to the keyword and parameter elements of each directive - DIRTV. See the MPX-32 Reference Manual Volume III, Chapter 7 for directive descriptions.

The action addresses in Section 4.2.2.1 are addresses within the SYSGEN program where action should be transferred when the scanner encounters that directive, keyword or parameter type. The SDINIT macro must be called prior to setting up the information tables. It assigns the equates for the parameter type tables. Finally, the macro KWEND must be called after the set-up is complete to specify the end of the tables.

Entry Conditions

Calling Sequence:

```
BL      SCANNER
```

Registers:

R1	Address of directive definition list; a byte address on a word boundary
R2	Address of directive to scan; a byte address on a word boundary
R3	Negative length of directive in bytes

Exit Conditions

Return Sequence:

```
TRSW    R0  (CC1 = 1 if error detected)
            (CC2 = 1 if terminating section directive was encountered. For example, section definition has null subsection link)
```

Registers:

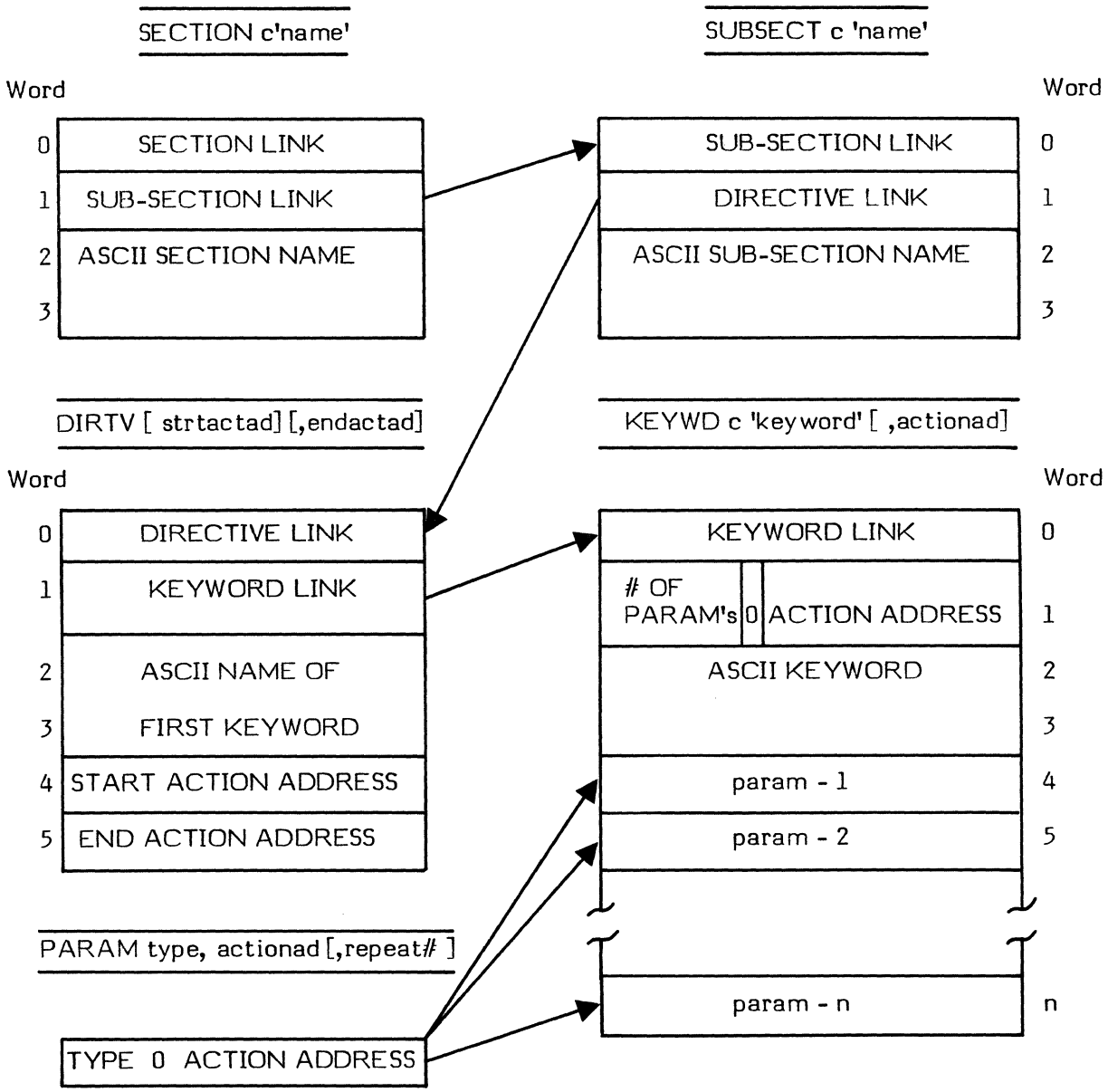
R1	Error message TCW if error detected
R2	Current directive pointer
R3	Negative length of remaining directive

Action Routine Linkage:

Inputs:
CC1 = 0
R0 = Return address
R2 = Byte address of item
R4 = Length of item, in bytes
R5 = Last character scanned
R6 = First four bytes of string
R7 = Second four bytes of string
(or)
R7 = Converted decimal number
(or)
R7 = Converted hexadecimal number

Outputs: CC1 = 1 if error detected by action routine.

4.2.2.1 Directive Definition List



Type label	Internal value	Description
G 'a'		ASCII Character
DIGIT	EQU X'84'	Digit (0-9)
ALPHA	EQU X'88'	Alphabetic (A-Z)
SPECL	EQU X'8C'	Special (not 0-9 or A-Z)
ANYS	EQU X'90'	Anything (X'00' - X'FF')
STRING	EQU X'94'	Alphanumeric string
SYMBOL	EQU X'98'	Symbol string
DNUMB	EQU X'9C'	Decimal number
HNUMB	EQU X'A0'	Hexadecimal number

4.3 Table Building

4.3.1 System Tables

The main function of SYSGEN is building the tables used by an MPX-32 system. Utilizing the supplied directives, SYSGEN tailors the tables for the installation required. Some of the system tables which SYSGEN builds are first formed as linked lists in SYSGEN's scratch space and are later inserted in the target file after all pertinent information has been collected. Section 4.3.1.1 provides a list of all the tables with which SYSGEN interfaces, and where information about them can be found.

4.3.1.1 Tables Referenced in SYSGEN

Name of Table	SYSGEN Interaction	Where Documented
Activation Table	Built by SYSGEN	Ref. Man., Vol. III, Ch. 7
Allocated Resource Table - ART	Allocated and zeroed by SYSGEN	Tech. Man., Vol. I, Ch. 2
Channel Definition Table - CHT	Partially built by SYSGEN	Tech. Man., Vol. I, Ch. 2
Controller Definiton Table - CDT	Built by SYSGEN	Tech. Man., Vol. I, Ch. 2
Device Identification Table - DID	Used by SYSGEN	Tech. Man., Vol. I, Ch. 4
Device Type Table - DTT	Used and filled in by SYSGEN	Tech. Man., Vol. I, Ch. 4
DQE Address Table - DAT	Built by SYSGEN	Tech. Man., Vol. I, Ch. 2
DQE Table	Allocated, zeroed & linked by SYSGEN	Tech. Man., Vol. I, Ch. 2
GPMC Jump Table	Allocated and zeroed by SYSGEN	See H.MUX0
Indirectly Connected Task Linkage Table - ITLT	Allocated and initialized by SYSGEN	Tech. Man., Vol. I, Ch. 1
Map Tables	MPL, MSD & MIDL built by SYSGEN	Tech. Man., Vol. I, Ch. 2
Memory Allocation Table	Built by SYSGEN	Tech. Man., Vol. I, Ch. 2
Memory Pool	Allocated and zeroed by SYSGEN	Tech. Man., Vol. I, Ch. 2
Memory Resident Descriptor Table	Allocated and initialized by SYSGEN	Tech. Man., Vol. I, Ch. 2
Module Table	Allocated and zeroed by SYSGEN	Tech. Man., Vol. I, Ch. 1
Module Address Table	Built by SYSGEN	Tech. Man., Vol. I, Ch. 2
Mounted Volume Table - MVT	Allocated and zeroed by SYSGEN	Tech. Man., Vol. I, Ch. 2
Patch Area	Allocated and zeroed by SYSGEN	Ref. Man., Vol. III, Ch. 9
Resourcemark Table - RMT	Allocated and zeroed by SYSGEN	Ref. Man., Vol. I, Ch. 2
Scratch Pad	Partially built by SYSGEN	32/70 Tech. Man.
Sequence Table	Built by SYSGEN	Ref. Man., Vol. III, Ch. 7
Shared Memory Table - SMT	Allocated and initialized by SYSGEN	Tech. Man., Vol. I, Ch. 2
SVC1 Table	Allocated and zeroed by SYSGEN	Tech. Man., Vol. I, Ch. 1
SVC2 Table	Allocated and zeroed by SYSGEN	Tech. Man., Vol. I, Ch. 1
Timer Table	Allocated and zeroed by SYSGEN	Tech. Man., Vol. I, Ch. 2
Unit Definition Table - UDT	Built by SYSGEN	Tech. Man., Vol. I, Ch. 2

4.3.2 Internal Tables

As SYSGEN processes directive input, it collects information in its internal tables for utilization later in the task. It has three tables that are limited to internal use: the partition table, the module table and the interrupt/trap table. See Section 4.3.2.1.

The partition table is built with information provided from the partition directives in phase one (S.PH01). Later, it initializes the shared memory table in phase four (S.PH04). SYSGEN's internal module table, not to be confused with the system module table, is a table of all system and user modules that are to be loaded in the target file. It is built in phase one and utilized in phase two (S.PH02) to build the load map table. The interrupt/trap table also builds the load map table and creates interrupt entries in the scratch pad in phase two.

4.3.2.1 SYSGEN Internal Tables

Partition Table

Word	0 7 8 15 16 23 24 31	
0	String forward address (PAR.LINK)	
1	Start logical page number (PAR.SPG)	Number of pages (PAR.NPG)
2	Physical map block number (PAR.PBN). See Note 1.	Reserved
3	Reserved	
4	Partition name (PAR.NAME)	
5		

Module Table

Word	0 7 8 15 16 23 24 31	
0	String forward address (MOD.LINK)	
1	Module number (MOD.NO)	Number of entry points (MOD.NEPT)
2	Module name (MOD.NAME)	
3		
4	Reserved	
5	Reserved	Module type (MOD.LTYP). See Note 2.

Interrupt/Trap Table

Word	0 7 8 15 16 23 24 31			
0	String forward address (INT.LINK)			
1	Interrupt type (INT.TYP). See Note 3.	Priority level (INT.L1)	Controller class (INT.CLS) See Note 4.	Reentrant descriptor (INT.REEN)
2	Interrupt handler name (INT.NAME)			
3				
4	Pointer to device handler list (INT.HNDA)			
5	Device type code (INT.DTC)	Channel number (INT.CHAN)	Subaddress (INT.SUBA)	Module type (INT.LTYP). See Note 2.

Notes:

1. Values for PAR.PBN for a memory disc only are assigned as follows:

Value	Meaning
-1	User did not specify a starting map block number. Bit UDT.MDST of UDT.STA2 in the corresponding UDT is reset.
-ve	User specified the starting map block number. This is the negative of that value. Bit UDT.MDST of UDT.STA2 is set. During phase four (S.PH04), this value is transferred to SMT.PAGE. It is then interpreted by SYSINIT or OPCOM when the position of the memory disc is determined.

2. Values for MOD.LTYP and INT.LTYP are assigned as follows:

Value	Meaning
1	System module (LTYP.SYS)
2	User module (LTYP.USR)
3	I/O handler module (LTYP.IO)
4	Non-I/O interrupt handler on trap (LTYP.INT)

3. Values for INT.TYP are assigned as follows:

Value	Meaning
1	Indirect (TYP.NDIR)
2	Direct (TYP.DIR)
3	Service interrupt (TYP.SI)
4	GPMC service interrupt (TYP.GPMC)
5	Extended I/O service interrupt (TYP.XIO)

4. Values for INT.CLS are assigned as follows:

Value	Meaning
3	RTOM interval timer (CLS.RTOM)
D	TCW class with extended addressing capability (CLS.TCWB)
F	Extended I/O (CLS.XIO)

4.4 Handler and Module Loading and Initialization

In phase three (S.PH03) of the SYSGEN task, the system modules, user modules, interrupt handlers and trap handlers are all loaded and initialized for the target file. SYSGEN reads the object input file (OBJ) and scans the load table built in phase two for a match on the program name in the binary object record file. When a match is found, the module is copied to a temporary disc file and subsequently loaded as often as it appears in the load table. If the object module does not match one entered in the load table, it is skipped.

The last entry point of all modules and handlers is reserved for SYSGEN initialization, and provides the capability of self-initialization. SYSGEN calculates the address of the last entry point of each module and does a branch and link to that location, thus initializing the module. On return from the initialization, by the macro M.XIR (see Chapter 1), the current entry in the load table is cleared, the current address pointer gets updated, and the initialization code is zeroed and overlayed with the next module.

4.5 Special Considerations

4.5.1 MAPTGT/MAPHOST Routines

SYSGEN initially obtains 16KW of extended memory in which to build the target system and acquires additional map blocks as needed. By using the internal routine MAPTGT, it can map its acquired memory to address zero, replacing the host operating system. This enables SYSGEN to use communication region equates as references within the target system instead of within the host. When it becomes necessary for SYSGEN to use host system variables or services, it utilizes the internal routine MAPHOST to put the host system back in place. Mapping back and forth prevents SYSGEN from being run with the debugger.

4.5.2 Special Case Activation

SYSGEN is a 16KW task. In order to allow SYSGEN to build target systems as large as 44 map blocks, the system allocator loads it at address X'60000', which is the 49th map block. This gives SYSGEN a maximum of 48 map blocks in which to map the target system and to obtain extra memory in nonextended address space for building system tables.

4.5.3 SYSINIT Loading

When SYSGEN has completed building the target image, the load table, and the symbol table, it then loads the object of the SYSINIT task, which directly follows J.SWAPR's object as the last object file on OBJ if the system debugger is not configured. When the system debugger is requested by the USERPROG directive, the unmapped portion of the debugger is loaded on the next page boundary immediately following H.SINIT. When SYSGEN's final output file (see Figure 4-2) is subsequently booted, control is transferred to the SYSINIT task which starts up the image and then exits. When present, the unmapped portion of the system debugger remains memory resident with 6KW of physical memory allocated to it. It is not included as part of the system map and does not increase the size of the logical address space occupied by MPX-32. See MPX-32 Reference Manual Volume III, Chapter 2.

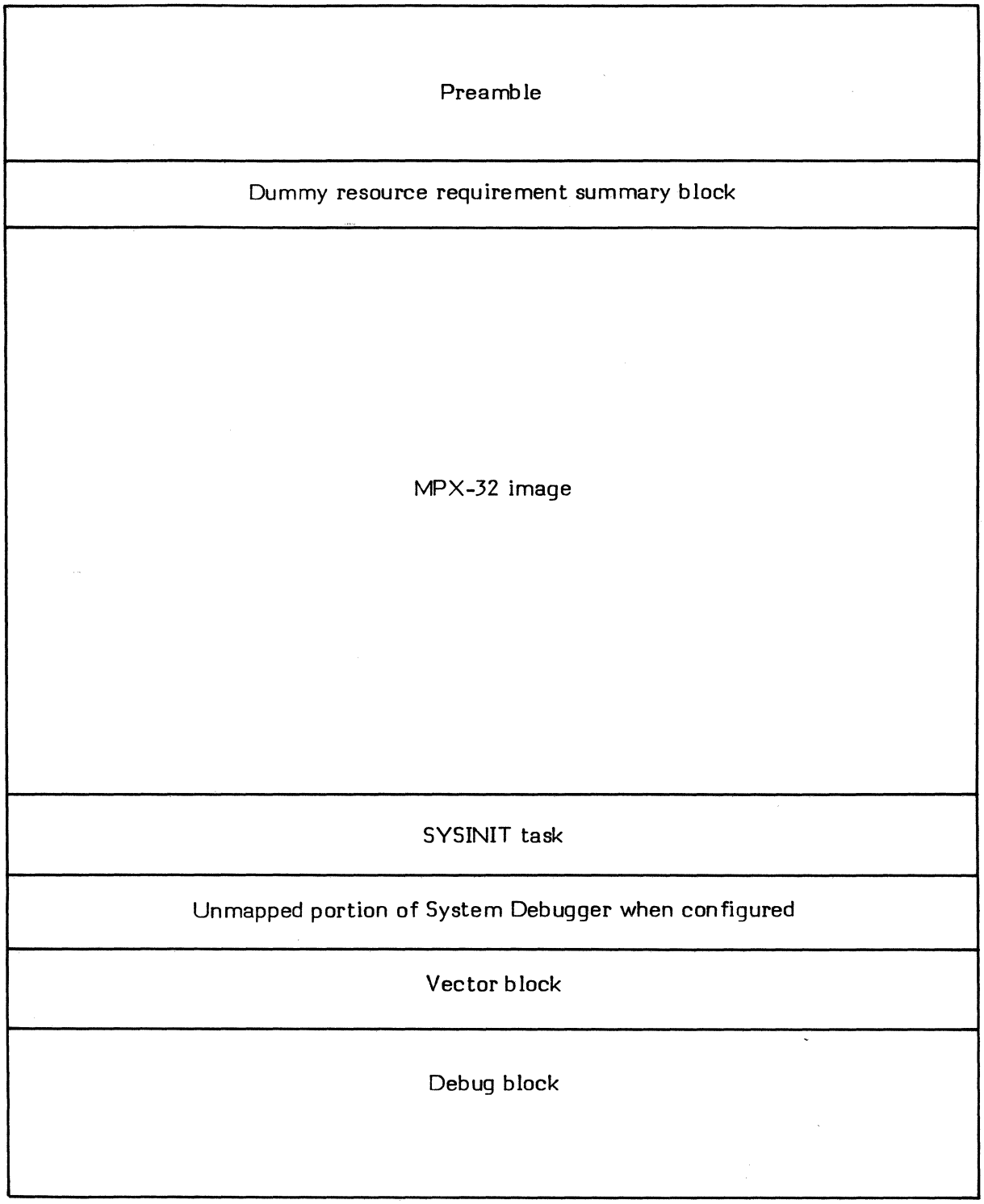


Figure 4-2. SYSGEN Output File Format

CHAPTER 5

BATCH TASK DESCRIPTIONS

5.1 CATALOGER

5.1.1 Introduction

The cataloger builds load modules from an object code file assigned to LFC SGO. External references are resolved from a user specified subroutine library assigned to LFCs DIR and LIB and from the system subroutine library assigned to LFCs LID and LIS. The catalog directives are input from LFC SYC and the load module map is output to LFC SLO. The load module is a permanent file created by the cataloger; its file name is the program name supplied in the CATALOG or BUILD directive.

Exit Conditions

Normal Exit:	SVC 1,X'55'
Abnormal Exits:	SVC 1,X'57' Abort

Abort cases are described in the MPX-32 Utilities Reference Manual.

5.1.2 Processing Regions

The cataloger consists of four processing regions, each is identified by a letter:

- X - External
- M - Main
- C - Control card interpretation and first object code pass
- B - Second object code pass

Program tags, subroutine names, and names of variables begin with the letter of the region with which they are associated.

5.1.2.1 X Region

The X region is composed of subroutines relating to MPX-32 provided services.

5.1.2.2 M Region

The M region is composed of subroutines, variables, and tables which are referenced by more than one region. It also contains the entry point called by MPX in response to the $EXECUTE CATALOG job control statement. When the entry point is called, the limits of the general table area are established and control is transferred to the C region. The general table area occupies the free memory allocated to the cataloger following the cataloger program logic. The utilization of this area is depicted in Figure 5-1.

5.1.2.3 C Region

The C region interprets cataloger directives and makes the first pass over the object code comprising each segment being cataloged. Information about each program element, its external definitions and common blocks is extracted and stored in the symbol table (SYMTAB). SYMTAB is built from the high memory end of the general table area toward low memory. SYMTAB data restored with the SYMTAB directive is stored first in the table. SYMTAB entry formats are presented in Figure 5-1. The first entry for each segment is the control entry. The control entry is followed by a program name entry.

For each segment, a table of external references from EXCLUDE directives is built from the low memory end of the general table area. This table is followed by a table of undefined external references. Names from INCLUDE directives are placed in the undefined external references table. The table also contains any unsatisfied external references encountered during the processing of a segment. If unsatisfied externals exist after all program elements have been processed from the SGO file for a segment, the subroutine libraries are searched for the externals. Program elements that satisfy external references are selected from the libraries. Any remaining undefined external references are ignored since they may be satisfied by segments that are subsequently processed.

If the first pass over the object code for all segments is successful, SYMTAB addresses are made module relative, and control is transferred to the B region.

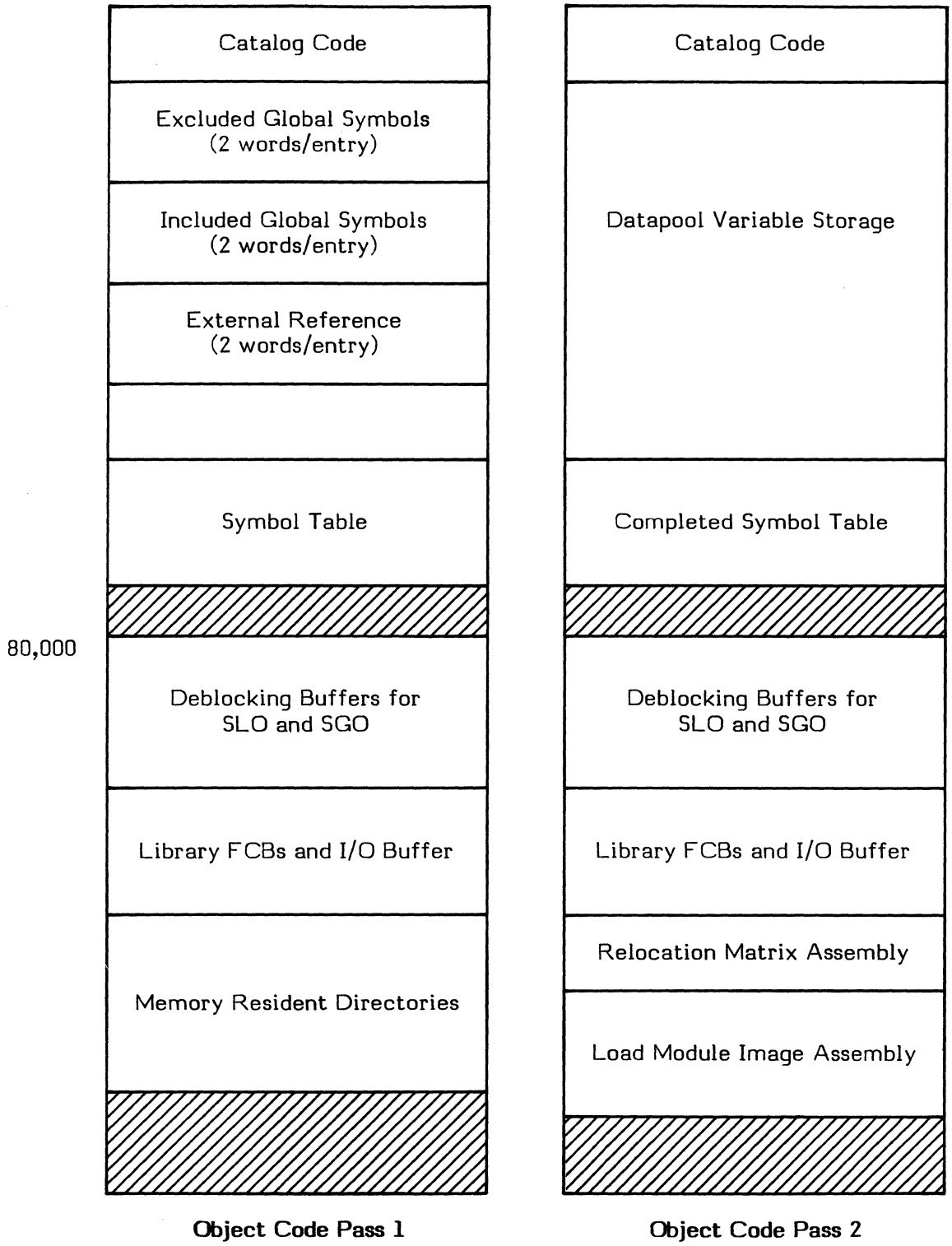


Figure 5-1. General Table Area

5.1.2.3.1 SYMTAB Entries

Linkback Entries

	0	6	8	16	31
0	000000	00	Overlay level	Sequence Number	
1	Must be zeros				
3					

If the Linkback directive is used, the identities of specified overlay segments are saved in entries built toward low memory in four-word blocks.

Segment (Module) Entry

	0	6	8	16	31
0	ID 100000	Flags. See Note 1.	Option flags. See Note 2.	Number of linkback entries	
1	000000	00	Overlay level (Main is zero)	Sequence number (Main is zero)	
2 3	Left-justified ASCII segment name				
4	000000	00	Transfer address		
5			Origin of segment TSA relative. See Note 3.		
6	Reserved				
7	Last (END) address of segment				

Notes:

1. Flags are as follows:

Flag	Description
1x	CATALOG directives for the overlay were preceded by an ORIGIN or LORIGIN directive
x1	Segment has a transfer address

2. Option flags are as follows:

Flag	Description
xxxxxxx1	Suppress printing of the module map
xxxx1xxx	Suppress output of load module to disc file
xx1xxxxx	Output segment's SYMTAB

3. Bit zero of the field is set if the origin is the end of a specified segment.

Defined Entry Point

	0	6	8 31
0	ID 010000	00	Reserved
1	Section number		Module relative address
2 3	Left-justified ASCII name		

Common Entry

	0	6	8 31
0	ID 001000. See Note 1.	Flags. See Note 2.	Size in bytes (zero if allocated in another element)
1	Block number		Module relative address. See Note 3.
2 3	Left-justified ASCII name		

Notes:

1. The ID is 001100 for the first common entry for a given global block.

2. The flags are as follows:

Flag	Description
11*	Common block is program initialized in this element
x1	Common block is Cataloger allocated in this element

* Catalog always allocates initialized common in the program that initializes it.

3. If allocated in another element, the address of the symbol table entry where the common is allocated.

Section Entry

	0	6	8	31
0	ID 000100	00	Section size in bytes	
1	Section number		Section origin	
2 3	Left-justified ASCII name			

Program Name

	0	6 7	8	16	31
0	ID 000010	Flags. See Note 1.	Subroutine library logical record number	Subroutine library block number containing first record	
1	Subroutine library file index	See Note 2.	Module transfer address if contained in this element		
2 3	Left-justified ASCII name				

Notes:

1. Flags are as follows:

Flag	Description
1x	Word 1 of entry contains module transfer address
x1	Program is from a subroutine library

2. If bit 7 is set, the transfer address is in the CSECT; otherwise, the transfer address is in the DSECT.

Control Entry

	0	6	8	16 31
0	ID 000001	Flags. See Note 1.	Control flags	Total of SYMTAB entries for this element
1	Maximum bound required in bytes		Number of bytes (mod 4) allocated this element for common and bounding	
2	Number of bytes (mod 4) of object code in this element			
3	Module relative address of this element, i.e., the address of its common if any			

Note:

1. The flags are as follows:

Flag	Description
1x	Program element is from a subroutine library
x1	Element is last in segment

2. If section code, bit 8 is set.

5.1.2.4 B Region

The B region makes a second pass over the object code and outputs the cataloged segments in load module format. Absolute overlays are output in absolute format.

At the beginning of the B region, the space in the general table area not occupied by SYMTAB is partitioned. An area large enough to build the core image of the largest program element in the segments being cataloged is reserved. An area for an associated relocation matrix is also reserved. Each bit in the matrix corresponds to a word in the program element data area, and is set to one if the word contains relative data. The remaining space is allocated for a datapool table. The three-word datapool table entries contain the datapool item name in the first and second words and the item's address in the third word. During the second pass over the object code, the datapool item table is searched for each datapool reference that is encountered. If not found, an attempt is made to locate the item in the datapool dictionary. If the item is found, it is added to the datapool table. Items are added sequentially to the table. Wraparound occurs when table space is exhausted with new items replacing previously stored items.

An image of each program element comprising each segment is built in memory before being written to the segment's disc file (if not suppressed). A module map of a segment is printed before the next segment is formatted. After all segments are processed, request SYMTABs are output.

5.1.3 Load Module Structure

A load module consists of one or more program elements in a form which requires only load origin biasing at load time. A program element is the unit of program organization bounded by the macro assembler PROGRAM and END directives. Program elements include programs written by the user and any subroutines called from subroutine libraries. All program elements must be assembled in the relative mode.

The six major load module segments are: preamble, resource requirement summary, CSECT data, CSECT relocation matrix, DSECT data, and DSECT relocation matrix. CSECT segments or DSECT segments can be omitted if they are empty. The resource requirement summary block is always present even if it is empty.

The load module preamble equates are described in Chapter 2.

Common blocks are allocated within a segment according to these rules:

- A common block is allocated preceding the program element which contains a common origin referencing the block.
- If a common block is not referenced by a common origin, it is allocated preceding the program element which defines the largest area.

Program areas, which are reserved but do not contain data and are not included in the module common delta, exist as words of zeros on disc. Therefore, those areas are initialized to zero when loaded into core. The module common delta is an area at the beginning of the module which occupies no space on disc and which is not initialized when loaded. It includes bounding and common which precedes the first program element and which precedes any common which is referenced by a common origin.

Each program element's assembled relative zero is placed on a doubleword boundary. Common blocks are placed on eight-word boundaries. The main segment of a module with overlays is placed on an eight-word boundary. If necessary, the size of the transient area is increased so that it consists of an integral number of eight-word units.

References can be made from an overlay segment to symbols contained within the main segment. These symbols may be contained in subroutines called from the subroutine libraries. The symbols are established using the assembler directives DEF and COMMON. Within the overlay, the assembler EXT and COMMON directives allow the symbols to be referenced. The main segment may reference symbols which are DEFs in overlay segments as main overlays reference symbols in lower level overlays. Other linkage depends on the use of the LINKBACK directive.

5.1.4 Symbol Table Output Format

The Symbol Table (SYMTAB) is output by the cataloger in a format similar to object records output by the assembler. Each record format is:

Byte 0	1	2,3	4,5	6
FF	Byte count*	Checksum	Sequence number	Data blocks

*Number of bytes in data block on card

Each data block is preceded by a control byte in the form XXXXNNNN. XXXX identifies the data block type and NNNN specifies the number of bytes of data in the block. If NNNN is zero, the number of bytes is sixteen. Data block types and their contents are as follows:

Type (Hex)	No. Bytes Data	Contents
0	16	Symbol table data output as four-word entries from high to low memory.
5	1 to 8	Name of the segment during whose cataloging the SYMTAB was output.
F	1	None - signals end of output.

A type five block is output first. Type zero blocks are output next and are terminated by a type F block. All cards, except the last, contain six data blocks. The last card can contain up to seven blocks (six data and one end).

5.1.5 Object Language

The object code is the output of the language processors and is the primary input to the cataloger. It describes the contents to be placed into the load module as a result of the inclusion of the object module into the cataloger's input stream.

5.1.5.1 Object Module Records

The object module consists of one or more variable length records up to 120 bytes in length. Each record contains six bytes of header information describing the object record.

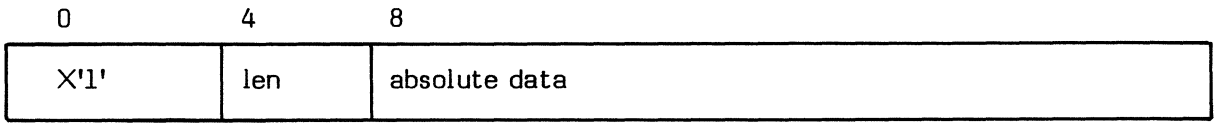
Byte	Field label	Contents
0	Record type	X'FF' or X'DF'. A value of X'DF' indicates the last record of the current object module.
1	Byte count	Number of data bytes in the current record. It ranges from 2 to 114 (X'2', to X'72') and does not include the six bytes of header information.
2,3	Checksum	Checksum of the data bytes within the record. It is computed by adding the data bytes and truncating the sum to a halfword.
4,5	Sequence	Sequence number of the current object record. The initial value is one. If the field overflows, the count is reset to one.

5.1.5.2 Object Commands

The data portion of the object records consists of a series of object commands. Each object command is described by a control byte. The control byte is the first byte (byte zero) of each object command and contains two fields, the function code and the byte count. The function code is the left four bits and ranges from X'0' to X'F'. The byte count is the right four bits and is the count of data bytes for each command, exclusive of the control byte. A byte count of zero is interpreted as X'10' data bytes.

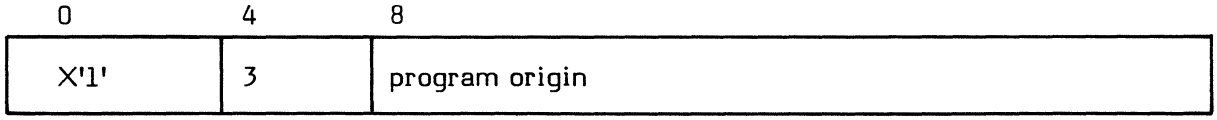
A stringback is a linked list of data that are terminated by a zero address. The word containing the zero address is absolute rather than relocatable. The cataloger makes that word relocatable if necessary. The addresses in the list are module relative and are nineteen-bit word addresses. The cataloger preserves bits 30 and 31 in stringing back the data.

5.1.5.2.1 Absolute Data



len — is the number of bytes of absolute data, 1 to 16. A value of zero is equivalent to a value of sixteen.

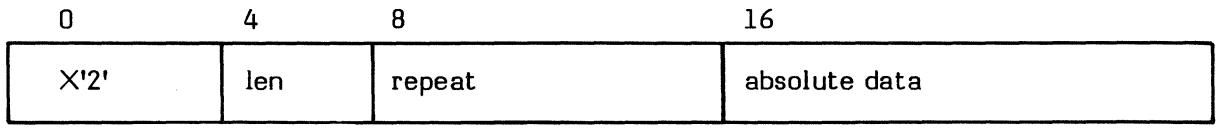
5.1.5.2.2 Program Origin



program origin — is a right-justified three-byte field containing a 19-bit origin

bit 8 — is set if the address is relocatable

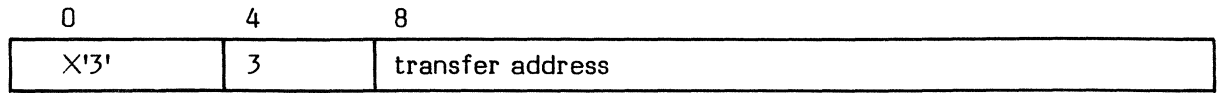
5.1.5.2.3 Absolute Data Repeat



len — is the number of bytes in the command

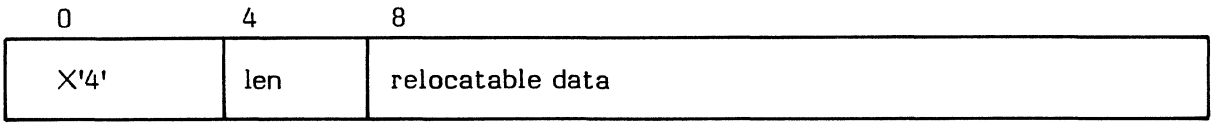
repeat — is the number of times to repeat data, 1 to 255. A value of zero is equivalent to a value of one.

5.1.5.2.4 Transfer Address



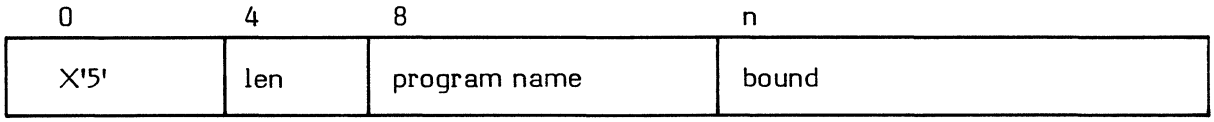
transfer address — is a right-justified three-byte field containing a 19-bit transfer address. Bit zero of the field must be set.

5.1.5.2.5 Relocatable Data



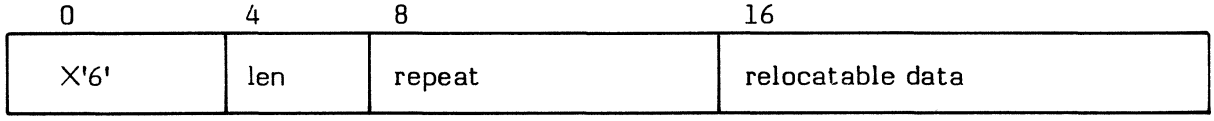
len — is the number of bytes of relocatable data, 4 to 16. A value of zero is equivalent to a value of sixteen. len must be a multiple of four.

5.1.5.2.6 Program Name



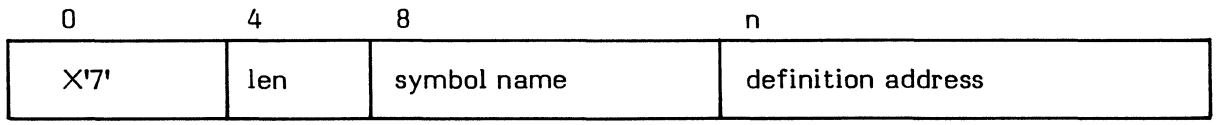
len — is the number of bytes in the program name plus three
program name — is the one- to eight-character program name
bound — is a three-byte field containing the minimum bounding requirement for the program. The minimum value is X'8'; the maximum value is X'20' or eight words.

5.1.5.2.7 Relocatable Data Repeat



len — is the number of bytes of relocatable data (4, 8, or 12) plus one
repeat — is the number of times to repeat data, 1 to 255. A value of zero is equivalent to a value of one.

5.1.5.2.8 External Definition



len — is the number of bytes in the symbol name plus three
symbol name — is the one- to eight-character name of the symbol being defined
definition address — is a three-byte field containing a right-justified 19-bit address. Bit zero of the field is set if the address is relocatable.

5.1.5.2.9 Forward Reference

0	4	8	n
X'8'	6	data	address

data — is a three-byte field containing a right-justified 19-bit address. Bit zero of the field is set if the address is relocatable.

address — is a three-byte field containing the right-justified 19-bit address of the stringback list. The contents of the data field is put into each word in the list. List is terminated by absolute zero link. Bit zero of the field must be set.

5.1.5.2.10 External Reference

0	4	8	n
X'9'	len	symbol name	stringback address

len — is the number of bytes in the symbol name plus three

symbol name — is the one- to eight- character name of the symbol being referenced

stringback address — is a three-byte field containing the 19-bit address of the stringback list. External address replaces the low order 19 bits in each word of the list. List is terminated by the absolute zero link. Bit zero of the field is set if the address is relocatable.

5.1.5.2.11 Common Definition

0	4	8	n	n + 1 byte
X'A'	len	common name	block	size

len — is the number of bytes in the common name plus three

common name — is the one- to eight-character name of common

block — is a one-byte block number assigned by the compiler

size — is a two-byte size of common in bytes

5.1.5.2.12 Common Reference

0	4	8	16
X'B'	len	block	common reference

len — is the number of bytes in the common reference plus one
block — is a one-byte common block number referenced by data
common reference — is 4, 8, or 12 bytes of data that reference the common block. The base address of the common block is added to the low order 19 bits of each reference word.

5.1.5.2.13 Datapool Reference

0	4	8	n
X'C'	len	symbol name	datapool reference

len — is the number of bytes in the symbol name plus four
symbol name — is the one- to eight-character name of the symbol in datapool
datapool reference — is a four-byte datapool reference. Symbol's value is added to the low order 19 bits of the datapool reference.

5.1.5.2.14 Escape to Extended Functions

0	4	8
X'D'	X	

Function code of X'D' indicates extended item. See Section 5.1.6.3.

5.1.5.2.15 Common Origin

0	4	8	16
X'E'	3	block	origin

block — is a one-byte common block number
origin — is a two-byte offset from beginning of common block

5.1.5.2.16 Object Termination

0	4	8
X'F'	1	0 0

This record terminates the object code for the current module.

5.1.5.3 Extended Object Commands

The extended object commands differ from objects commands in the following ways:

- Byte one contains the function code ranging from hexadecimal 1 to B.
- Byte two contains the length of the item including overhead bytes zero to three.

5.1.5.3.1 Section Definition

0	4	8	16	24
X'D'	0	X'0 1'	X'1 0'	bounding
section number		section size in bytes		
section name				

bounding — is the section bounding requirement (X'8' to X'20')
section number — is a one-byte field containing zero if DSECT or one if CSECT
section name — is an eight-byte field containing **DSECT* or **CSECT*

5.1.5.3.2 Section Origin

0	4	8	16	24
X'D'	0	X'0 2'	X'0 8'	bounding
section number		origin		

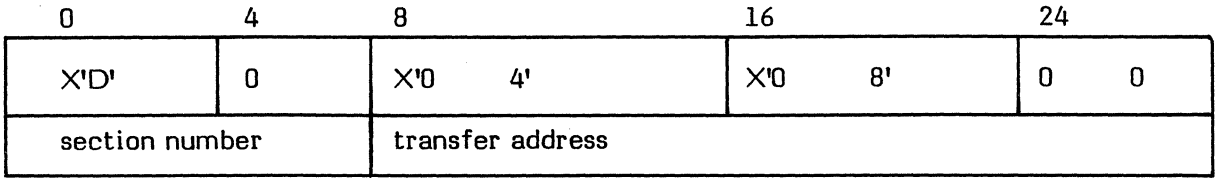
bounding — is the section bounding requirement (X'8' to X'20')
origin — is the offset within the section to establish as the new origin

5.1.5.3.3 Section Relocatable Reference

0	4	8	16	24
X'D'	0	X'0 3'	len	0 0
section number		repeat count	4 to 248 bytes of relocatable data	

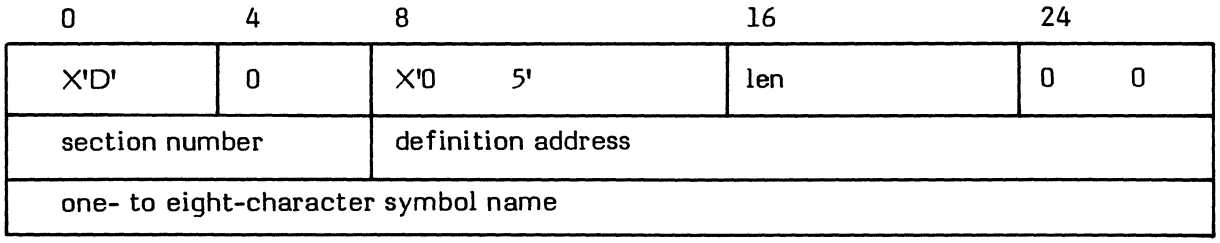
len — is the number of bytes in the command
section number — is the section number where the relocatable data references
repeat count — is the number of times to repeat the data. A value of zero is equivalent to a value of one.
relocatable data — is the data in multiples of four bytes whose right 19 bits are to be relocated by the section base

5.1.5.3.4 Section Transfer Address



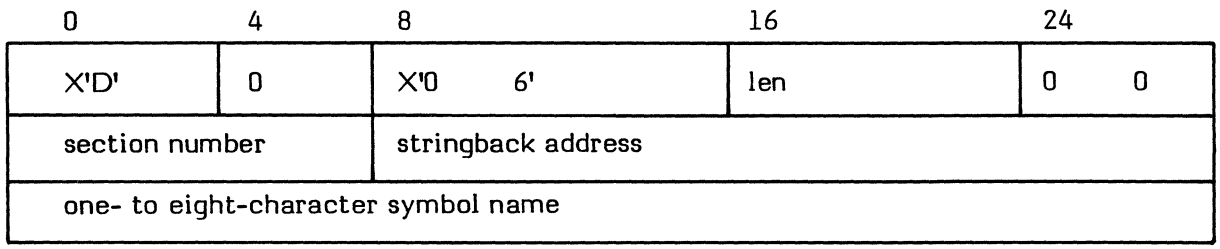
transfer address — is the offset within the section that is the transfer address

5.1.5.3.5 Section External Definition



len — is the number of bytes in the command
section number — is the section number where the symbol is defined
definition address — is the offset within that section

5.1.5.3.6 Section External Reference



len — is the number of bytes in the command
section number — is the section where the stringback list begins
stringback address — is the offset within that section
symbol name — is the global symbol referenced

Note: If the address is zero and bit zero of the address is set, a stringback is performed to address zero of the section.

5.1.5.3.7 Section Forward Reference

0	4	8	16	24
X'D'	0	X'0 7'	X'0 C'	0 0
section number		definition address		
section number		stringback address		

definition address — is the offset within the "section number" section where the symbol is defined

stringback address — is the offset within the "section number" section where the stringback list begins

5.1.5.3.8 Large Common Definition

0	4	8	16	24
X'D'	0	X'0 8'	len	0 0
common number		common size in bytes		
one- to eight-character common name				

len — is the length of the command

common number — is the identifier assigned by the compiler to the common block

common name — is the name of common block

5.1.5.3.9 Large Common Origin

0	4	8	16	24
X'D'	0	X'0 9'	X'0 8'	0 0
common number		common origin		

common number — is the number assigned by the compiler to the common block

common origin — is the offset from beginning of common

5.1.5.3.10 Large Common Reference

0	4	8	16	24
X'D'	0	X'0 A'	len	0 0
common number		repeat count	4 to 248 relocatable data	

len — is the length of the command
common number — is the identifier assigned by the compiler
repeat count — is the number of times to repeat the data. A value of zero is equivalent to a value of one.
relocatable data — is four-byte multiples of data which have the address of the referenced block added to the low order 19 bits

5.1.5.3.11 Debugger Information

0	4	8	16	24
X'D'	flg	X'0 B'	len	flag
type		address		
size			left-justified 8-character symbol name	
left-justified 8-character common name				

len — is the number of bytes in the command

Value	Meaning if Set
18	Symbol is not in common
26	Symbol is in common

flg — is as follows:

Bit	Meaning if Set
4	Symbol is in extended memory (address is that of a 24-bit pointer to the symbol)
5	Symbol is a formal parameter (address is that of a pointer to the symbol)
6	Symbol is in the common. The common name follows the symbol's name.
7	Symbol is in the datapool

flag — is as follows:

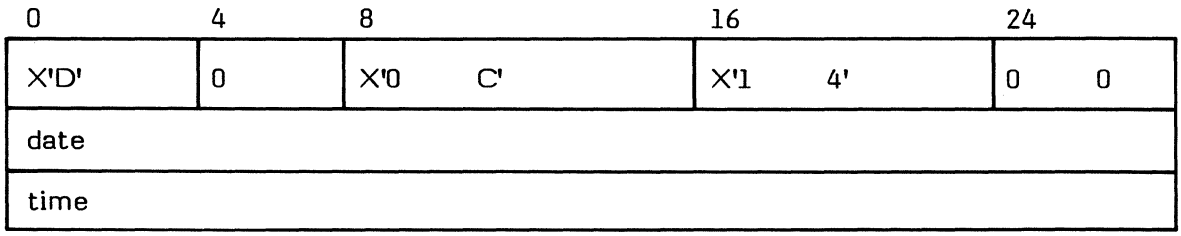
Bit	Meaning if Set
6	Address is absolute
7	Symbol is in CSECT

type — is as follows:

Type	Description
0	integer*1
1	integer*2
2	integer*4
3	integer*8
4	real*4
5	real*8
6	complex*8
7	complex*16
8	bit logical
9	logical*1
10	logical*4
11	character
14	statement label
15	procedure

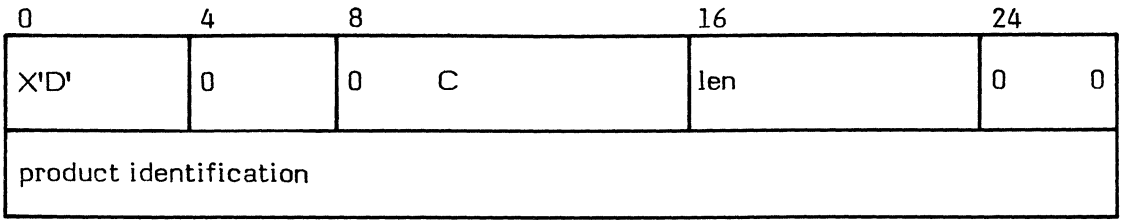
address — 23-bit address. Bits 9 to 28 indicate byte address, and bits 29 to 31 indicate bit within byte.
size — length of datum in bytes
symbol name — is an eight-character, left-justified, blank-filled variable name
common name — is an eight-character, left-justified, blank-filled common name

5.1.5.3.12 Object Creation Date/Time



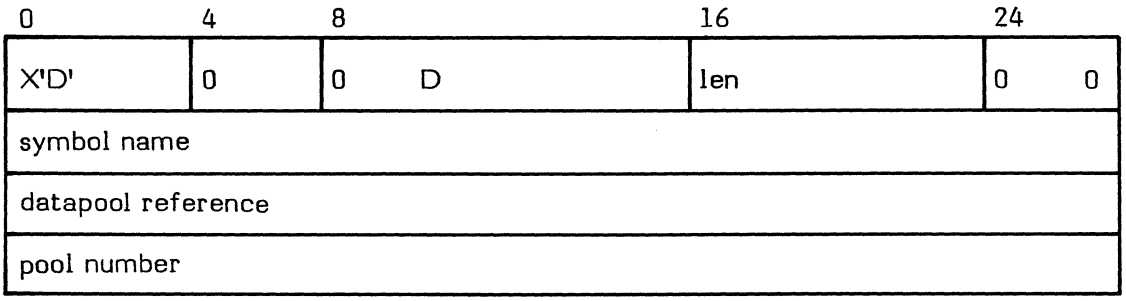
date/time — is the day and time the object code was generated
date — is the eight-byte ASCII date (mm/dd/yy)
time — is the eight-byte ASCII time (hh:mm:ss)

5.1.5.3.13 Product Identification Information Leader



len	is the number of bytes in the command
product identification	is a user supplied string of up to 32 bytes of text identifying the generated object code

5.1.5.3.14 Multiple Datapool Reference



len	is the number of bytes in the command
symbol name	is an one- to eight-character name of a symbol in the datapool
datapool reference	is a four byte datapool reference. The datapool symbol's value is added to the datapool reference.
pool number	is a value from 0 to 99 that identifies DPOOL00 to DPOOL99

5.1.5.4 Comparison of Assembler Instructions with Generated Object Commands

A source listing is provided for an example program OBJTCOMD. This program is not meant to have any utility or to suggest any recommended programming practices. It is written to show assembly instructions and the object commands generated by those instructions. The comments appearing on the right side of the listing lines are added to aid in the comparison of the assembler instructions with the generated object commands.

Compare the source listing in Figure 5-2 with the object code dump in Figure 5-3. This dump demonstrates the object code generated when instructions are assembled, and the position of object commands within the project.

Figure 5-2. Sample Source Listing

```
00014                                          C.3227   SETF
00015                                          C.TRACF  SETF
00016                                          C.MEMO   SETF
00017                                          C.SALONE SETF
00018                                          C.SYSGON SETF
00019                                          *                                                    OBJECT COMD TYPE
00020                                          *                                                    ----------------
00021                                                   PROGRAM  OBJTCOMD OBJECT_COMD_DEMO            5-     D10C
00022                                                   DEF      OBJTCOMD                                    D005
00023                                          DATAPOOL COMMON   D1(1),D2(2)
00024                                                   EXT      EXTEOBJT
00025    00000                                          ABS
00026    80000                                          ORG      X'80000'
00027                                          COMMLARG COMMON   L1(1)                                       A-
00028   C00000                                 ORGCOM   ORG      L1                                          E3
00029   C00000  00000001                               DATAW     1                                           0-
00030   P00000                                         REL                                                   D002
00031   P00000                                 OBJTCOMD EQU      $                                           D008
00032   P00000  AD800000   A00000                     LW        3,D1                                        C-
00033  *P00000                                         CSECT                                         D002    D001
00034  *P00000  AC800000   C00000                     LW        1,L1                                        B-
00035  *P00004  F8800001   X00000                     BL        EXTEOBJT                            0-      D006
00036   P00004                                         DSECT                                         D002    D001
00037   P00004  EC000009   P00008                     BU        TRANSFER                                    D003
00038   P00008             P00000              TRANSFER END      OBJTCOMD                            D008    D004

*  0000  ERRORS IN OBJTCOMD
```

```
BLOCK  WORD
 0000  0000  00000100 40000078 FF701908 0001584F ....8.........CO
       0004  424A5443 4F4D4400 0008D001 10080000 BJTCOMD.........
       0008  00082A2A 44534543 542A4401 10080100 ..**DSECT*D.....
       000C  00082A2A 43534543 542A4B43 4F4D4D4C ..**CSECT*.COMML
       0010  41524700 0004D00C 14003035 2F313032F ARG.......05/10/
       0014  38343136 3A32343A 3532D10C 14004F42 8416:24:52....OB
       0018  4A45435 4 5F434F4D 445F4445 4D4FD005 JECT_COMD_DEMO..
       001C  10000080 00004F42 4A5443 4F 404400000 ....OBJTCOMD..
       0020  00780078 DF571088 0002E300 00004000 .....W..........
       0024  00000100 02080000 00000006 4431AD80 ............D1..
       0028  0000D402 08000100 00008500 AC800000 ................
       002C  04F88000 01D00208 00000000 04D00030A ................
       0030  00000EC 00009004 06100001 80000445 ................
       0034  585445 4F 424A5404 00800000 0000001F1 XTEOBJT.........
```

Figure 5-3. Sample Object Code Dump

5.2 Macro Assembler

5.2.1 General Information

In addition to the assembler functions stated in the MPX-32 Utilities Manual, the assembler can generate base or nonbase code depending on the condition of Macro Assembler option 16.

Extended MPX-32 modules control option 16 with macro assembler directives to generate the appropriate code and adaptors that allow a module to execute in extended memory.

5.2.2 Directives

Some macro assembler directives form the adapter source coding that must be used to run a module in extended memory. Generation of the adaptation sequences is performed by a combination of macro assembler directives and special communication sequences. The adaptation coding enables the properly sectioned location of nonbase code, base code, adaptation code, and DSECT data with the proper changes in the instruction mode.

Recognition of base vs. nonbase assembly language is established by Macro Assembler Option 16. The OPTR (Option Reset), OPTS (Option Set), and OPTT (Option Test) macro assembler directives are used to test the state of option 16 and change the mode of the instruction set during the generation of adaptation sequences.

The following assembler directives can be used to implement extended MPX-32 modules and nonbase to base mode conversions.

Directive	Function
SSECT	Assembles the SYSGEN section of program source code
FLG_MPX SSECT	Allows communication of state changes in the object code to identify and load the code and data sections
OPTR	Resets a selected assembly option
OPTS	Sets a selected assembly option
OPTT	Tests a selected assembly option
SDEF	Identifies SYSGEN linkage symbols within a program that can be externally referenced
SEXT	Identifies externally defined SYSGEN linkage symbols that are referenced by the program
SORG	Assigns a specified value to the location counter

5.2.2.1 SSECT Directive

The SSECT directive determines the location of code and data sequences, and assembles the SYSGEN section of program source code. All symbolic names are assigned relocatable memory addresses relative to the beginning of the SYSGEN section. The total number of COMMON blocks and SYSGEN sections must be less than 254.

Syntax:

Label	Operation	Operand
label	SSECT	sname

sname is an ASCII constant indicating the section where succesive code or data should be positioned. If sname is not specified, the default is in the nonextended code section. The following are valid snames:

Section name	Code/data position
ADP_MPX	Adaption code sequence section
DSECT	DSECT data section
EXT_MPX	Extended MPX-32 code section
ALT_MPX	Alternate extended MPX-32 code section to support the IFBASE/ELSE/ENDIF construct

Usage:

```
RM.17.LPN   EQU    $
            BU     RM17.NOW
            SSECT  EXT_MPX
RM.17.NOW   EQU    $
            GOTO   %SKIP2
            ANOP
```

For an additional example, see the FLG_MPX SSECT Directive Usage section.

5.2.2.2 FLG_MPX SSECT Directive

The FLG_MPX SSECT directive signals the change in state in the object code by identifying and loading the code and data sections. This directive is followed by an ASCII data doubleword that indicates changes in type and status.

Syntax:

Label	Operation	Operand
label	SSECT	FLG_MPX
	DATAD	C'keyword'

keyword is an ASCII doubleword. Only one keyword can be specified for each SSECT FLG_MPX directive. Valid keywords are:

Keyword	Description
BASECODE	Switch instruction set processing mode to the base instruction set. Marks extended code and the start of adaptive code for DEFed entry followed by ASCII label for DEFed entry.

NONBASE Switch instruction set processing mode to nonbase instruction set.

BASENTRY Extended module BL entry point sequence follows. An additional ASCII double word containing the entry point symbolic name follows the BASENTRY data double word.

ENDADAPT Indicates the end to BASENTRY adapter sequence. If, during SYSGEN processing, the entry is not referenced by a nonextended module, SYSGEN eliminates the adaption sequence to conserve memory space.

IFBASE Indicates the beginning of an IFBASE/ELSE/ENDIF statement. Coding following the directive up to the next occurrance of an ELSE directive will be included in the system image if the extended modules addresses another extended module. This directive is used when processing a module in the extended base code object format.

ELSE Divides coding for an IFBASE/ELSE/ENDIF statement. Coding preceeding the directive is used for an extended-to-extended control transfer. Coding following the directive is used for a base to nonbase control transfer.

ENDIF Indicates the conclusion of an IFBASE/ELSE/ENDIF statement

Usage: The following example illustrate the use of the SSECT and FLG_MPX directives:

```
            SSECT     FLG_MPX
            DATAD     C'NONBASE'
            SSECT     ADP_MPX
            OPTR      16
RM17.NWT    EQU       $
            LW        R1,C.REGS
            LFBR      B0,T.MPXBR,R1
            LPSD      RM17LPSD
RM17.WPSD   GEN       8/X'86',5/0,19/W(RM17.LPN)
            DATAW     X'80004000'
            SSECT     FLG_MPX
            DATAD     C'BASECODE'
            OPTS      16
            SSECT     ADP_MPX
```

5.2.2.3 OPTR Directive

The OPTR directive resets option 16, regardless of the option's previous value. It then assigns the new option value to the symbol.

Syntax:

Label	Operation	Operand
symbol	OPTR	16

symbol is optional, and the value assigned to it can be redefined by the OPTR, OPTS, and OPTT directives

Usage: See the OPTS Directive Usage section.

5.2.2.4 OPTS Directive

The OPTS directive sets option 16, regardless of the option's previous value. It then assigns the new option value to the symbol.

Syntax:

Label	Operation	Operand
symbol	OPTS	16

symbol is optional, and the value assigned to it can be redefined by the OPTR, OPTS, and OPTT directives

Usage: The following example illustrates the use of the OPTR and OPTS directives:

```
                SSECT   FLG_MPX
                DATAD   C'NONBASE'
                SSECT   ADP_MPX
                OPTR    16
RM17.NWT        EQU     $
                LW      R1,C.REGS
                LFBR    B0,T.MPXBR,R1
                LPSD    RM17LPSD
RM17.WPSD       GEN     8/X'86',5/0,19/W(RM17.LPN)
                DATAW   X'80004000'
                SSECT   FLG_MPX
                DATAD   C'BASECODE'
                OPTS    16
                SSECT   ADP_MPX
```

5.2.2.5 OPTT Directive

The OPTT directive tests the selected assemble option and assigns a value of one to the symbol if option 16 is set. A value of zero is assigned to symbol if the option 16 is not set.

Syntax:

Label	Operation	Operand
symbol	OPTT	16

symbol is optional and the value assigned to it can be redefined by the OPTR, OPTS, and OPTT directives

5.2.2.6 SDEF Directive

The SDEF directive identifies SYSGEN linkage symbols within a given program which may be referenced by another program or subroutine as entry points or data within the MPX-32 SYSGEN process.

The symbols referenced in the operand field must be defined in the program where the directive is used.

SDEF directives must precede data definitions and executable statements in the source program.

Syntax:	Label	Operation	Operand
	label	SDEF	sym [,sym]

sym is a symbolic name local to the program

Usage: See the SEXT directive description.

5.2.2.7 SEXT Directive

The SEXT directive identifies SYSGEN linkage symbols that are entry points or data in another program or subroutine, but referenced by the given program.

The symbols referenced in the operand field must be defined in a different program than the one where the SEXT directive is used. The symbols are given defined addresses at load time if corresponding SDEF directives in another program or subroutine are present.

Symbols defined by SEXT directives may not be used within a common defintion or in the operand field of the EQU directive.

Syntax:	Label	Operation	Operand
	label	SEXT	sym [,sym]

sym is a symbolic name defined in another program or subroutine

Usage: The following samples of listed output illustrate the use of SEXT and SDEF directives:

REFERENCING PROGRAM

Location Counter	Machine Instruction	Byte Address	Label	Operation	Operand
				PROGRAM	EXTDEF1
				SEXT	CAL4
S00000				SSECT	CAL
S00000			CALS	EQU	$
S00000	D4000018	S00018		STW	0,CAL5R0
S00004	F8800001	X00000		BL	CAL4
S00008	F8800005	X00000		BL	CAL4
S0000C	F8800009	X00000		BL	CAL4
S00010	F880001D	S0001C		BL	CAL2
S00014	EC100019	S00018		BU	*CAL5R0
S00018	00000000		CAL5R0	DATAW	0
S0001C	D4000028	S00028	CAL2	STW	0,CAL2R0
S00020	C9800003			LI	3,3
S00024	EC100029	S00028		BU	*CAL2R0
S00028	00000000		CAL2R0	DATAW	0
P00000				END	

REFERENCED PROGRAM

Location Counter	Machine Instruction	Byte Address	Label	Operation	Operand
				PROGRAM	EXTDEF2
S00000				SDEF	CAL4
S00000			CAL4	EQU	$
S00000	D4000014	S00014		STW	0,CAL4R0
S00004	D5800018	S00018		STW	3,WORD3
S00008	D600001C	S0001C		STW	4,WORD4
S0000C	D6800020	S00020		STW	5,WORD5
S00010	EC100015	S00014		BU	*CAL4R0
S00014	00000000		CAL4R0	DATAW	0
S00018	00000000		WORD3	DATAW	0
S0001C	00000000		WORD4	DATAW	0
S00020	00000000		WORD5	DATAW	0
P00000				END	

5.2.2.8 SORG Directive

The SORG directive assigns the value specified in the operand field to the location counter. Symbolic names are assigned absolute or relocatable values relative to the point of origin until a subsequent ABS, REL, ORG, or SORG directive is encountered.

Syntax:

Label	Operation	Operand
label	SORG	value

value — a previously defined operand that is not an external reference. See Usage for an example.

Usage: The followng example assigns the value 1000 (hexadecimal) to TAGA and START:

```
TAGA    SORG    X'1000'
START   LW      R2,TAGA
```

5.2.3 Options

In addition to the options defined in the Macro Assembler section of the MPX-32 Utilities Manual, the assembler has options for control and macro percentage parameters.

Option	Description
16	Generates extended mode opcodes and instruction formats. This option is declarable internally and in-line with source coding only (i.e., the OPTR and OPTS directives). Option 16 set externally has no affect on the generation of base code.

5.2.4 Errors and Aborts

The following error flag is generated by the assembler:

Error	Description
S	Identifies the usage of indirect addressing when the base register code generation option bit is set

5.3 MPXDB

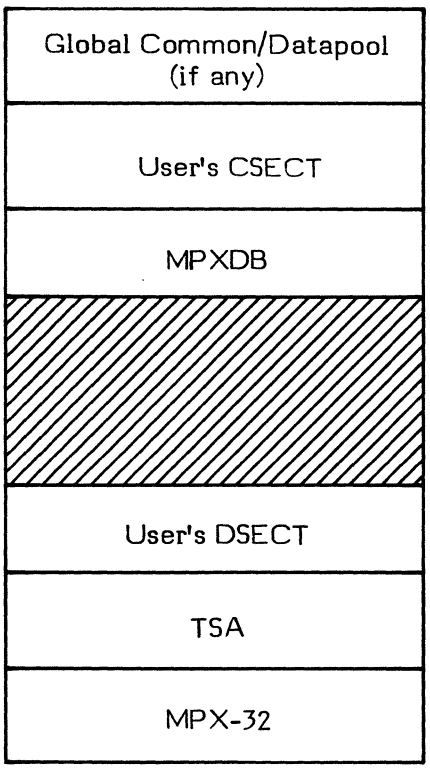
MPXDB functions an unsolicited overlay of a nonbase mode task being debugged. When attached to a user task, it provides a set of commands that the user can use to monitor and control the execution of the task. It is an interactive tool for the on-line terminal user operating under TSM, and can be used to debug a task in batch mode. Every effort has been made to allow the user's normal access to the operating facilities.

5.3.1 The MPXDB Environment

The activation sequence for a nonbase mode task to be run under MPXDB is similar to the normal activation sequence except that:

- The size of the address space constructed is increased by the size of MPXDB.
- Control is given to MPXDB startup entry point instead of the transfer address of the user task.

The address space constructed for MPXDB is as follows:



The MPXDB environment is established for a task by a call to H.REXS,29 (M.DEBUG service). The task can call H.REXS,29 anytime. The TSM DEBUG directive and the JCL $DEBUG directive cause H.REXS,29 to be called as part of the activation sequence for a user task.

Notes:

The combination of MPXDB and the user's code is a single task with a single TSA and a single dispatch queue entry. When MPXDB gains control at its start-up entry point, it makes dynamic assignments for its file codes according to whether it is running on-line or batch. Any dynamic assignments for these file codes made by the user task are prohibited. To minimize conflict with user file codes, all MPXDB file codes begin with the character #.

When MPXDB gains control, whether at activation or upon the occurrence of a trap or abort, it runs privileged, allowing it to replace user instructions with traps. When MPXDB transfers control to the user's task, it restores the privilege state (as cataloged) of the user's task.

5.3.2 Entry Points

MPXDB begins with a Halfword Address Table (HAT) in the following format:

```
DEBUG   DATAW   5
        ACH     DEBUG.1
        ACH     DEBUG.2
        ACH     DEBUG.3
        ACH     DEBUG.4
        ACH     DEBUG.5
```

The entry points have the following functions:

Entry Point	Function
1	Start-up
2	Restart
3	Trap/break receiver
4	Reentry after BREAK directive
5	User abort receiver

5.3.2.1 Entry Point 1 - Start-up

File codes are assigned to the operating mode (on-line or batch), and files are opened. T.CONTXT is saved for a possible RESTART directive. The first immediate command is read from #IN and DEBUG proceeds under control of the command stream.

Entry Conditions

The user PSD in T.CONTXT points to the cataloged transfer address of the user task. The user registers in T.CONTXT all contain zeroes. T.REGP points to T.REGS+0W.

EP1 is called because of a call to H.REXS,29 (M.DEBUG service) by the user task or as part of the activation sequence for the user task.

Exit Conditions

Exit is through any of the H.EXEC calls described in section 5.2.3, or M.EXIT in response to an immediate EXIT directive, or H.REXS,50 described in section 5.2.4.

5.3.2.2 Entry Point 2 - Restart

All files are closed and deallocated. The T.CONTXT image saved at entry point one is copied to T.CONTXT. All defaults are reset; for example, the parameter input file reverts to #IN, SS (single-step) is reset. The trap table and base table are reinitialized. Control is then passed to entry point one.

Entry Conditions

T.CONTXT contents are unpredictable, and are not used. T.REGP points to T.REGS+0W. This entry point is called by H.EXEC,24, which is called by MPXDB in response to an immediate RESTART directive.

Exit Conditions

Exit is through any of the H.EXEC calls described in section 5.2.3, or M.EXIT in response to an immediate EXIT directive, or H.REXS,50 described in section 5.2.4.

5.3.2.3 Entry Point 3 - Trap/Break Receiver

T.CONTXT and the trap table are analyzed to distinguish breaks from traps. For a trap, the count for that trap is incremented by one.

For a conditional trap whose IF expression equals zero, control is passed back to the user task. For conditional traps whose IF expression does not equal zero, a trap report is issued and the IF command is displayed on #OT. For unconditional traps, a trap report is issued on #OT. In either case, MPXDB proceeds under control of the commands in the trap list.

For a break, the parameter input file reverts to #IN, a break report is issued on #OT, and DEBUG reads the next immediate command from the parameter input file.

Entry Conditions

This entry point is called when the user task executes an SVC 1,X'66' (MPXDB trap) instruction or receives a break. T.CONTXT indicates the user context following the execution of the last user instruction. T.REGP, T.REGS, and flags in DQE.ATI allow DEBUG to report the nesting, if any, of push-down levels due to any task interrupts active at the time of the trap or break.

Exit Conditions

Exit is through any of the H.EXEC calls described in section 5.2.3, or M.EXIT in response to an immediate EXIT directive, or H.REXS,50 described in section 5.2.4.

5.3.2.4 Entry Point 4 - M.BRKXIT Receiver

Execution of the user's M.BRKXIT is reported on #OT. MPXDB then reads the next immediate command from the parameter input file.

Entry Conditions

This entry point is called as the result of the user's execution of M.BRKXIT. The user's break receiver is run only as the result of an MPXDB call to H.EXEC,23. T.CONTXT, T.REGS, and T.REGP are the same as they were immediately before MPXDB called H.EXEC,23.

Note: After MPXDB calls H.EXEC,23, if the user task never executes M.BRKXIT, the break receiver push-down level in T.REGS will never be cleared. Each trap or break report on #OT will remind the user of this condition with its push-down analysis.

Exit Conditions

Exit is through any of the H.EXEC calls described in section 5.2.3, or M.EXIT in response to an immediate EXIT directive, or H.REXS,50 described in section 5.2.4.

5.3.2.5 Entry Point 5 - Abort Receiver

A report of the user abort, similar in form to a trap or break report, is displayed on #OT. The abort report includes the abort code message found in the dispatch queue entry. MPXDB then reads the next immediate command from the parameter input file.

Entry Conditions

This entry point is called when the user task encounters an abort condition. T.CONTXT indicates the user context following the execution of the last user instruction. T.REGS, T.REGP, and flags in DQE.ATI allow MPXDB to report the nesting, if any, of push-down levels due to any task interrupts active at the time of the abort. This entry point is never entered while the user task has an abort receiver established (M.SUAR service).

Exit Conditions

Exit is through any of the H.EXEC calls described in section 5.2.3, or M.EXIT in response to an immediate EXIT directive, or H.REXS,50 described in section 5.2.4.

5.3.3 H.EXEC Calls

DEBUG uses three special entry points in H.EXEC for control transfers, as follows:

H.EXEC,22

H.EXEC,22 is called by MPXDB in response to an immediate go or track directive, to begin or continue execution of the user task.

H.EXEC,23

H.EXEC,23 is called in response to an immediate break directive, to pass control to the user's break receiver. When the user task executes M.BRKXIT, control is passed to MPXDB entry point four.

H.EXEC,24

H.EXEC,24 is called in response to an immediate restart directive, to clear any push-down levels in T.REGS and perform related clean-up functions prior to passing control to MPXDB entry point two.

5.3.4 H.REXS Calls

This section describes the H.REXS calls which perform special MPXDB-related functions. MPXDB makes free use of many other H.REXS calls.

H.REXS,29

H.REXS,29 (M.DEBUG) is not called by MPXDB. It is called by a user task, or as part of the activation sequence, to establish the MPXDB environment for a user task.

H.REXS,30

H.REXS,30 is called in response to an immediate kill directive. Its function is to destroy the MPXDB environment previously established by H.REXS,29 (M.DEBUG), leaving the user task intact and transferring control to it at a specified context. In particular, H.REXS,30 makes MPXDB memory available for allocation by the user task.

H.REXS,42

H.REXS,42 (SVC 1,X'66') is the MPXDB trap instruction. It is stored in the user task, replacing the user's instruction, in response to the set, go, and track directives. Execution of SVC 1,X'66' by the user task causes control to pass to MPXDB entry point three after T.CONTXT is loaded with the user context.

H.REXS,50

H.REXS,50 (SVC 1,X'7E') is called by the user program to exit. If MPXDB is associated with the task, entry point five is entered and a user exit message is generated.

5.3.5 File Code Usage

MPXDB has no cataloged assignments. When it gains control at entry point one, it dynamically assigns #IN, #OT, #01 and #04 according to the operating mode, on-line or batch. It assigns #02 and #03 in response to log, dump, file, and store directives. The following table lists the MPXDB file codes and their uses:

#IN Control input (commands)

On-line:	ASSIGN4 #IN = UT
Batch:	ASSIGN2 #IN = SYC (only if not already assigned)

#OT Primary output (displays, trap reports, diagnostics, etc.)

On-line:	ASSIGN4 #OT = UT
Batch:	ASSIGN2 #OT = SLO,10000 (only if not already assigned)

#01 Log file (log of all I/O on UT)

On-line: ASSIGN3 #01 = DC,300
Batch: Not used

#02 SLO files for log and dump directives

On-line: ASSIGN2 #02 = SLO,x (where x is determined by the size of the log or dump being produced)
Batch: Not used

#03 FILE and STORE files

On-line: ASSIGN1 #03 = file (where file is as specified in FILE or STORE command)
Batch: Same as on-line

#04 Patch file (saved CM commands)

On-line: ASSIGN3 #04 = DC,100
Batch: Not used

5.3.6 TSA References

The following TSA areas are referenced by MPXDB:

T.REGS	With DQE.ATI to analyze
T.REGP	User task interrupt status for abort, trap and break reports
T.CONTXT	User task context as of its last executed instruction

5.3.7 Communication Region References

None.

5.3.8 Dispatch Queue Entry (DQE) References

The following areas of the DQE are referenced by MPXDB:

DQE.USHF	To determine operating mode, on-line or batch
DQE.ATI	Together with T.REGS and T.REGP to analyze user task interrupt status for abort, trap, and break reports

CHAPTER 6

SYSTEM TRACE

System Trace is an MPX-32 debug facility designed to assist in determining the causes of system crashes. System events are recorded circularly in a trace table which can be dumped in the event of a system crash.

Thirty trace event types are available for recording as follows:

1	Task activation
2	Task termination
3	Dispatch CPU to task
4	Task relinquishes CPU
5	Queue I/O
6	End I/O
7	Interrupt/trap handler entry
8	Interrupt/trap handler exit
9	M.SHUT
10	M.OPEN
11	M.IOFF or BEI
12	M.IONN or UEI
13	M.CALL
14	SVC Type 1
15	M.RTRN or M.RTNA
16	Inswap Task
17	Outswap Task
18	Dispatch IPU Task
19	Relinquish IPU Task
20	CALM
21-22	Mobile Event Trace
23	SVC Type 15
24	SVC Type 2
25-30	Reserved

Occurrences of trace events are signaled by SVC instructions. These SVCs are generated by the expanded BEI and UEI macros for trace types eleven and twelve, and by the expanded M.TRAC macro for all other trace types. The M.TRAC macro is implanted within MPX-32 as required by each trace type. SVC types X'A', X'B', X'C', and X'D' are used by the System Trace event recorder.

The System Trace event recorder is an SVC processor which is assembled within H.IP06. Recorded events each use an eight-word entry in the trace table which occupies memory from absolute locations 78000 to 7FFFF. The table is circular in that when the last entry in the table is used, the first entry is reused to record the next event.

System Trace event recording is controlled by flags in a word in the communications region, C.TRACE. Bit zero of C.TRACE controls all trace types. If this bit is set, no tracing is performed. If this bit is not set, tracing is controlled by bits one through 30 of C.TRACE. Bits one through 30 correspond to trace types one through 30, respectively, and may be set to turn each trace type off. Bit 31 of C.TRACE is reserved as an indicator that the trace table recording control words have been initialized by the event recorder. If the value of this bit is not disturbed, recording is continuous; for example, the control words are not reset. If the value of this bit is set to zero, the event recorder initializes the recording control words to indicate that the trace table is empty.

The first eight words of the trace table are reserved for control information. The first word contains the absolute memory address within the trace table at which the last trace entry was stored. Bit zero of the second word is a wraparound indicator. This bit is set if wraparound from the last to first trace table entry has occurred.

The trace table dump routine is incorporated in resident MPX-32. This routine formats each trace table entry and writes it to the line printer. The routine is controlled by the contents of C.TRACD which is equated to absolute memory location 78008. Bits one through 15 of C.TRACD correspond to trace types one through 15, respectively, and may be set to inhibit printing of any trace types. Bits sixteen through 31 of C.TRACD may be used to limit the number of trace table entries eligible for printing. If this field contains zero, all trace table entries are eligible for printing. If this field contains a nonzero value, this is the number of most recently recorded entries eligible for printing.

To use the trace table dump routine, the contents of C.TRACD should be set as desired and control transferred to the routine at its entry point. The symbol TRACDUMP is associated with the entry point via a DEF. The routine must be entered unmapped. After the dump is completed, the routine halts. Each trace type is detailed next. Fields in printout formats are underlined to indicate actual values. Numeric hexadecimal fields are indicated by "x". Numeric decimal fields are indicated by "d". The printout of each trace table entry occupies one printer line.

6.1 Trace Type 1 - Task Activation

Macro:

```
M.TRAC    1

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',1
```

Implanted in H.ALOC so that the address of the task's dispatch queue entry is contained in register seven to be returned and may be obtained as follows:

```
LW        R2,C.TSAD
LW        R2,T.REGP,R2
LW        R2,7W,R2
```

Trace Table Entry

Word 0	Type 01	TSA address (DQE.TAD)
1	Interrupt counter (C.INTC)	
2	Load module name (DQE.LMN)	
3		
4	Owner name (DQE.ON)	
5		
6	DQE entry # (DQE.NUM)	Task activation sequence number (DQE.TAN)
7	Scheduling flags (DQE.USHF)	

Printout:

```
C.INTC    ACTIVATE TASK = xxxxxxxx  DQE.LMN DQE.ON

          DQE.USHF = xxxxxxxx  DQE.TAD = xxxxxxxx
```

6.2 Trace Type 2 - Task Termination

Macro:

```
M.TRAC    2

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',2
```

Implanted in H.EXEC so that C.CURR contains the address of the task's dispatch queue entry number.

Trace Table Entry

Word 0	Type 02	TSA address (DQE.TAD)
1	Interrupt counter (C.INTC)	
2	Load module name (DQE.LMN)	
3		
4	Owner name (DQE.ON)	
5		
6	DQE entry # (DQE.NUM)	Task activation sequence number (DQE.TAN)
7	Scheduling flags (DQE.USHF)	

Printout:

```
C.INTC    TERMINATE TASK = xxxxxxxx  DQE.LMN  DQE.ON

          DQE.USHF = xxxxxxxx  DQE.TAD = xxxxxxxx
```

6.3 Trace Type 3 - Dispatch CPU to Task

Macro:

```
M.TRAC    3

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',3
```

Implanted in H.EXEC so that register two contains the address of the task's dispatch queue entry number.

Trace Table Entry

Word 0	Type 03	TSA address (DQE.TAD)
1	Interrupt counter (C.INTC)	
2	Load module name (DQE.LMN)	
3		
4	Owner name (DQE.ON)	
5		
6	DQE entry # (DQE.NUM)	Task activation sequence number (DQE.TAN)
7	Scheduling flags (DQE.USHF)	

Printout:

```
C.INTC    DISPATCH TASK = xxxxxxxx  DQE.LMN  DQE.ON

          DQE.USHF = xxxxxxxx  DQE.TAD = xxxxxxxx
```

6.4 Trace Type 4 - Task Relinquishes CPU

Macro:

```
M.TRAC    4

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',4
```

Implanted in H.EXEC so that register two contains the address of the task's dispatch queue entry number.

Trace Table Entry

Word 0	Type 04	TSA address (DQE.TAD)
1	Interrupt counter (C.INTC)	
2	Load module name (DQE.LMN)	
3		
4	Owner name (DQE.ON)	
5		
6	DQE entry # (DQE.NUM)	Task activation sequence number (DQE.TAN)
7	Scheduling flags (DQE.USHF)	

Printout:

```
C.INTC    RELINQ TASK = xxxxxxxx  DQE.LMN  DQE.ON

          DQE.USHF = xxxxxxxx  DQE.TAD = xxxxxxxx
```

6.5 Trace Type 5 - Queue I/O

Macro:

```
M.TRAC    5

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',5
```

Implanted in H.IOCS so that register three contains the I/O queue entry address.

Trace Table Entry

Word 0	Type 05	FCB or TCPB address (IOQ.FCBA)	
1	Interrupt counter (C.INTC)		
2	Handler function word 1 (IOQ.FCT1)		
3	Handler function word 2 (IOQ.FCT2)		
4	Handler function word 3 (IOQ.FCT3)		
5	32-bit flag word (IOQ.FLGS)		
6	Task activation sequence number (C.TSKN)		
7	Channel # (IOQ.CHNO)	Subaddress (IOQ.SUBA)	

Printout:

```
C.INTC    QUE I/O TASK = xxxxxxxx  DEV=xxxx

          IOQ.FLGS = xxxxxxxx  FN WDS = xxxxxxxx xxxxxxxx xxxxxxxx

          FCB = xxxxxxxx
```

6.6 Trace Type 6 - End I/O

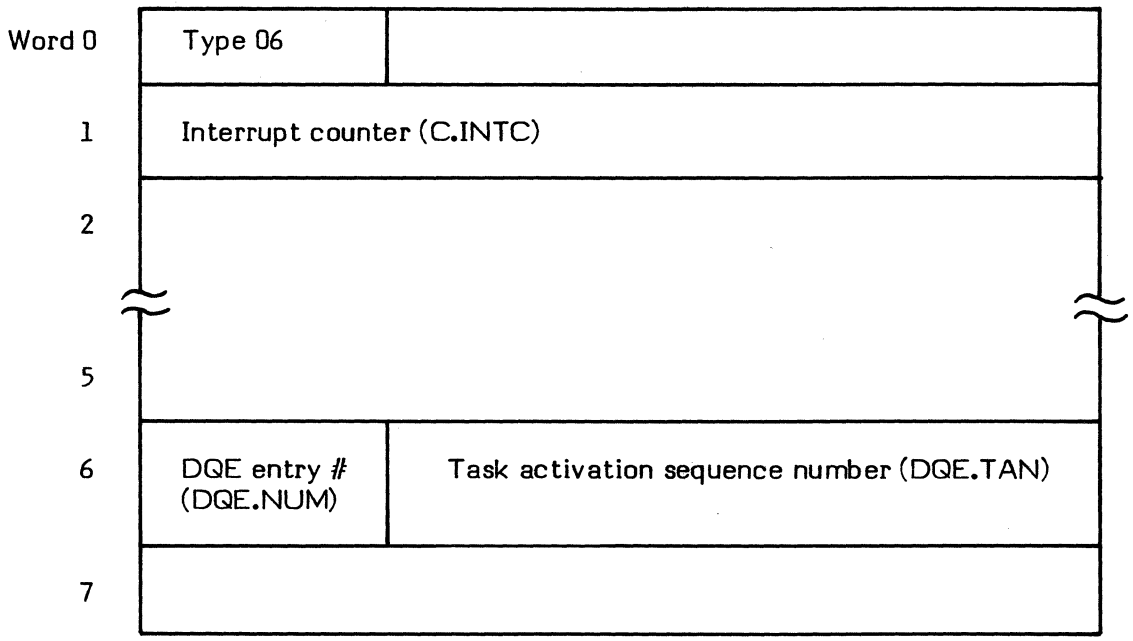
Macro:

```
M.TRAC    6

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',6
```

Implanted in H.EXEC (S.EXEC1, S.EXEC2, S.EXEC3 and S.EXEC4) so that register one contains the task's dispatch queue entry number.

Trace Table Entry



Printout:

```
C.INTC    END I/O TASK = xxxxxxxx
```

6.7 Trace Type 7 - Interrupt/Trap Handler Entry

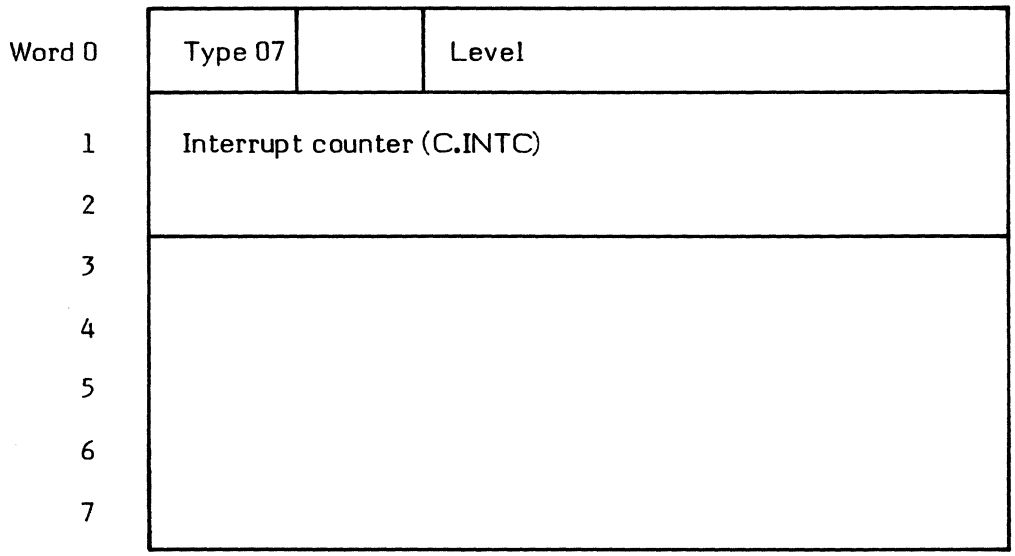
Macro:

```
M.TRAC    7,level

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'B',X'level'
```

Implanted in interrupt/trap handler.

Trace Table Entry



Printout:

```
C.INTC    ENTERINT xxxx
```

6.8 Trace Type 8 - Interrupt/Trap Handler Exit

Macro:

```
M.TRAC    8,level

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'C',X 'level'
```

Implanted in the interrupt/trap handler.

Trace Table Entry

Word 0	Type 08		Level
1	Interrupt counter (C.INTC)		
2			
3			
4			
5			
6			
7			

Printout:

```
C.INTC    EXITINT xxxx
```

6.9 Trace Type 9 - M.SHUT

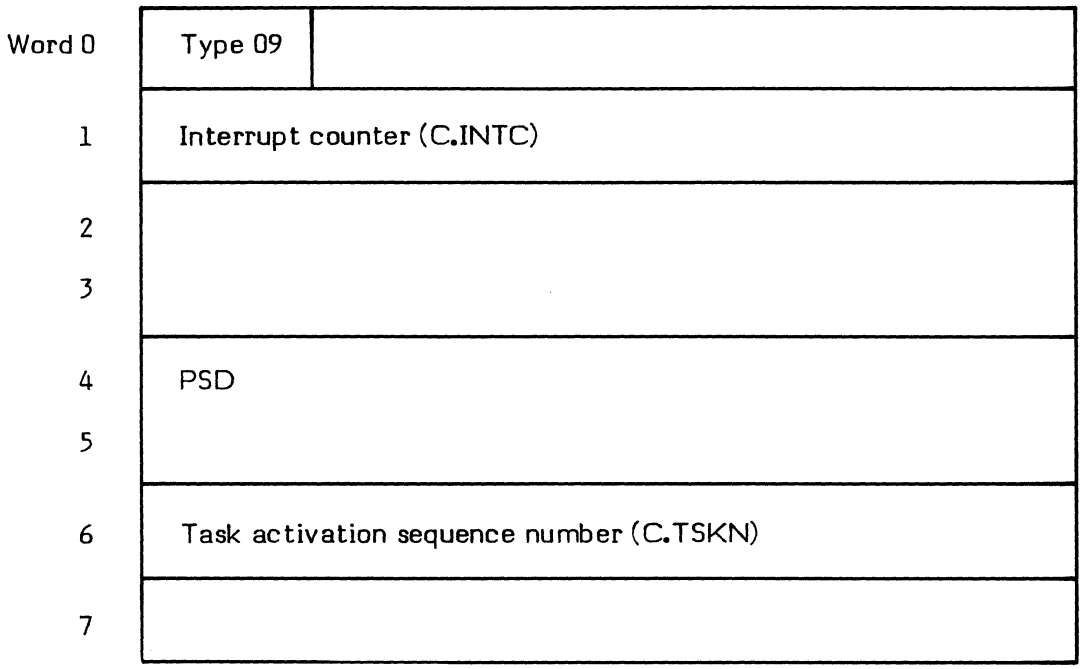
Macro:

```
M.TRAC    9

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',9
```

Implanted in M.SHUT macro.

Trace Table Entry



Printout:

```
C.INTC    M.SHUT TASK = xxxxxxxx  PSD = xxxxxxxx xxxxxxxx
```

6.10 Trace Type 10 - M.OPEN

Macro:

```
M.TRAC    10

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',10
```

Implanted in M.OPEN macro.

Trace Table Entry

Word 0	Type 10	
1	Interrupt counter (C.INTC)	
2		
3		
4	PSD	
5		
6	Task activation sequence number (C.TSKN)	
7		

Printout:

```
C.INTC      M.OPEN TASK = xxxxxxxx  PSD = xxxxxxxx  xxxxxxxx
```

6.11 Trace Type 11 - M.IOFF or BEI

Implemented by BEI macro whose prototype is as follows:

```
BEI   DEFM
      TBM       0,C.TRACE
      BCT       1,$+3W
      DATAW     X'00060002'
      SVC       X'A',11
      ENDM
```

Trace Table Entry

Word 0	Type 11	
1	Interrupt counter (C.INTC)	
2		
3		
4	PSD	
5		
6	Task activation sequence number (C.TSKN)	
7		

Printout:

C.INTC M.IOFF TASK = xxxxxxxx PSD=xxxxxxxx xxxxxxxx

6.12 Trace Type 12 - M.IONN or UEI

Implemented by UEI macro whose prototype is as follows:

```
UEI   DEFM
      TBM       0,C.TRACE
      BCT       1,$+3W
      DATAW     X'00070002'
      SVC       X'A',12
      ENDM
```

Trace Table Entry

Word 0	Type 12	
1	Interrupt counter (C.INTC)	
2		
3		
4	PSD	
5		
6	Task activation sequence number (C.TSKN)	
7		

Printout:

C.INTC M.IONN TASK = xxxxxxxx PSD= xxxxxxxx xxxxxxxx

6.13 Trace Type 13 - M.CALL

Macro:

```
M.TRAC    13

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',13
```

Implanted in H.IP06.

Trace Table Entry

Word 0	Type 13		Bits 20-31 of the SVC
1	Interrupt counter (C.INTC)		
2			
3			
4	PSD		
5			
6	Task activation sequence number (C.TSKN)		
7	Stack frame pointer (T.REGP)		

Printout:

```
C.INTC    M.CALL TASK = xxxxxxxx PSD=xxxxxxxx xxxxxxxx
          MODULE = name,dd
```

6.14 Trace Type 14 - SVC Type 1

Macro:

```
M.TRAC    14

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',14
```

Implanted in H.IP06.

Trace Table Entry

Word 0	Type 14		Bits 20-31 of the SVC
1	Interrupt counter (C.INTC)		
2			
3			
4	PSD		
5			
6	Task activation sequence number (C.TSKN)		
7	Stack frame pointer (T.REGP)		

Printout:

```
C.INTC    SVC1TASK = xxxxxxxx  PSD=xxxxxxxx  xxxxxxxx
          SVC=dddd
```

6.15 Trace Type 15 - M.RTRN or M.RTNA

Macro:

```
M.TRAC    15

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',15
```

Implanted in M.RTRN and M.RTNA macros.

Trace Table Entry

Word 0	Type 15	
1	Interrupt counter (C.INTC)	
2		
3		
4	PSD	
5		
6	Task activation sequence number (C.TSKN)	
7	Stack frame pointer (T.REGP)	

Printout:

```
C.INTC    M.RTRN/A TASK = xxxxxxxx  PSD = xxxxxxxx  xxxxxxxx
```

6.16 Trace Type 16 - Inswap Task

Macro:

```
M.TRAC    16

TBM       0.C.TRACE
BCT       1,$+2W
SVC       X'A',16
```

Implanted in H.IP06.

Trace Table Entry

Word 0	Type 16	TSA address (DQE.TAD)
1	Interrupt counter (C.INTC)	
2	Load module name (DQE.LMN)	
3		
4	Owner name (DQE.ON)	
5		
6	DQE entry # (DQE.NUM)	Task activation sequence number (DQE.TAN)
7	Scheduling flags (DQE.USHF)	

Printout:

```
C.INTC    INSWAP TASK = xxxxxxxx DQE.LMN DQE.ON

          DQE.USHF = xxxxxxxx DQE.TAD = xxxxxxxx
```

6.17 Trace Type 17 - Outswap Task

Macro:

```
M.TRAC    17

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',17
```

Implanted in IP06.

Trace Table Entry

Word 0	Type 17	TSA address (DQE.TAD)
1	Interrupt counter (C.INTC)	
2	Load module name (DQE.LMN)	
3		
4	Owner name (DQE.ON)	
5		
6	DQE entry # (DQE.NUM)	Task activation sequence number (DQE.TAN)
7	Scheduling flags (DQE.USHF)	

Printout:

```
C.INTC    OUTSWAP TASK = xxxxxxxx DQE.LMN DQE.ON

          DQE.USHF = xxxxxxxx DQE.TAD = xxxxxxxx
```

6.18 Trace Type 18 - Dispatch IPU Task

Macro:

```
M.TRAC    18

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',18
```

Implanted in H.CPU so that register two contains the address of the dispatch queue entry address.

Trace Table Entry

Word 0	Type 18	TSA address (DQE.TAD)
1	Interrupt counter (C.INTC)	
2	Load module name (DQE.LMN)	
3		
4	Owner name (DQE.ON)	
5		
6	DQE entry # (DQE.NUM)	Task activation sequence number (DQE.TAN)
7	Schedule flags (DQE.USHF)	

Printout:

```
C.INTC    DISP IPU TASK = xxxxxxxx DQE.LMN DQE.ON

          DQE.USHF = xxxxxxxx DQE.TAD = xxxxxxxx
```

6.19 Trace Type 19 - Relinquish IPU Task

Macro:

```
M.TRAC    19

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',19
```

Implanted in H.CPU so that register two contains the address of the dispatch queue entry address.

Trace Table Entry

Word 0	Type 19	TSA address (DQE.TAD)
1	Interrupt counter (C.INTC)	
2	Load module name (DQE.LMN)	
3		
4	Owner name (DQE.ON)	
5		
6	DQE entry # (DQE.NUM)	Task activation sequence number (DQE.TAN)
7	Schedule flags (DQE.USHF)	

Printout:

```
C.INTC    RELINQ IPU TASK = xxxxxxxx DQE.LMN DQE.ON

          DQE.USHF = xxxxxxxx DQE.TAD = xxxxxxxx
```

6.20 Trace Type 20 - Reserved

6.21 Trace Type 21 - Mobile Event Trace 1

Implanted by the system debugger by the ET command.

Trace Table Entry

Word 0	Type 21	Stack frame address (T.REGP)
1	Interrupt counter (C.INTC)	
2	DQE schedule flags (DQE.USHF)	
3	Requested task interrupts (DQE.RTI)	Active task interrupts (DQE.ATI)
4	PSD	
5		
6	DQE entry # (DQE.NUM)	Task activation sequence number (DQE.TAN)
7	Swap inhibit flags (DQE.SWIF)	System action interrupt requests (DQE.SAIR)

Printout:

C.INTC ET #1 TASK = xxxxxxxx PSD = xxxxxxxx xxxxxxxx

USHF = xxxxxxxx RTI/ATI = xxxxxxxx T.REGP = xxxxxxxx

SWIF = xx SAIR = xx

6.22 Trace Type 22 - Mobile Event Trace 2

Implanted by the system debugger by the ET command.

Trace Table Entry:

Word 0	Type 22	Contents of GPR 0
1	Contents of GPR 1	
2	Contents of GPR 2	
3	Contents of GPR 3	
4	Contents of GPR 4	
5	Contents of GPR 5	
6	Contents of GPR 6	
7	Contents of GPR 7	

Printout:

R0 = xxxxxxxx R1 = xxxxxxxx R2 = xxxxxxxx R3 = xxxxxxxx

R4 = xxxxxxxx R5 = xxxxxxxx R6 = xxxxxxxx R7 = xxxxxxxx

6.23 Trace Type 23 - SVC Type 15

Macro:

```
M.TRAC    23

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',23
```

Implanted in H.IP06.

Trace Table Entry

Word 0	Type 23		Bits 20-31 of the SVC
1	Interrupt counter (C.INTC)		
2			
3			
4	PSD		
5			
6	Task activation sequence number (C.TSKN)		
7	Stack frame pointer (T.REGP)		

Printout:

```
C.INTC    SVC15 TASK = xxxxxxxx PSD=xxxxxxxx xxxxxxxx
          SVC=dddd
```

6.24 Trace Type 24 - SVC Type 2

Macro:

```
M.TRAC    24

TBM       0,C.TRACE
BCT       1,$+2W
SVC       X'A',24
```

Implanted in H.I06.

Trace Table Entry

Word 0	Type 24		Bits 20-31 of the SVC
1	Interrupt counter (C.INTC)		
2			
3			
4	PSD		
5			
6	Task activation sequence number (C.TSKN)		
7	Stack frame pointer (T.REGP)		

Printout:

```
C.INTC    SVC2 TASK = xxxxxxxx PSD=xxxxxxxx xxxxxxxx
          SVC=dddd
```

CHAPTER 7

SYSTEM INITIALIZERS AND BUILDERS

This chapter describes the components of MPX-32 that are responsible for loading and initializing the system when it is booted. The components involved are:

- SDT loader contained within the Volume Manager
- SYSINIT, J.INIT, J.TINIT, and RESTART system tasks

A system boot can be performed in various operating environments and involve various system devices. In a given environment, all of the above components or a subset can be required.

Figures 7-1 through 7-3 describe the components which come into play when the system is booted from a Software Distribution Tape (SDT), from the IOP console, and from an online RESTART directive.

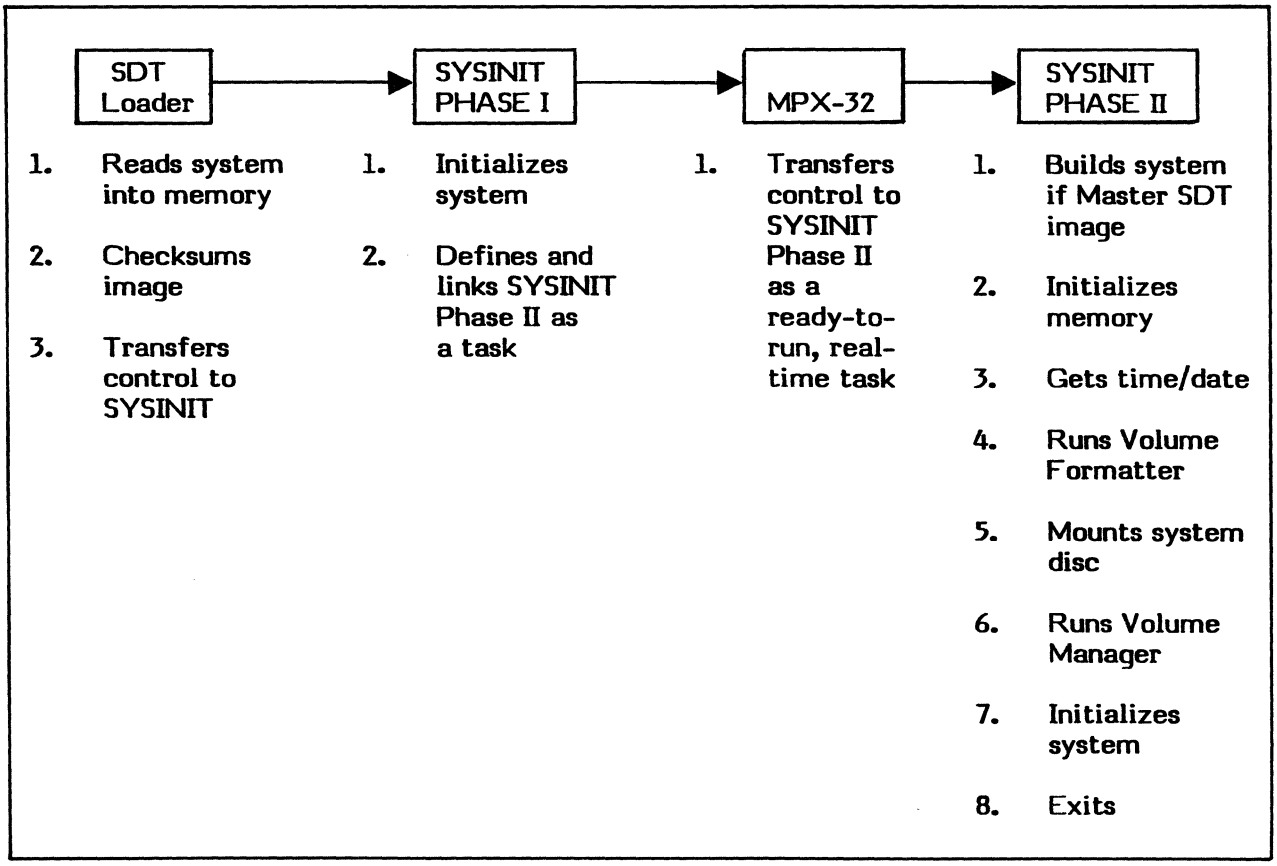


Figure 7-1.
Components and Functions in Boot from an SDT

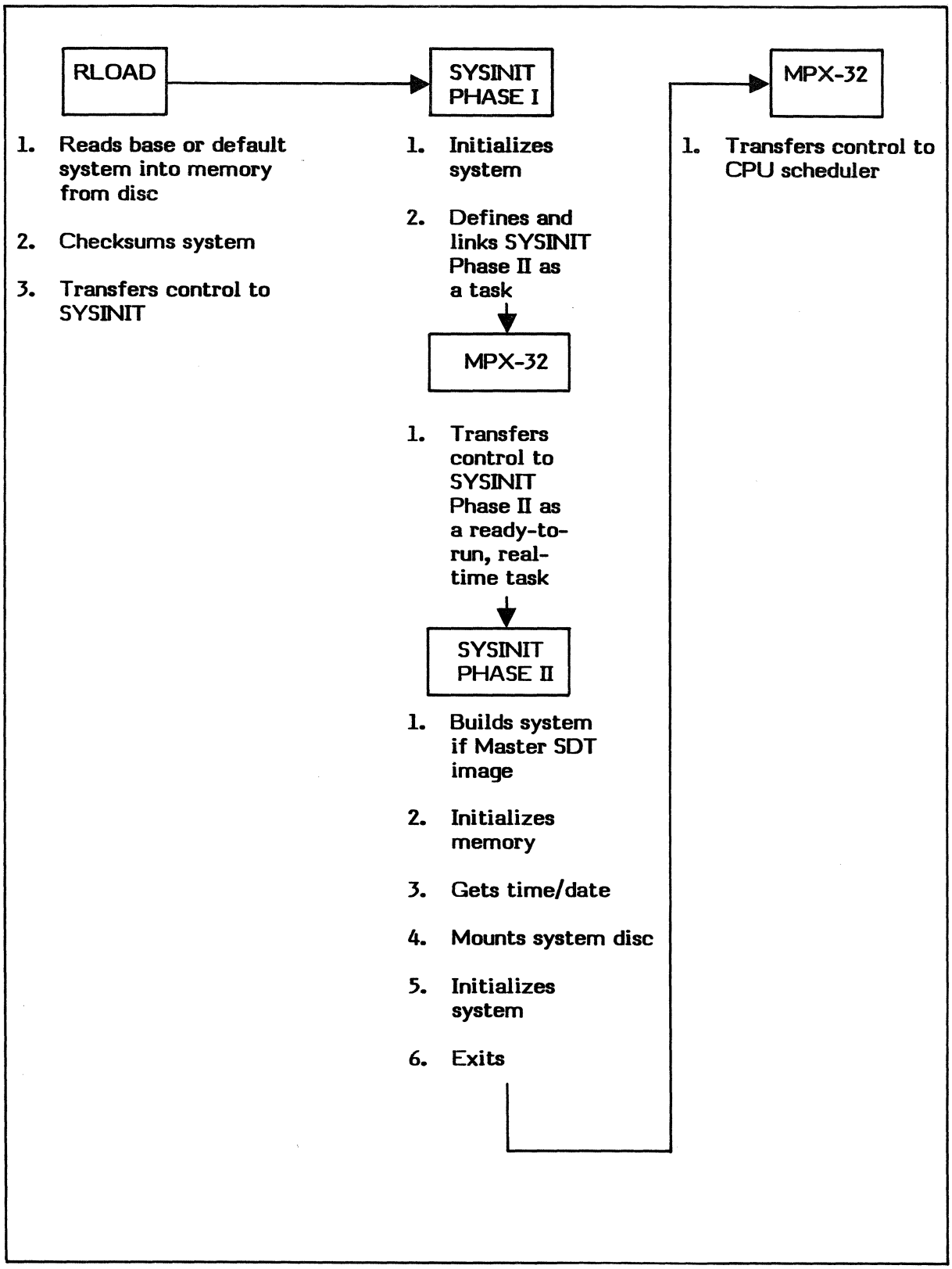


Figure 7-2
Components and Functions in Boot from IOP Console

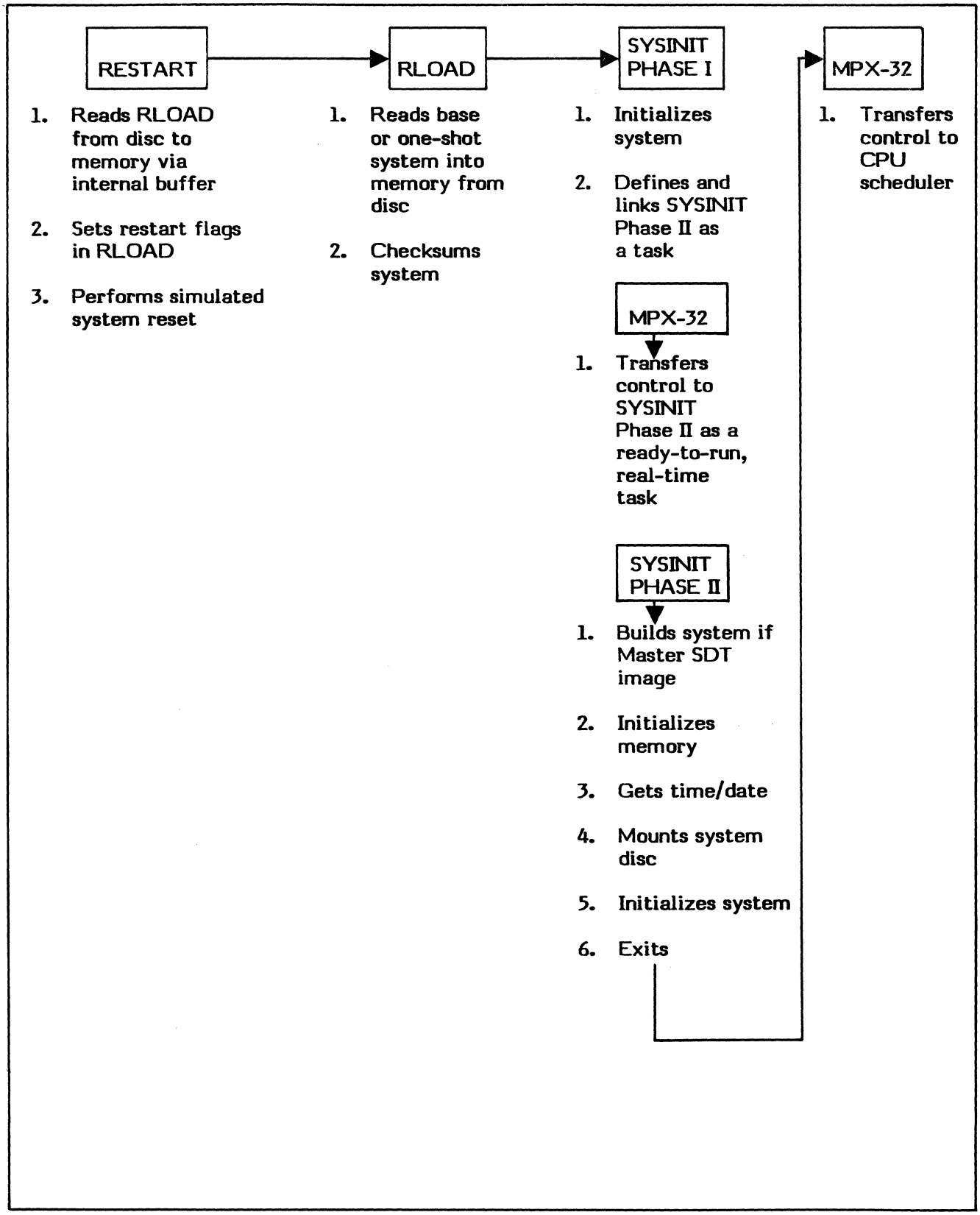


Figure 7-3
Components and Functions in Boot from On-line RESTART

7.1 SDT Loader

The SDT loader is a section of code written to a System Distribution Tape (SDT) by the portion of the Volume Manager that processes the SDT directive. Its purpose is to read the system image from the SDT into memory. Control is then transferred to the MPX-32 initialization program SYSINIT which builds a functional system.

7.1.1 Activating

The SDT loader is activated from the SDT by the IPL = devaddr (device address) console command. The loader code must be the first piece of information contained on the SDT.

7.1.2 Required Input

The SDT loader requires no input. However, the starting load point may be altered by supplying a value other than zero in register three prior to depressing the IPL button. The load point defaults to X'780'.

7.1.3 Processing

The SDT loader is written to tape as absolute code by supplying the SDT directive to the Volume Manager. The loader code is then followed by the system image whose name is supplied in the directive.

When an IPL is performed, firmware reads the SDT loader from the IPL medium into memory, starting at absolute address zero. Control is then transferred to location zero. The loader then reads the system image into memory from tape or floppy disc sequentially starting at the load point specified in register three or the default load point if register three is zero.

The image is read in 192W blocks by means of a buffer reserved in low memory. The load module preamble is read first and the appropriate information is extracted from it. Then the Resource Requirement Summary (RRS) block of the preamble is ignored, and the remainder of the image is loaded into memory.

A checksum is performed while loading the image. The checksum is compared with the checksum value supplied in the preamble. If the checksum value does not agree, the loader halts execution.

After loading the system image, the loader transfers control to SYSINIT Phase I to allow initialization of the operating system to proceed.

7.1.4 Results

Loading has commenced at the appropriate load point, with all memory locations below the load point undefined. For a normal boot from a System Distribution Tape (SDT), memory resident code consists of the version of MPX-32 specified in the SDT directive, immediately followed by the SYSINIT load module. From this point, control is transferred to the SYSINIT entry point.

7.2 The DBOOT Program Section

7.2.1 Activating

The system disc bootstrap, DBOOT, is placed at sector zero of all system discs by the Volume Formatter. This enables booting from any properly formatted disc.

To boot the computer, the console is entered in panel mode and given the command IPL = xxxx, where xxxx is the channel and subaddress of the system disc.

The CPU firmware reads 120 bytes from sector zero of the disc.

The IOCL starting at location eight is executed. When this IOCL terminates, the Program Status Doubleword (PSD) is loaded with the contents of locations zero and four and I/O status is placed back into locations zero and four.

7.2.2 Processing

The bootstrap code loads the operating system, checksums it, and transfers control to SYSINIT Phase I to complete the initialization of the MPX-32 system.

7.2.2.1 IPL

When an IPL is performed, 120 bytes from sector zero are transferred to memory location zero. The CPU then issues a Start I/O (SIO) on the IOCL residing at location eight. This IOCL, contained in the disc bootstrap code, reads the remainder of the boot code into memory and then reads a section of the volume descriptor. This section begins with an IOCL to load the operating system. This IOCL is read so that it begins in the memory location following the IOCL that is reading it. This causes it to be executed as well since the last IOCD in the original chain has its command chain bit set. As a result, when the IPL sequence is over and control is transferred to software level, the entire operating system has been read into memory and is ready for checksumming.

7.3 The SYSINIT Program Section

7.3.1 Activating

SYSINIT is comprised of two sections: Phase I and Phase II. Phase I is entered by either the SDT loader or the disc bootstrap loader and runs as stand-alone code. Phase II is actually the first MPX-32 task and is entered by the MPX-32 execution scheduler as a result of the first real-time clock interrupt.

The mechanism for transfer of control from either loader to Phase I is an indirect branch through C.REGS. This is made possible by SYSGEN loading SYSINIT as the last part of the resident system image, setting C.REGS to the address of the SYSINIT TSA, and then setting the indirect bit in C.REGS. SYSINIT, having a hand built TSA, has its entry point address as the first word of its TSA, thus enabling the transfer of control.

7.3.2 Processing

Phase I of SYSINIT runs as stand alone code because, when it is entered, the hardware and the software are not ready for OS operation. Phase I performs the required hardware initialization functions while Phase II performs the software initialization functions.

The hardware initialization functions performed by Phase I are as follows:

1. Load CPU scratchpad from the image built by SYSGEN.
2. Update memory allocation tables to reflect the memory occupied by SYSINIT.
3. Build a dispatch queue entry for SYSINIT and link it on the ready-to-run state queue.
4. Set up the interrupt vector locations.
5. Determine the Unit Definition Table (UDT) address of the IPL device for all start-ups except for a Master SDT boot. See section on booting from a Master SDT for details.
6. Enable all peripheral interrupt levels.
7. Enable all software interrupt levels.
8. Enable traps.
9. Set the CPU mode.
10. Set mapped mode, unblock interrupts, and wait for a clock interrupt.

The software initialization functions performed by Phase II are as follows:

1. Memory initialization.
2. Request date and time to be entered.
3. Disc start-up initialization.
4. User SDT start-up initialization.
5. Master SDT start-up initialization.

7.3.2.1 Memory Initialization

The first function performed by SYSINIT on a nonmaster SDT boot is the initialization of memory. This clears parity errors which occur in MOS memory after power up, locates nonpresent sections of physical memory, and locates defective memory modules.

The initializer revectors parity and nonpresent memory traps to point to handlers within SYSINIT. Then, for MOS memory, every location is read, and if it is not in a multiprocessor shared section of memory, written back. This clears all potential parity errors. If a parity error is detected on the read, a test data pattern is written to the location and then read back. If another parity error or a mismatched data pattern is detected, the module is marked as malfunctioning and the testing continues. If a nonpresent memory trap is encountered on a read, the module is flagged as nonpresent and testing continues.

For core memory, one location is checked in each module to determine if the module is present.

At the end of the routine, the traps are reset to point to their normal handlers and SYSINIT continues software initialization.

7.3.2.2 System Date and Time

This routine prompts the user to enter the date and time for use by the system. See the MPX-32 Reference Manual Volume III, Chapter 2 for valid entry formats.

7.3.2.3 Disc Start-up Final Initialization

For disc start-up, the system volume must be mounted. SYSINIT automatically allocates the volume, reads the volume descriptor, and builds a Mounted Volume Table entry (MVTE). This allows a call to be made to the mount service H.REMM,17 which causes a run request to be sent to J.MOUNT (the system nonresident media mounting program) which actually performs the mount. See Section 3.3 for details.

Next, J.SWAPR is built into the system by SYSGEN. Its SYSGEN initialization entry point links its own dispatch queue entry to the suspend queue. Execution is held until SYSINIT finishes its functions, then SYSINIT issues a resume request for J.SWAPR.

Once the swapper is running, SYSINIT completes system initialization by activating the following sequence of tasks:

1. J.INIT - installs system patches, mounts default public volumes, and loads ACS.
2. J.EXPAND - enters the any-wait queue and expands the swap file when resumed by J.SWAPR.
3. J.TINIT - initializes user terminals.
4. J.TSM - builds environment for interactive users.
5. J.TDEFI - initializes the TERMDEF facility, if present.
6. User sequentially run tasks, see SYSGEN SEQUENCE directive.
7. User activate table, see SYSGEN ACTIVATE directive.

SYSINIT then exits.

7.3.2.4 Tape Start-up Final Initialization

For tape boots, it is assumed there is no information on the disc to become the system volume. As a result, all tasks needed to build the disc environment are included on the System Distribution Tape (SDT). These tasks are:

1. Volume Formatter (J.VFMT) - This task builds the maps and data structures needed for file maintenance, places a disc bootstrap at sector zero, and writes a copy of the operating system to the desired disc. This assumes an empty disc. J.VFMT can also be used to replace an operating system image only, leaving other data intact, thus giving the user warm start capability. See MPX-32 Reference Manual Volume III, Chapter 13.
2. Mount Program (J.MOUNT) - This task is activated to mount the formatted volume created by J.VFMT.
3. Volume Manager Program (VOLMGR) - This program is activated so files in the file save area following the system information on an SDT can be restored.

After all tasks on the tape have been activated, J.SWAPR is invoked. Once the swapper is running, SYSINIT completes system initialization by activating the following sequence of tasks:

1. J.INIT - installs system patches, mounts default public volumes, and loads ACS.
2. J.EXPAND - enters the any-wait queue and expands the swap file when resumed by J.SWAPR.
3. J.TINIT - initializes user terminals.
4. J.TSM - builds environment for interactive users.
5. User sequentially run tasks, see SYSGEN SEQUENCE directive.
6. User activate table, see SYSGEN ACTIVATE directive.

SYSINIT then exits.

7.3.2.5 Master SDT

The Master System Distribution Tape for MPX-32 contains three system images, one each for the 32/27, 32/7x, and 32/87.

The format of the Master SDT is shown below and described in the paragraphs which follow.

BOOT	IMAGE 32/27	EOF	IMAGE 32/7X	EOF	IMAGE 32/87	EOF	J.VFMT	EOF	J.MOUNT, J.SWAPR, VOLMGR	EOF	EOF	SAVED FILES

7.3.2.5.1 Tape Boot Loader

The first record on an SDT is the boot loader. This code is contained within the Volume Manager and written to the tape as a result of the SDT command. The boot loader first determines the type of machine it is executing on by reading the CPU status word. It then sets up the CPU scratchpad RAM to allow the IPL device to be read as logical device X'1000'.

Two flags are set inside the bootstrap code by the Volume Manager. One specifies the IPL device is a floppy disc. The other indicates a master, as opposed to a user, SDT format.

If the flag indicating Master SDT is set, the boot loader uses the machine type to determine if it is necessary to set command chain bits in the initialization IOCD list. These chained commands are one or two skipfile commands used to advance the tape to the proper system image for the CPU type being IPLed. Image 32/27 is for the 32/27 computer, Image 32/7X is a null image, and Image 32/87 is for all other CONCEPT/32 computers.

Once the tape has been positioned, the image is read into memory. Control then passes to SYSINIT, and system initialization begins.

7.3.2.5.2 SYSINIT - Phase I Initialization

Phase I of SYSINIT prepares the MPX-32 environment. This allows the Phase II portions of SYSINIT to run as an MPX-32 task; therefore, system service calls can be used to perform I/O rather than stand-alone I/O routines.

The first functions SYSINIT performs are reading the IPL device and data return transfer response locations in scratchpad, loading scratchpad from the C.SPAD area created by SYSGEN, initializing the interrupt vector area in low memory, allocating SYSINIT's memory space in the MPX-32 memory tables, and building and linking a CPU dispatch queue entry for SYSINIT. If booting from a disc, SYSINIT may abort. See the MPX-32 Reference Manual, Volume III, Chapter 6.

If the machine is a 32/67 or 32/97, the shared memory region's high and low bounds are set in C.SHRHI and C.SHRLO.

After these operations, SYSINIT compares the doubleword value at C.SYSTEM with the four names reserved for Master SDT starter systems. If a match is found, a flag bit is set and a skip count control word is loaded to allow SYSINIT to correctly process a Master SDT IPL sequence for the current machine type. The skip count word is used to determine where the next desired information resides on the SDT.

The four reserved system names are:

MSTR.27	-	Master system image for a 32/27
FLOP.SYS	-	Master system image contained on a floppy disc instead of tape
MSTR.75	-	Null image
MSTR.87	-	Master system image for all other CONCEPT/32 computers

The final activity in Phase I of SYSINIT is enabling all interrupts and then unblocking them. This allows the clock interrupt to occur, causing a context switch to SYSINIT Phase II, the task portion of SYSINIT.

7.3.2.5.3 SYSINIT - Phase II Initialization

Phase II of SYSINIT checks for a Master SDT boot. This is indicated by the flag set in Phase I. If set, the proper peripheral configuration must be set up to allow further initialization.

The Master SDT system images are SYSGENed in a special way. The following controller and device entries are included:

```
/CHANNELS
CONTROLLER=DM00,PRIORITY=06,CLASS=F,MPX=XIO,HANDLER=(H.IFXIO,I)
DEVICE=(00,3,2),DISC=MH080,DTC=DM,HANDLER=(H.DCXIO,S),OFF
CONTROLLER=DM08,PRIORITY=07,CLASS=F,MPX=XIO,HANDLER=(H.IFXIO,I)
DEVICE=(00,2,2),DISC=MH080,DTC=DM,HANDLER=(H.DCXIO,S),OFF
CONTROLLER=M910,PRIORITY=08,CLASS=F,MPX=XIO,HANDLER=(H.IFXIO,I)
DEVICE=00,DTC=M9,HANDLER=(H.MTXIO,S),OFF
CONTROLLER=CT7E,PRIORITY=12,CLASS=F,MUX=IOP,SUBCH=F,HANDLER=(H.IFXIO,I)
DEVICE=FC,DTC=CT,HANDLER=H.CTXIO,LINSIZ=80,PAGE=24
CONTROLLER=LF7E,PRIORITY=12,CLASS=F,MUX=IOP,SUBCH=F
DEVICE=F8,DTC=LP,SPOOL=(BL,RL),HANDLER=H.LPXIO
CONTROLLER=DM7E,PRIORITY=12,CLASS=F,MUX=IOP,SUBCH=C
DEVICE=(C0,2,2),DISC=MH080,DTC=DM,HANDLER=H.DCXIO,OFF
```

All possible handlers for a one disc, one tape system are included. Three dummy discs are included so the UDT configurations are similar, and the IOP disc is configured.

The first device initialized is the tape. The IPL device address scratchpad word, saved in Phase I, provides the address of the tape. Using this information and known UDT indices due to the special format SYSGEN directives, SYSINIT configures the UDT, CDT, DCA, and CHT tables, and marks the tape on-line.

The second device initialized is the disc. SYSINIT prompts the operator for the device address and the type of controller (XIO or IOP).

Using answers to the prompts and the known UDT indices, SYSINIT selects the UDT entry with the proper handler for the disc. SYSINIT uses a copy of the SYSGEN module SJ.STBLS, which contains device dependent parameters for the various disc drives, to fill in the following UDT areas: sectors per block, sectors per allocation unit, sectors per track, total sectors, sector size, and number of heads on the unit. The CDT, DCA, and CHT are modified to contain the correct channel and device information. The drive attribute registers are then constructed and the disc is marked on-line.

If the disc specified is a cartridge disc, the even subaddress must be specified when SYSINIT prompts for the device address. SYSINIT then sets up two UDTs and associated tables, one each for the fixed media and the removable media parts of the disc. Either portion may be used in the Volume Formatter step and as a response to SYSINIT's prompt for the system disc address.

After the tape and disc devices are configured, SYSINIT loops through the first seven UDT entries. The device address fields for off-line entries are filled with X'FFFF'. This prevents UDT searches from finding the wrong entry.

After the UDT device address fields are filled, SYSINIT activates the Volume Formatter from the Master SDT.

SYSINIT contains a skip file control word which is used to locate the Volume Formatter. The control word consists of byte fields indicating the relative file positions on the tape. The word defaults to the values required for a user SDT and is modified during Phase I of SYSINIT if a Master SDT is being used. When the tape is positioned, SYSINIT reads the Volume Formatter load module preamble into the system buffer and performs a parameter task/run request activation with the bit indicator 'C.TAPACT' set. The bit variable informs H.REMM, which performs the activation sequence, that the preamble is in the system buffer and activation is from a tape. The bit C.SYSB is also set, which indicates to H.REMM that no swap file should be obtained. During the parameter task part of the activation, SYSINIT passes assignments for the system console and IPL device to the Volume Formatter. During the run request part, SYSINIT waits for completion of the Volume Formatter before continuing the activation sequence.

The activation call causes J.VFMT to be loaded, then placed, into the suspend queue. Control then passes back to SYSINIT. Before SYSINIT resumes J.VFMT, the tape must be positioned at the start of the correct system image. SYSINIT rewinds the tape, advances one record, then uses the skip file control word to advance the required number of files to reach the desired image. For example, the 32/27 would require no skip files where the 32/87 would require two. After the tape is positioned, J.VFMT is resumed. The Volume Formatter copies the tape image of the MPX-32 operating system to disc. The copy becomes the default system image.

When SYSINIT is informed of J.VFMT's completion, the parameter send block associated with the run request is checked for completion errors. If an error is reported, the error is displayed on the system console and SYSINIT aborts.

After running J.VFMT, SYSINIT runs J.MOUNT, J.SWAPR, and the Volume Manager in the same manner. J.MOUNT mounts the system volume, J.SWAPR is the swapper, and the Volume Manager allows the restoration of saved files from the SDT.

7.3.3 Autodisc Subroutine

The autodisc subroutine determines the geometry of any F-class disc except memory disc. This subroutine is used by any program that communicates with an unmounted disc.

Entry Condition

Calling Sequence:

```
EXT     AUTODISK, AUTOFLAG
LW      R7, DISKSPEC
BL      AUTODISK
```

where:

DISKSPEC	contains a word describing the device to be verified. See the MPX-32 Reference Manual Volume I, Chapter 5, Resource Requirement Summary description.
AUTODISK AND AUTOFLAG	are defined labels in the autodisc subroutine.

Exit Conditions

Return Sequence:

Registers:

Normal Return

None

CC1 is zero

Abnormal Return

CC1 is set

R6 error types as follows:

Error Type	Description
1	Error in disc assignment
2	I/O error reading track label zero
3	Media verification pointer not in track label zero
4	I/O error reading media verification sector
5	SYSGEN attributes do not match actual disc parameters and ANY was not specified as the disc type
6	Device not a disc
7	I/O error initializing an IOP disc
8	Device inoperable

R7 error message transfer word address

Autodisc subroutine performs the following:

1. Verifies that the requested device is a nonfloppy disc.

2. Assigns the disc.

3. Reads track label zero.

4. Compares the device dependent parameters computed from the track label with those generated by SYSGEN. If the parameters match, the autodisc subroutine returns to the caller and indicates a successful match. If the parameters do not match, the autodisc subroutine continues processing.

5. Determines if the relevant operating system tables should be modified. There are two ways to determine this:

   - The total sectors field of the UDT is zero. This indicates the disc type was SYSGENed as ANY.

   - Bit two of AUTOFLAG is set.

   If neither of these conditions exists, error type five is generated.

6. Modifies the following table entries:

   UDT.SPT
   UDT.STA2
   UDT.SPAU
   UDT.NHDS
   UDT.SECS
   DCA.SCYL

7. Updates the drive attribute information (DATR). If a disc processor is used, there is no onboard RAM to hold the DATR. A 224-word buffer, containing the DATR information, is allocated following the interrupt fielding module. See H.XIOS in Volume II for details. The autodisc subroutine updates the buffer and places the updated DATR in the MPX-32 drive initialization list.

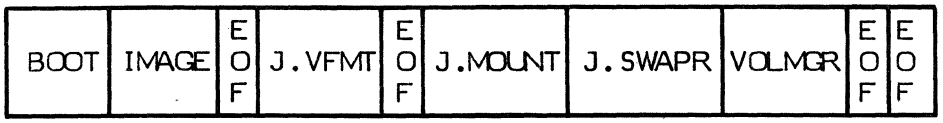
   If an IOP disc is used, there is an onboard RAM which holds the DATR. The IOP allows an initialize controller command to reload the DATRs without a preceding reset channel command. The updated DATR is also placed in the MPX-32 drive initialization list.

   It is necessary to update the MPX-32 drive initialization list as RESTART uses the MPX-32 copy of the DATR to construct the IOCL to load a new MPX-32 image.

8. Deallocates the device and returns to the caller.

7.3.4 Floppy Disc Support

The MPX-32 Master SDT is available on floppy disc. The floppy disc format is as follows:



SYSINIT processes the floppy disc Master SDT in the same manner as the magnetic tape Master SDT, with the following exceptions:

- The floppy disc drive must be device FL7EF0.
- There is only one system image on the floppy disc which corresponds to the machine type.
- The saved files needed for an operational MPX-32 system are contained on separate floppy discs instead of at the end of the system image.

SYSINIT determines a floppy disc Master SDT is being used by the name of the system image being used, for example, FLOP.SYS. Since the floppy disc is addressed by sector number rather than a sequential manner, a subroutine is called. This routine uses information from the preambles of the operating system, J.VFMT and J.MOUNT, to locate the various modules on the disc. The information is placed into the File Assignment Table (FAT) entry for the disc before access is attempted. This logic is used instead of the various rewind/skipfile combinations used by the magnetic tape SDT.

When requesting the Volume Formatter, SYSINIT passes the disc sector address of the system image to the FAT for input to J.VFMT. The image may then be read. H. REMM uses the field RR.PLEN to initialize the starting disc sector address.

When the volume is formatted and mounted, the swapper and Volume Manager are activated. At this point, remove the SDT disc and insert the saved file disc to allow restoration of the needed system files.

7.3.5 Memory Disc

Each memory disc has an associated UDT and SMT. The UDT is constructed by SYSGEN in the same manner as for any other device. The SMT is constructed by SYSGEN in the same manner as for a memory partition, with the exception of SMT.PAGE. For a memory disc, it contains the following values:

value = -1	start of disc not specified at SYSGEN
value = -ve	start of disc specified at SYSGEN, value is the negative of the specified starting map block.

If the DEAL parameter was specified in the SYSGEN DEVICE directive for a memory disc, bit UDT.MDAL in UDT.STA2 is reset. As a result, SYSINIT will not attempt to allocate memory for the memory disc. The memory can be allocated later by the OPCOM ONLINE directive. The value of SMT.PAGE retains the value explained above until the memory disc is marked ONLINE by OPCOM. After memory disc is marked online, the memory disc starting page number is stored by OPCOM into the Shared Memory Table (SMT).

If the DEAL parameter was not specified (UDT.MDAL set), SYSINIT attempts to locate enough contiguous free memory for the memory disc. If the START parameter was specified in the SYSGEN DEVICE directive for the memory disc, bit UDT.MDST in UDT.STA2 is set. As a result, SYSINIT attempts to allocate the memory starting at the specified map block number.

If a starting map block was not specified, UDT.MDST is reset and SYSINIT allocates the memory wherever possible, starting at the high end of the presently configured memory. In either case, SYSINIT does not locate a single-ported memory disc in Multiprocessor Shared Memory System (MSMS) memory, nor does it locate a dual-ported memory disc in non-MSMS memory.

A dual ported memory disc always has its memory allocated by SYSINIT. If SYSINIT fails to allocate memory for a memory disc, the Shared Memory Table (SMT) is cleared for the memory disc that was attempting to allocate memory.

Provided that the memory for the disc was allocated successfully, SMT.PAGE is set to the starting page number of the memory disc. UDT.MDAL is set to indicate that the memory has been allocated.

7.4 On-line RESTART

RESTART is a privileged task which simulates an IPL from the IOP console. RESTART can test a newly SYSGENed version of MPX-32, or replace the current default system image with a new image.

7.4.1 Activating

RESTART runs as a privileged, interactive TSM task. A user activating RESTART must be privileged. See M.KEY, MPX-32 Reference Manual Volume III, Chapter 10.

7.4.2 Required Input

RESTART accepts a pathname as an optional parameter of an activation request. For example:

```
RESTART@VOL1(SYSTEM)TEST.SYS
```

In this case, the image TEST.SYS contained in the system directory of volume VOL1 is booted and VOL1 becomes the system volume. If a volume other than the current system volume is specified in the pathname, the selected volume must be formatted as a system volume. That is, it must contain bootstrap code at sector zero.

If a pathname is not specified at activation, the default image on the current system volume is rebooted.

7.4.3 Processing

The functions performed by RESTART are as follows:

1. The TSM line buffer is checked for a user-supplied pathname. If one is specified, it is moved to a Resource Requirement Summary (RRS) buffer to allow the file to be assigned. If one is not specified, a flag is set to indicate the default image should be used.

2. If a pathname is specified, the file is assigned. The Unit Definition Table (UDT) address of the device it resides on is determined, and the first five sectors of that device, boot code and volume descriptor are read by space definition.

3. The image preamble is used to determine the size of the image and the system name. The name is compared with the user-supplied name. The size is placed in a table to be placed into the boot code, and an IOCD list is constructed that will be used to read the new image.

4. The required drive attribute registers are constructed.

5. At this point, the operator is prompted before proceeding with reboot. If the request was for the default system, a request for reboot is issued. If a pathname was specified, the user is also asked if this image should be made the default system image. If the user replies yes to the default system option, the volume descriptor on the target volume is updated with the new system image definition. This allows IOP console restarts to locate the correct image. In addition, the resource descriptor for the new image is marked as not deletable. This prevents inadvertent relocation of the default system image by SYSGEN, SAVE, RESTORE, COPY, etc. which causes an IOP console IPL to fail. The previous default image, if not the SDT image, has this flag reset so it may be deleted.

6. The CPU scratchpad IPL device address location is updated to reflect the new IPL device. This location is used by the bootstrap code as it sets up scratchpad to perform all of its I/O from logical channel 08.

7. A simulated system reset is performed. All interrupts are disabled and channels are reset.

8. The bootstrap code read from the target disc is updated by RESTART and moved into low memory.

9. Control is passed to the bootstrap code, which reads the image and transfers control to the system initializer program, SYSINIT.

CHAPTER 8

INTERNAL PROCESSING UNIT (IPU)

8.1 Overview

The IPU is a parallel processor connected directly to the SelBUS. Synchronization between the CPU and the IPU is maintained by one CPU trap and sixteen IPU traps. The traps and default trap vector locations are shown in Table 8-1.

Task execution in the IPU is transparent to the user. Scheduling for the IPU is accomplished by the MPX-32 Executive with no user intervention. However, IPU biasing or inhibiting may be used to maximize the performance of processor or I/O bound tasks.

When an IPU is configured in a system, two modules must be included in the resident operating system to perform IPU biased scheduling and to provide trap handlers for IPU related traps. These modules are H.CPU and H.IPU.

IPU accounting can be performed by a second interval timer (RTOM). IPU execution time and idle time are tabulated by the resident handler, H.IPUIT.

When both a CPU and an IPU are configured, memory "Read'N Lock" is automatically enabled. When one processor is accessing a memory location, the other processor is prohibited from accessing that location. Memory locations are unlocked when they are not being accessed.

8.1.1 IPU - Memory Interface

The IPU can address all locations of physical memory. Task loading and initialization are performed by the CPU before the task is queued for IPU execution. The Task Service Area (TSA) includes pointers to all physical map blocks used by the task, allowing the IPU to remap to the task's address space. This mechanism allows the CPU and IPU to be coordinated in use of memory.

8.1.2 IPU - CPU Interface

Table 8-1 shows IPU related traps and default trap vector locations. The trap vectors may be stored at alternative locations if the IPU scratchpad is set up appropriately.

Table 8-1
IPU Trap Structure

IPU Trap Vector Location	CPU Trap Vector Location	Trap Condition
20	80	Power fail
24	84	Power on/autostart
28	88	Memory parity
2C	8C	Nonpresent memory
30	90	Undefined instruction
34	94	Privilege violation
38	98	Supervisor call
3C	9C	Machine check
40	A0	System check
44	A4	Map fault
48		Undefined IPU instruction
4C		CPU issued SIPU instruction
	AC	IPU issued SIPU instruction
50	B0	Address specification error
54	B4	Console attention
58	B8	Privilege mode halt
5C	BC	Arithmetic exception
60	C0	Cache fault

8.2 Task Scheduling and Execution

8.2.1 Task Biasing

There are three scheduling options related to IPU task execution: IPU biased, CPU only, and unbiased. IPU biased tasks are scheduled for IPU execution whenever possible. Some SVCs are executed directly by the IPU, but most SVCs and all privileged instructions cause a task to be returned to the CPU for execution. An IPU biased task is rescheduled for IPU execution at the instruction following the instruction which caused the trap.

A CPU only task is never scheduled for IPU execution.

An unbiased task can be scheduled in the CPU or the IPU depending on available resources. The CPU scheduler, S.EXEC20, is responsible for selecting unbiased tasks for IPU or CPU execution.

8.2.2 Standard CPU/IPU Scheduling

There are two head cell addresses used by the operating system to control IPU task execution: C.CIPU and C.RIPU. The currently executing IPU task is linked to the head cell, C.CIPU. C.RIPU is a standard linked list head cell containing the dispatch queue (DQE) addresses of all IPU biased tasks awaiting IPU execution.

Tasks are linked to the above state queues by the CPU, and control is passed to the IPU after the task has been linked to the C.CIPU head cell.

8.2.3 Optional CPU/IPU Scheduling

IPU-biased tasks are not automatically enqueued on the RIPU list as they are in the standard CPU/IPU scheduler. The RIPU list is used only when the task in the CPU is replacing the IPU task.

8.2.4 Standard Scheduling of IPU-biased Tasks

Tasks are biased to the IPU by specifying OPTION IPUB at either catalog or run time, or by calling the dynamic IPU bias service, M.IPUBS. Biased tasks are queued on the IPU ready-to-run queue by priority with the highest priority task at the head. Tasks linked to C.RIPU are executed by the IPU ahead of higher priority unbiased tasks. Task replacement of IPU biased tasks occurs when a higher priority task is linked to C.RIPU or the currently executing IPU task executes an instruction causing control to be returned to the CPU. If a biased task is linked to C.RIPU while a higher priority unbiased task is currently executing in the IPU, the higher priority task continues to run in the IPU.

8.2.5 Optional Scheduling of IPU-biased Tasks

The optional CPU/IPU scheduler is enabled by the SYSGEN DELTA directive. This directive replaces system modules H.EXEC and H.CPU with H.EXEC2 and H.CPU2. The optional scheduling approach does not enqueue IPU-biased tasks on the RIPU list as the standard scheduler does.

Instead, all tasks are linked to their appropriate ready-to-run state chains. When the delta value is zero, scheduling is performed on both processors according to the tasks' original priorities. When the delta value is greater than zero and less than 55, the value is subtracted from the original priority of the IPU-biased task to create a new priority. The new priority is used during IPU scheduling and when the IPU-biased task needs the CPU for system service execution. When IPU-biased tasks run on the CPU, the new priority does not apply.

8.2.6 Scheduling Unbiased Tasks

When an IPU biased task is not linked to C.CIPU or C.RIPU, IPU task selection proceeds with unbiased tasks. The ready state queues are searched for the first eligible task, starting with the highest priority real-time task. The conditions for IPU eligibility are:

- Task is not CPU only
- Task is not inhibited for IPU execution because it has executed an instruction not available to the IPU
- There are no run requests or messages outstanding against the task
- There are no system action requests outstanding against the task

If all the above conditions are met, the task is linked to C.CIPU and control is passed to the IPU. If there are no ready-to-run tasks meeting all of the above conditions, the IPU remains idle.

8.2.7 Scheduling CPU Only Tasks

Tasks with OPTION CPUO specified are never scheduled for IPU execution. Typically, these tasks are I/O bound.

8.2.8 IPU Task Execution

A task ceases execution in the IPU when one of the following events occur:

- The IPU encounters a system service request (SVC or CALM). Some SVCs are executed by the IPU and control is not returned to the CPU.
- The IPU encounters an exceptional or error condition; for example, privilege violation, undefined instruction.
- The CPU executes an SIPU instruction.

Tasks running with batch priorities (55 through 64) are not subject to time distribution while being executed in the IPU.

8.3 IPU Executive Module Description

The resident module H.IPU is responsible for initializing the IPU, dispatching tasks linked to the IPU current state queue, C.CIPU, handling all IPU traps, and returning control to the CPU on exceptional or error conditions.

8.3.1 Entry Point 1 - IPU Executive

This entry point is used when the CPU issues an SIPU instruction. The first time the trap occurs, a branch is made to the initialization subroutine, S.IPU2. The on-line restart in progress flag is checked and, if set, a simulated IPU reset is performed. If the IPU is currently executing a task, the CPU is requesting a context switch in the IPU. The context of the task is preserved and the IPU issues an SIPU instruction to allow the CPU to link the new task to C.CIPU.

If there is not a current task, the IPU dispatches control to the task linked to C.CIPU.

If there is a current task but the IPU is executing within H.IPU, the IPU is in the process of handling a trap and is allowed to continue.

8.3.2 Entry Point 2 - Undefined IPU Instruction

This entry point sets the IPU inhibit bit in the current task's DQE, which corrects the PSD to point to the last instruction executed by the task, and returns control to the CPU to re-execute the instruction.

8.3.3 Entry Point 3 - Memory Parity Error

This entry point sets the IPU inhibit bit in the current task's DQE, which corrects the PSD to point to the last instruction executed by the task, and returns control to the CPU to re-execute the instruction.

8.3.4 Entry Point 4 - Nonpresent Memory

This entry point causes the current task to be aborted. The error code is stored in the DQE and the abort request bit is set before returning control of the task to the CPU.

8.3.5 Entry Point 5 - Undefined Instruction

This entry point causes the current task to be aborted. The error code is stored in the DQE and the abort request bit is set before returning control of the task to the CPU.

8.3.6 Entry Point 6 - Privilege Violation

This entry point causes the current task to be aborted. The error code is stored in the DQE and the abort request bit is set before returning control of the task to the CPU.

8.3.7 Entry Point 7 - Map Fault

This entry point causes the current task to be aborted. The error code is stored in the DQE and the abort request bit is set before returning control of the task to the CPU.

8.3.8 Entry Point 8 - SVC Trap Handler

This entry point contains a secondary vector table for the IPU SVCs. SVC types zero, three, and five through fifteen are returned to the CPU for processing. The PSD is corrected to point to the SVC instruction, the current context is saved, and control is returned to the CPU.

Some SVC types one and two are executable directly by the IPU. These SVCs have bit one in the SVC table set. The SVC number is retrieved from the trap status word and the SVC table entry is checked. If the SVC is not executable by the IPU, control is returned to the CPU as for an SVC type zero. If the service is executable by the IPU, a PSD is constructed and control is transferred to the correct SVC processor.

An SVC type four is the mechanism for returning from an SVC. This entry point is reached by the macro M.IPURTN, which determines if any registers are to be returned and issues an SVC type four. The registers and PSD are popped from the stack and control is returned to the task.

8.3.9 Entry Point 9 - Arithmetic Exception Trap Handler

When an arithmetic exception occurs, the arithmetic exception bit in the current task's TSA is set and may be tested by the M.TSTE service. The return registers are set to the following values, depending on the cause of the exception:

Underflow	Zero
Positive overflow	Maximum positive value
Negative overflow	Maximum negative value

8.3.10 Entry Point 10 - Privilege Mode Halt

The function this entry point performs depends on where an error occurs. If an error occurs in a task, an abort request is set up for the task and control is returned to the CPU. If an error occurs in the operating system, the registers are set up as for the CPU M.KILL macro and stored in the buffer HLT.REG within H.EXEC. The register contents are as follows:

Register	Meaning
0	Old PSD word one
1	Old PSD word two
2	Address of the instruction causing the error
3	Instruction causing the error
4	Trap status word
5	ASCII code of reason for trap, for example, HT02
6	Address of register save block
7	ASCII 'TRAP'

A flag is set requesting the IPU to be marked off-line so no further tasks are scheduled for the IPU.

8.3.11 Entry Point 11 - Address Specification

The function this entry point performs depends on where an error occurs. If an error occurs in a task, an abort request is set up for the task and control is returned to the CPU. If an error occurs in the operating system, the registers are set up as for the CPU M.KILL macro and stored in the buffer HLT.REG within H.EXEC. The register contents are as follows:

Register	Meaning
0	Old PSD word one
1	Old PSD word two
2	Address of the instruction causing the error
3	Instruction causing the error
4	Trap status word
5	ASCII code of reason for trap, for example, AD01
6	Address of register save block
7	ASCII 'TRAP'

A flag is set to request the IPU be marked off-line so no further tasks are scheduled for the IPU.

8.3.12 Entry Point 12 - Cache Fault

The function this entry point performs depends on where an error occurs. If an error occurs in a task, an abort request is set up for the task and control is returned to the CPU. If an error occurs in the operating system, the registers are set up as for the CPU M.KILL macro and stored in the buffer HLT.REG within H.EXEC. The register contents are as follows:

Register	Meaning
0	Old PSD word one
1	Old PSD word two
2	Address of the instruction causing the error
3	Instruction causing the error
4	Trap status word
5	ASCII code of reason for trap (for example, CP01)
6	Address of register save block
7	ASCII 'TRAP'

A flag is set to request the IPU be marked off-line so no further tasks are scheduled for the IPU.

8.3.13 Entry Point 13 - Machine Check

The function this entry point performs depends on where an error occurs. If an error occurs in a task, an abort request is set up for the task and control is returned to the CPU. If an error occurs in the operating system, the registers are set up as for the CPU M.KILL macro and stored in the buffer HLT.REG within H.EXEC. The register contents are as follows:

Register	Meaning
0	Old PSD word one
1	Old PSD word two
2	Address of the instruction causing the error
3	Instruction causing the error
4	Trap status word
5	ASCII code of reason for trap, for example, MC01
6	Address of register save block
7	ASCII 'TRAP'

A flag is set to request the IPU be marked off-line so no further tasks are scheduled for the IPU.

A second trap within a task causes the IPU to halt as above since this is assumed to be a hardware failure.

8.3.14 Entry Point 14 - System Check

The function this entry point performs depends on where an error occurs. If an error occurs in a task, an abort request is set up for the task and control is returned to the CPU. If an error occurs in the operating system, the registers are set up as for the CPU M.KILL macro and stored in the buffer HLT.REG within H.EXEC. The register contents are as follows:

Register	Meaning
0	Old PSD word one
1	Old PSD word two
2	Address of the instruction causing the error
3	Instruction causing the error
4	Trap status word
5	ASCII code of reason for trap, for example, SC01
6	Address of register save block
7	ASCII 'TRAP'

A flag is set to request the IPU be marked off-line so no further tasks are scheduled for the IPU.

A second trap within a task causes the IPU to halt as above since this is assumed to be a hardware failure.

8.3.15 Entry Point 15 - Power Fail Trap

This trap saves the general purpose and base registers, and saves the IPU scratchpad key in memory location X'6D4'. This provides the required parameters for a power-up auto-restart when power is restored to the system. The privileged mode halt trap is disabled and the IPU halts.

8.3.16 Subroutine S.IPU1 - Perform Stack Push

This subroutine pushes the registers and Program Status Doubleword (PSD) of the current task into the next stack frame as defined by T.REGP. This is a clean-up activity in preparation for the return of task control to the CPU.

Calling Sequence

```
LA      R2,addr
LD      R6,psd
BL      S.IPU1
```

addr is the address of the register save block

psd is the current task PSD

Exit Sequence

```
TRSW  R0
```

8.3.17 Subroutine S.IPU2 - IPU Initialization

This subroutine initializes the IPU by storing the Master Process List (MPL) address in the IPU scratchpad, loading the shared map registers for the operating system, setting up the address of the IPU history buffer, and enabling the privilege mode halt trap.

Calling Sequence

```
BL      S.IPU2
```

Exit Sequence

```
TRSW  R0
```

8.3.18 Subroutine S.IPU3 - Terminate IPU Execution

This subroutine generates an SIPU instruction which causes a trap in the CPU to indicate the IPU has finished processing.

Calling Sequence

```
BL      S.IPU3
```

Exit Sequence

None

8.3.19 Subroutine S.IPU4 - Generate IPU History Buffer

This subroutine maintains a circular buffer containing information on the last twenty traps in the IPU. The buffer contains the task name, PSD at the time of the trap, and a code of the reason for the trap. Each entry is four words long in the following format:

Word	Meaning
0-1	Task name
2	PSD word one
3	PSD word two with bits 10 - 14 containing function code as follows:

Code	Meaning
2	Nonpresent memory
3	Undefined instruction
4	Privilege violation
5	SVC type zero or three
6	SVC type one
7	SVC type two
8	Machine check
9	System check
10	Map fault
11	Unidentified IPU instruction or memory parity error
12	Start IPU
13	Address specification
14	Privilege mode halt
15	Arithmetic exception
16	Cache fault
17	SVC type four

8.4 IPU Auto Start Trap Processor - H.IPUAS

The interrupt or trap signal occurs at priority level X'01'. This trap occurs during the power up sequence, provided the following operating conditions are met:

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

If any of the conditions are not met, an automatic trap halt is exectued.

If the conditions are met, H.IPUAS disables the privileged halt trap and halts the IPU.

The H.IPUAS trap handler can be replaced with a user-supplied routine as follows:

1. Specify the new trap in the SYSGEN SYSTRAP directive.
2. At SYSGEN, the address of the user-supplied routine's trap context block must be saved in memory location X'24'.
3. The five operating conditions must be met.

8.5 IPU Task Scheduler - H.CPU/H.CPU2

Two IPU task schedulers are provided. H.EXEC and H.CPU are the standard IPU scheduling modules; H.EXEC2 and H.CPU2 are the optional IPU scheduling modules. The differences that apply to H.CPU2 are noted in the following sections.

8.5.1 Entry Point 1 - Field IPU Halt

This entry point fields the IPU 'HALT' trap. It schedules IPU tasks based on the following criteria:

1. If the head cell count of the IPU current state is zero, the IPU is idle. Entry point two schedules the current task.
2. If the head cell count is greater than zero and the Inhibit IPU flag is set, entry point two unlinks the current task, relinks it at its base priority state, then schedules the new IPU task.
3. If the head cell count is greater than zero and the Inhibit IPU flag is reset, entry point two unlinks the task from the current state, relinks it to the IPU request queue, then schedules the new IPU task.

Note: For the optional scheduler, H.CPU2 never links the task to the IPU request queue. The task is always linked to the ready to run queue. If it is a real-time task, it is linked at its base priority minus its delta value. All other tasks are linked at their base priority.

8.5.2 Entry Point 2 - Schedule IPU Biased Tasks

If tasks are queued to the IPU request queue, this entry point unlinks the highest priority task, relinks it to the IPU current state, and calls the IPU start subroutine. If there are no tasks on the IPU request queue, this entry point goes to entry point three for unbiased task selection.

8.5.3 Entry Point 3 - Schedule Unbiased Tasks

This entry point begins at the real-time state queue to select an IPU candidate. It tests each encountered task for IPU eligibility as follows:

1. IPU inhibit flag = reset
2. CPU only flag = reset
3. No system actions (DQE.SAIR = 0)
4. No run requests (DQE.RTI = 0)
5. Execution address is not in the operating system

This entry point continues testing each lower priority task until an eligible candidate is found. It unlinks that task from its ready state and relinks it to the IPU current state. This entry point then calls the start IPU subroutine. If no eligible task is found, the IPU remains idle.

8.5.4 Subroutine S.CPU1 - Link Task to IPU Request State

This subroutine links a task to the IPU request queue. If the task is higher priority than the current IPU task, IPU task replacement takes place.

8.5.5 Subroutine S.CPU2 - IPU Eligibility Test

This subroutine contains the tests for task eligibility to run in the IPU. The items checked are:

- Is the task executing in the monitor (DQE.OSD)
- Is the task CPU only (DQE.IPUR)
- Is the task IPU inhibited (DQE.IPUH)
- Is a system action request pending against the task (DQE.SAIR)

8.6 IPU Accounting Module Descriptions

8.6.1 Entry Point 1 - Field Interval Timer Interrupt

This entry point fields the IPU accounting interval timer interrupt. If there is a current IPU task, it updates a local IPU execution time accumulator. If the IPU is idle, the IPU idle time accumulator (C.IDLA1) is updated. The timer is then reset for one second and the handler is exited.

8.6.2 Subroutine S.IPUIT1 - Perform Accounting After IPU Trap

This subroutine is called by H.CPU after an IPU HALT trap is fielded. It updates the TSA of the current IPU task with the accumulated IPU execution time and resets the interval timer to accumulate idle time.

8.6.3 Subroutine S.IPUIT2 - Perform Accounting Prior to Starting the IPU

This subroutine is called by H.CPU just prior to calling S.EXEC80 to start the IPU. It updates the IPU idle time accumulator and resets the timer to accumulate execution time.

8.7 IPU SYSGEN Directives

To generate a system with an IPU configured, the following SYSGEN directives must be used:

```
//HARDWARE
/PARAMETERS
MACHINE = machine type
IPU
 .
 .
 .
/TRAPS
```

If an IPU accounting interval timer is present, the following SYSGEN directives should be used:

```
/INTERRUPTS
PRIORITY = xx,RTOM = (channel,subaddress),PROGRAM = H.IPUIT,INTV
```

Note: It is recommended that xx be 5E. However, if a scientific accelerator is also configured, priority 3F should be used for the IPU accounting interval timer.

8.8 SVCs Executable by an IPU

Certain SVCs are executable directly by the IPU. When modifying these SVCs or when writing new services which are executed by the IPU, the following guidelines should be followed:

- The service must not be called by SYSGEN or J.SWAPR.
- C.TSAD must be used instead of C.REGS.
- T.PRNO must be used instead of C.CURR.
- M.IPURTN must be used instead of M.RTRN.
- The macro M.SVCP must be used in the SYSGEN initialization entry point to set bit one in the SVC table.

APPENDIX A

SYSTEM TABLES AND VARIABLES CROSS-REFERENCE

TABLE NAME	DESCRIPTION	SECTION
ART	Allocated Resource Table	2.3
CDT	Controller Definition Table	2.7
CHT	Channel Definition Table	2.6
CNP	Caller Notification Packet	2.5
DAT	Dispatch Queue Address Table	2.13
DCA	Device Context Area	2.8
DQE	Dispatch Queue Entry	2.12
DTT	Device Type Table	2.9
FAT	File Assignment Table	2.14
FCB	File Control Block	2.15
FPT	File Pointer Table	2.16
IOQ	I/O Queue (IOQ) Entry	2.17
J.SOEX	Run Request	2.33.3
J.SOUT	Run Request	2.33.4
J.SSIN	Run Request	2.33.1
J.TSM	Run Request	2.33.2
M.BB.DEQ	Bad Block Descriptor	2.41.3
M.BD.DEQ	Descriptor Map (DMAP) Deallocation File Descriptor	2.41.6
M.BS.DEQ	Space Map (SMAP) Deallocation File Descriptor	2.41.11
M.DD.DEQ	Descriptors Descriptor	2.41.5
M.DI.DEQ	Directory Descriptor	2.41.7

TABLE NAME	DESCRIPTION	SECTION
M.DM.DEQ	Descriptor Allocation Map Descriptor	2.41.4
M.DN.TEQ	Directory Entry Table	2.10
M.FI.DEQ	File Descriptor	2.41.8
M.ME.DEQ	Memory Parition Descriptor	2.41.9
M.SM.DEQ	Space Allocation Map Descriptor	2.41.10
M.VO.DEQ	Volume Descriptor	2.41.12
M.KEY	M.KEY Entry Format	2.19
M.PRJCT	M.PRJCT Format	2.20
M.RDACC	Resource Descriptor Access Parameters	2.41.1
M.RDACT	Resource Descriptor Accounting Parameters	2.41.1
M.RDCOM	Resource Descriptor	2.41.1
M.RDID	Resource ID	2.41.1
M.RDSPD	Resource Descriptor Space Definition	2.41.2
M.RIQ	Resource Inquiry Table	2.29
MATA	Memory Allocation Table	2.21
MDT	Memory Resident Descriptor Table	2.24
MEML	Memory Attribute List	2.22
MIDL	Map Image Descriptor List	2.20
MRRQ	Message or Run Request Queue	2.25
MVT	Mounted Volume Table	2.27
RD.SEGDF	Segment Definitions	2.41.13
RD.USER	User Area	2.41.14
RCB	Resource Create Block	2.28
RLB	Resource Logging Block	2.30
RRS	Resource Requirement Summary Entries	2.31
SMD	System Master Directory	2.34
SMT	Shared Memory Table	2.32

TABLE NAME	DESCRIPTION	SECTION
T.MEMLA	Memory Attribute List	2.22
T.MIDLA	Map Image Descriptor List	2.20
TCPB	Type Control Parameter Block	2.38
TSA	Task Service Area	2.35
UDT	Unit Definition Table	2.39
VAT	Volume Assignment Table	2.40
N/A	Blocking Buffer Control Cells	2.4
N/A	Communications Region	2.2
N/A	Dispatch Queue Area	2.11
N/A	Disc Resident Resource Descriptors	2.41
N/A	Handler Tables and Corresponding Hardware	2.17
N/A	I/O Table Linkages	2.17
N/A	Load Module Preamble	2.42.3
N/A	Load Module Structure	2.42.2
N/A	Memory Layout	2.1
N/A	Module Address Table	2.26
N/A	Memory Pool Management	2.23
N/A	Executable Image Preamble	2.42.5
N/A	Executable Image Structure	2.42.4
N/A	Shared Executable Image Preamble	2.42.7
N/A	Shared Executable Image Structure	2.42.6
N/A	Spooled File Data Structures	2.33
N/A	Terminal Line Buffer	2.36
N/A	Timer Table	2.37
N/A	Volume Format	2.42.1

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

GOULD
Electronics

Users Group Membership Application

USER ORGANIZATION: _____

REPRESENTATIVE(S): _____

_____

_____

ADDRESS: _____

_____

TELEX NUMBER: _____ PHONE NUMBER: _____

NUMBER AND TYPE OF GOULD CSD COMPUTERS: _____

_____

OPERATING SYSTEM AND REV. LEVEL: _____

_____

APPLICATIONS (Please Indicate)

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

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

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

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

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

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

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

8. Other

Please return to:

Users Group Representative

Date: _____

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

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

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

Fold and Staple for Mailing

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE

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

Fold and Staple for Mailing

(Detach Here)

GOULD
Electronics