Burroughs

B 800 Systems
MCP
Memory Dump Analysis

USER'S GUIDE

PRICED ITEM

Printed in U.S.A.

February 1981

1118452

"The names used in this publication are not of individuals living or otherwise. Any similarity or likeness of the names used in this publication with the names of any individuals, living or otherwise, is purely coincidental and not intentional."

Any comments or suggestions regarding this publication should be forwarded to Systems Documentation, Technical Information Organization, TIO-East, Burroughs Corporation, P.O. Box CB7, Malvern, Pennsylvania 19355.

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

The intent of this document is to provide additional information to system software support personnel on how to analize B 800 system dumps. It is assumed that the reader is already familiar with the B 800/CMS virtual machine structure. This manual should not be expected to provide all the information required for a complete analysis of a memory dump. However, the information should enable the support person to:

1. Investigate hardware/software status from clear/starts caused by hardware malfunctions, user program logic errors, or MCP failures.

2. Gain additional insight into the status of the system at the time of the failure.

3. Suggest methods for correcting or detouring the problem.

4. Better document the field trouble reports (FTR) when reporting clear/starts or other system faults.

In some cases, the information obtained from the dump analysis may not provide enough data to exactly define the failure or the cause. However, in reviewing documentation from several failures, similarities may appear which may lead to the cause of the failure.

A list of some of the terms and acronyms used in this document is contained in Appendix A. The information is based on the 3.01.50 B 800 CMS MCP release.

SECTION 1
MCP CLEAR/START TROUBLESHOOTING

The objective of this trouble shooting is to analyze certain areas in a system dump, enabling the documenting of an FTR, and to attempt to gather as much knowledge as possible about the status of the system at the time of the MCP failure.

The following is a guide to information that should be documented on an FTR. Analyses of the following areas should be kept on each reported failure and reviewed with other reports in a search for patterns and similarities. The steps are:

1. Obtain a copy of the SPO log from the last warmstart banner. Review the log for possible error messages from the MCP. Make note of the patch level from the warmstart banner.

2. Verify from the MAT that the task entries do exist. The MCP may just be starting or terminating a task. (Refer to Mix Table Worksheet.)

3. Identify the contents of BMAR, MPCR, and ERROR. These values can be obtained from OS register locations @0083@, @0084@, and @0085@ respectively.

4. Identify the current slice (OS register @000C@).

5. Determine whether the memory links are correct and identify the micro slices in the micro extent area. (Refer to Slice Memory Mapping section.)

6. Identify the current task TID (OS register @000A@) and its status.

7. If the current task is a user program, provide a brief functional description of the program. Analyze the current task for the last MCP communicate. (Refer to Current Task section.)

8. Identify the hardware configuration.

9. Identify any devices with queued I/Os and their status. (Refer to Peripheral I/O Descriptor section.)

10. Examine the MCP trail for the history of MAD and interrupts. This trail helps identify the last path taken by the MCP up to the failure. (Refer to MCP trail section.)

In some cases, the preceding information, by itself, does not provide enough data to exactly define the cause of the problem. Only through the review of

documentation from several failures do similarities start to develop. Investigation into these similarities produces new ideas and theories which, when investigated, may result in clearly defining a set of operations reproducing the failure or finding of a possible detour around the failure.

SECTION 2
MIX TABLE ANALYSIS

The objective is to extract the mix table information from a Hex dump.

Every attempt should be made to accompany each system dump with the SPO listing from the last warmstart banner to the system failure. In cases where the SPO listing should be verified against the system dump, or if the SPO listing is not available, the mix entries can be extracted from the Hex system dump by using the mix attribute table (MAT) and the task detail table (TDT).

The MAT is a task management table located in the operating system table region. It is the mechanism by which the operating system controls task switching and priorities of tasks. Each entry is one D-word long and a maximum of 16 entries are divided into priority classes A, B, and C. In between each class is a wall, @FFFF@.

The task detail table is the logical extension of the mix and, as such, contains a great amount of task-related data. The TDT is an overlayable data segment owned by the SCL task. It is comprised of one entry per task and is indexed by the task number.

Summarize the mix contents as follows by locating the information in the MAT and TDT as subsequently described.

Independent Runners:

Task No.	Priority	Task ID	Status
F -	C	- Data comm	- _____
E -	C	- Working set bailiff	- _____
D -	C	- Help/CTM	- _____
C -	C	- SCL Loader	- _____
B -	C	- Data comm space handler	- _____
A -	C	- Log	- _____
9 -	B	- SYS-SUPERUTL	- _____

Use the following table to identify the status for each of the independent runners previously listed.

User programs:

Task No. Priority Task ID Status

8 - _____ - _____ - _____

7 - _____ - _____ - _____

6 - _____ - _____ - _____

5 - _____ - _____ - _____

4 - _____ - _____ - _____

3 - _____ - _____ - _____

2 - _____ - _____ - _____

1 - _____ - _____ - _____

0 - _____ - _____ - _____

Identify the priority, task name, and status for each task number.

MAT - MIX ATTRIBUTE TABLE ANALYSIS

Complete the following:

The operating system register at location @000E@ = _____.

The contents of this register identify the first D-word in the mix attribute table (MAT) and should be equal to @FFFF@. The format of the MAT is: wall (FFFF), C tasks, wall (FFFF), B tasks, wall (FFFF), A tasks, wall (FFFF). If no B or A tasks are in the mix, the walls are next to one another. Each entry is one D-word long. The objective is to extract the priority and wait key status for each task.

C FFFF _____ _____ _____ _____ _____ _____

_____ _____ _____ _____ _____ _____ _____ _____ _____

B FFFF _____ _____ _____ _____ _____ _____ _____ _____ _____

A FFFF _____ _____ _____ _____ _____ _____ _____ _____ _____ FFFF

Each entry in the MAT can be defined as abcd where:

a = task state
  = 0 - runnable
  = 4 - short wait

cd = wait key status
  = 00 - running
  = 03 - virtual memory lock

a = 8 - suspended swappable
  = C - suspended swapped

b = task ID = mix number- 1
  = 9 - super utility
  = A - log
  = B - DC space
  = C - SCL loader
  = D - help task
  = E - W.S. bailiff
  = F - DC Q handler

cd = 04 - SCL loader lock
   = 05 - open/close lock
   = 06 - M.H. accept,display
   = 07 - EOJ lock
   = 09 - I/O wait
   = 0A - logging initiated by WMST
   = 0B - logging initiated recall
   = 10 - operator stopped
   = 11 - suspended by pause
   = 12 - no user disk
   = 13 - no file
   = 14 - duplicate file
   = 21 - task waiting restore
   = 22 - rstinp
   = 23 - swpint
   = 40 - help task not in use
   = 41 - help task waiting CTM
   = 42 - task being loaded
   = 43 - DC result Q suspended
   = 44 - DC space handler is idle
   = 45 - task wait on accept
   = 46 - task wait zip with pause
   = 47 - task wait event timer
   = 48 - access mode restriction
   = 49 - waiting device
   = 4A - directory full
   = 4B - task wait for ds/dp
   = 4C - waiting SYSMEM file
   = 4D - task wait for device assignment
   = 4E - DC wait key
   = 4F - wait MCS allow/disallow
   = 60 - wait on DC buffer space
   = 61 - wait on subnet Q limit
   = 62 - wait station Q limit
   = 63 - wait MCS EOJ
   = 64 - wait MCS Q limit
   = 65 - wait DC reload
   = 6D - wait on sysconfig access
   = 6E - wait trace buffer
   = 6F - VM wait on I/O

TDT - TASK DETAIL TABLE ANALYSIS

The task detail table is found by locating data segment 1 of the SCL-loader task. The objective is to extract the task ID for each task.

1. The base memory address of the slice descriptor table (SDT) is the value contained in OS register location @0017@ = _____.

2. The preceding memory location identifies the base of the SDT. The SCL-loader is the thirteenth entry in this table. To locate the SCL loader slice descriptor add @3C@ to the base address of the SDT, giving _____.

3. Each SDT entry is five D-words long. Using the address just evaluated, extract the SCL loader slice descriptor.

_____ _____ _____ _____ _____ (SCL slice descriptor)
1 2 3 4 5

4. D-word 4 identifies the memory location of the SCL loader task control block. Proceed to that memory location and extract two D-words.

_____ _____ (first two words of SCL loader TCB)
1 2

5. D-word 2 of the SCL TCB identifies the memory location of the data segment table base for the SCL loader. Since data segment numbers are zero-relative and the data segment desired is number 1, add @04@ to this memory address.

_____ (SCL data segment 1)

6. This memory location finds the SCL loader segment descriptor (four D-words long). Extract four D-words.

_____ _____ _____ _____ (TDT segment descriptor)
1 2 3 4

7. D-word 4 for the TDT segment descriptor identifies the memory address where the task detail table information can be found. The TDT can now be mapped. Each entry in this table is 23 D-words long and is indexed by the task number.

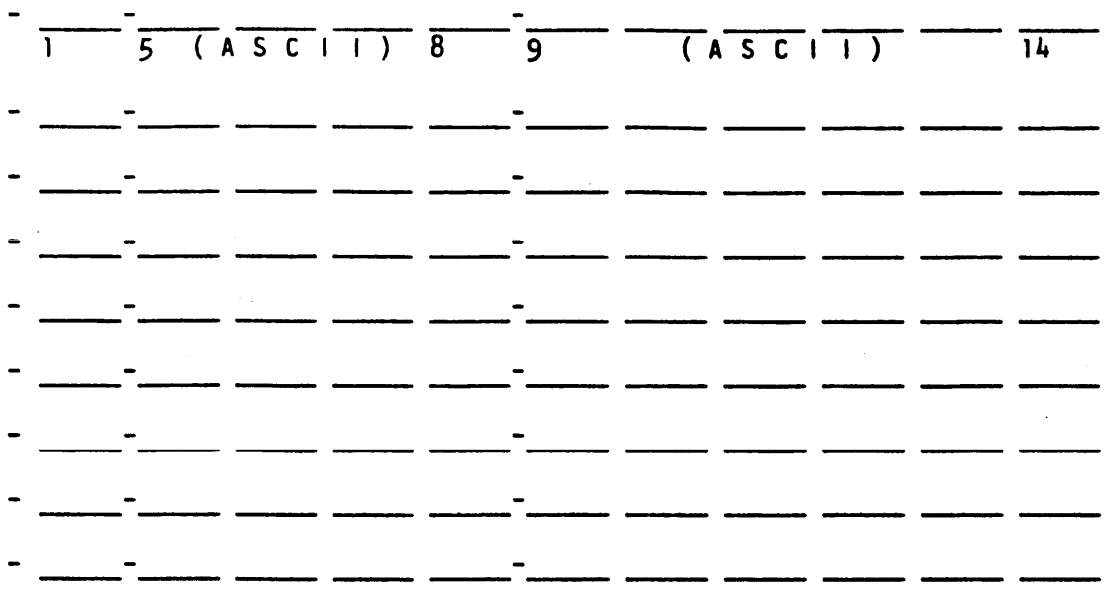
D-word 1 = task history and is defined as:

abcd where a = 0 - SPO initiated
= 8 - regular zip
= 2 - zip with DS
= C - zip with pause
c = son's task ID
d = father's task ID

D-words 5 - 8 contains the pack ID in ASCII.

D-words 9 - 14 contains the task ID in ASCII.

Mark off sets of 23 D-words on the Hex dump and extract the task history, pack-ID, and task-ID. Only if a user task is present in the mix is there a TDT entry. The number of user tasks should have been determined by the MAT analysis.


1
5 (ASCII)
8
9
(ASCII)
14

SECTION 3
CURRENT TASK ANALYSIS

This section utilizes the current task worksheet located at the end of this manual.

Part 1 concerns current task information in low memory. (What program was running? What was it doing?)

This part can be completed with a minimum of effort. The resulting information may be adequate in many cases as a preliminary analysis. It may, for example, establish a pattern of repeated failures while executing a certain program or at a certain point in a program. This could be helpful for reporting, reproducing, or detouring a system failure.

Current task-ID _____ (word @000A@)

Mix no. _____ (task-ID + 1)

Program name

_____

Write down the contents of memory location @000A@ from a Hex dump. Task-ID values of 0-8 correspond to mix numbers of 1-9. Task-IDs greater than 8 are not assigned to a user task. Thus, if current task-ID is greater than 8, the remainder of this worksheet does not apply.

The SPO log listing accompanying the dump gives the disk codefile name for this task. If the SPO log is not available, the codefile name can be found in the task detail table in the dump (see Mix Table Worksheet).

Current task TCB _____ (word @000B@) also referred to as TBC base

Write down the contents of memory location @000B@. This gives the starting address in memory of the task control block (TCB) for the currently executing program.

During a particular execution of a program, the MCP and interpreters build a TCB and use it for the storage and exchange of information about the task's progress. The TCB is examined in part 2 of this worksheet.

Several important virtual registers used by the interpreter during program execution are located in low memory for convenient access by the interpreter. There are several S-registers used by the interpreters, but we are particularly interested in these:

Code segment number (PSN) (word @0094@) _____

Code displacement in bytes (PCA) (word @0093@) _____

Words @94@ and @93@ (PSN and PCA) contain program segment number and program current address, which together constitute the next instruction pointer.

Code segment address (word @009A@) _____

Code segment size (bytes) (word @009B@) _____

Words @9A@ and @9B@ contain the absolute memory address and size in bytes of the current program code segment.

Current S-op (word @00AD@) _____

Word @AD@ contains the actual S-opcode being executed by the S-interpreter.

Line-count or BIL-flags (word @00B6@) _____

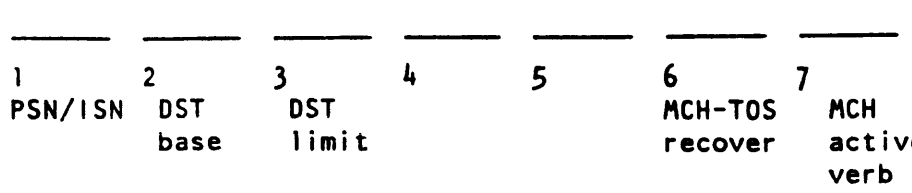
Line-count (decimal) _____

Word @B6@ contains BIL flags for an MPLII program, or line-count for a COBOL/RPG program. To use the latter, convert from Hex to a decimal number. If non-zero, this identifies, in the compile listing, the line of source code being executed. (in order for this line-count to be meaningful, or non-zero, the program must have been compiled with $line-code if COBOL, or with a 1 in column 16 of H-spec if RPG).

Part 2 concerns current task information from TCB.

If further information about the current task is required, one or more of the component parts of its task control block must be examined. The communicate parameter, data segment table, and control stack can each be examined independently.

The first step is to select the part of the TCB to be considered. Go to the beginning of the current task control block in the dump. Its address (TCB base) was found in part 1. Extract seven D-words from the dump.



PSN/ISN (D-word 1) - program and interpreter identification.

The low-order (rightmost) byte of this word is either @10@ (for an MPLII program) or @11@ (for a COBOL/RPG program). The high order (leftmost) byte of this word tells, via the slice descriptor table (SDT), where the task's program control block (PCB) is located.

The communicate parameter area (CPA) concerns the task's most recent request for MCP assistance.

A "communicate" is a request from a task for the MCP to perform a function (usually a logical I/O operation such as file open/close or a read from a disk file). Often, when a sysdump is taken, the S-interpreter is executing a communicate for the current task. For a COBOL program, the current S-op code is @2F@. The S-interpreter executes this op code by placing the request (communicate verb and object), in the communicate parameter area (CPA) within the TCB, then calling on the MCPs master communicate handler (MCH) routine to perform the requested function.

To find the CPA, add MCH active verb (D-word 7) _____

to the TCB base (found at @000B@) _____

giving the CPA address _____

The contents of the D-word at this address is _____

The high order (leftmost) byte of this word is the communicate verb (see communicate verb table 3-1 which follows for values). The low order (rightmost) byte of this word is the communicate object, usually the data segment of a file information block (FIB). For example, a value of @8203@ means mean that the MCP was requested to read (verb 82) the file thats FIB is data segment 3.

Table 3-1. Communicate Verbs

Class	Verb	Bits	Communicate (byte 0 = verb, byte 1 = object)
B	00-0F		file assignment (object = FIB segment number)
B	01		file open
B	02		file close
C	10-2F		field oriented I-O (object = segment number)
C	"	01	zip
C	"	02	display
C	"	04	pause

Table 3-1. Communicate Verbs (Cont.)

Class	Verb	Bits	Communicate (byte 0 = verb, byte 1 = object)
C	"	08	conditional
C	20		accept
D	30-3F		data communications
E	40		date-time
E	41		terminate
E	42		wait
E	43		system status
F	70-7F		machine dependent (B800)
F	71		configurator slice
A	80-9F		file type I-O (object = FIB segment number)
A	"	01	conditional communicate
A	80		test status
A	82		read (not console)
A	84		write (not console)
A	86		rewrite
A	88		delete
A	8A		stream control
A	8C		start
A	8E		overwrite
A	90		read-write
A	92		read (console)
A	94		write (console)
A	96		get
A	98		put
A	9A		redefine workarea

CONTROL STACK

The control stack is concerned with the unused returns from subroutine calls.

The control stack is used by both the interpreter and MCP during the execution of the task. The interpreter stores return addresses for subroutine jumps on its part of the stack. To some extent, the history of a program's execution can be determined from its subroutine "nesting" at a given point in time, especially if the program was written in a well-structured fashion.

For COBOL/RPG tasks, the first portion of the control stack (the interpreter's part) is also called the "perform stack." It consists of entries four D-words long with the following format:

1	2	3	4
line-count	byte offset	segment number	K

The line-count field (D-word 1) is a Hex number. Its decimal equivalent relates the return address to the source program compile listing. The return address, stored in the stack as segment number and byte offset, is used by the interpreter to find the next S-op code upon exiting from a subroutine. The K value in a stack entry is used by the interpreter to distinguish between exits from overlapping subroutines.

To extract the COBOL/RPG perform stack from a Hex dump:

add MCH-TOS recover (D-word 6 of TCB) _____

to the TCB base (found in part 1 at @000B@) _____

giving _____

subtract @01@ giving _____

Go to the resulting address in the dump and mark off one record. It represents the boundary between the interpreter's part of the stack and the MCP's part.

Next, go to the address given in D-word 3 of the TCB (DST limit). The control stack begins here, immediately following the data segment table. Extract repeated four D-word entries from the dump up to the boundary marked previously (there may be one word left over).

Line-count	Byte offset	Segment number	K
1	2	3	4
1	2	3	4

NOTE

There may be more than 10 perform stack entries but this occurs infrequently.

When analyzing the perform stack, remember that the oldest entry is at the beginning of the stack, which was written down first, while the most recent stack entry is at the bottom of the list of entries just recorded. Also, remember to convert the line-count values to decimal before applying them to a compile listing.

SECTION 4
SLICE MEMORY MAPPING

The objective is to determine which micro slices and task control slices are in memory and their sizes. This information is to be used in conjunction with the Slice Memory Mapping Worksheet to be found at the end of this manual.

Pseudo extent, micro extent, and slice extent areas of memory can be mapped by this process. Pseudo extent ranges from the addresses contained in OS registers @000F@ to @0010@. Micro extent ranges from the addresses contained in OS registers @0010@ to @0011@. Slice extent ranges from the addresses contained in OS registers @0011@ to @0012@.

Memory usage can be determined by mapping out each memory link to and from the slice descriptor table. Each memory slice begins with a pointer into the SDT. The SDT provides the slice number and length of this slice. Adding the beginning address of the slice to the slice length produces the starting address of the next memory slice.

To begin mapping memory, the address of the slice descriptor table base must be known. This base address can be found in operating system register @0017@. The start of micro extent area must also be known and can be found in OS register @0010@. By proceeding to the start of micro extent in the Hex dump, the D-word value points to the SDT entry for this slice of memory. By examining this entry in the slice descriptor table, both the slice number and slice length can be determined. (All calculations are in Hex.)

Slice number	The slice number can be calculated by subtracting the address of the SDT-base from the address of this slice's SDT entry. Dividing this answer by @05@ gives the slice number.
Address of the next slice	The address of the next slice is calculated by using information from the SDT entry for this slice. Each SDT entry is five D-words long. By adding D-word 3 and D-word 4 together the memory address of the next slice is given.

Repeating this process identifies the following memory slices:

For example:

OS register @0017@ = DCFE SDT-base
OS register @000F@ = 2218 start of pseudo slices

go to address 2218. Contents of D-word 2218 = DEED

address into SDT for this slice's SDT entry

```
calculate slice number        SDT entry =  DEED
                     subtract SDT base  =  DCFE
                              giving        1EF
                 divide by @05@ giving      63 (slice number)
slice number @63@ SPO-Console controller

Go to address DEED (pointer into SDT for this slice)
extract 5 D-words:  B810  0607  03A4  2219  0001
          D-word    1     2     3     4     5

                    D-word 3 =  03A4
             added to D-word 4 =  2219
                    giving       25BD (beginning address of the
                                       next slice)
```

repeat this process but use the beginning address of the next slice instead of OS register @000F@.

SECTION 5
PERIPHERAL - I/O DESCRIPTORS

The objective is to determine any queued I/O for a specified drive, the I/O task number, the op code issued, and, if an error, the DDP status word. This is done using the Hex dump. This information is used in conjunction with the worksheet contained in the back of this manual.

This information is obtained by examining the PHT entry for the selected device. The interrupt scan control table (ISCT) should be scanned by the port number to extract the PHT memory address. Known information for this objective must be:

device = _____ on port number = _____

Results are: I/O queued? (Y) (N)

task # = _____ op code = _____ error = _____

task # = _____ op code = _____ error = _____

task # = _____ op code = _____ error = _____

task # = _____ op code = _____ error = _____

task # = _____ op code = _____ error = _____

To begin analyzing an I/O descriptor, the interrupt scan control table (ISCT) must be located. The ISCT memory location is contained in OS register @0043@. Fill in:

The operating system register at location @0043@ = _____

The contents of this register identify the beginning memory address of the ISCT. Each entry is 1 D-word long and the 12 entries correspond to ports 1 through 12. The ISCT gives the PHT memory address for that port. A non-zero entry identifies which ports have been warmstarted. Locate the memory address found in the ISCT register @0043@ and fill in the following blanks:

Port	Address	Port	Address
1	_____	7	_____
2	_____	8	_____
3	_____	9	_____

Port	Address	Port	Address
4	_____	10	_____
5	_____	11	_____
6	_____	12	_____

Extract the PHT memory address from the ISCT corresponding to the port number. This address identifies the beginning of this device's entry in peripheral handler tables (PHT) 5-1 and 5-2. Each PHT entry varies in length depending upon the device. The first D-word can be used to verify the port number and the device kind, this is defined as abcd where ab equals port number (zero relative) and cd equals device kind.

Then add @01@ to the PHT address from above giving _____.

This result identifies the memory address of the device controller for this device.

Then add @03@ to the PHT address from above, giving _____.

This result identifies the maximum queue number and the current queue number. The maximum queue number tells how many devices are on this port. The current queue number identifies which of these devices is in use. For example, there can be a 100 TPI drive and a 18 MB fixed disk configured on the system (DKA, DKB, DFC, and DFD). The maximum queue number equals 3 and, if the current queue number equals 2, then DFC is the unit to which the I/O has been directed.

Next, add @02@ to the PHT address from above, giving _____.

This result identifies the I/O descriptor's memory address. A value of @0000@ indicates that no I/Os are queued. A non-zero value points to the memory address of the descriptor. The I/O descriptor is six D-words long. Proceed to this address and extract four D-words as follows:

_____ _____ _____ _____ (queued I/O descriptor)
1 2 3 4

If additional I/Os are queued for this device, D-word 2 points to the next I/O descriptor or back into the PHT table for the device.

D-word 1 is used to identify the task number and op code. It is defined as abcd where ab equals interlock and cd equals op-code.

Convert interlock into bits = _ _ _ _ _ _ _ _
7 6 5 4 3 2 1 0
a b

bit 7 = 0 - peripheral handler finished
bit 6 = 1 - key not found (DK)/end-of-file (MT)
bit 5 = 1 - permanent error
bit 7 = 1 - in use by peripheral handler
bit 6 = 1 - task waiting I/O

bit 5 = 1 - waiting for W.S.B.
bit 4 - 1 = task number = mix number - 1
bit 0 = 1 - I/O needed
= 0 - I/O no longer needed

Op code is 1 byte and defined as:

	Disk	Tape	LP	Card
00	read	read	n/a	read
01	write	write	write	n/a
02	indirect read 5-byte key	rewind	n/a	n/a
42	indirect read 13-byte key	n/a	n/a	n/a
82	indirect read 21-byte key	n/a	n/a	n/a
C2	indirect read 29-byte key	n/a	n/a	n/a
03	write with raw check	write TM	n/a	n/a
04	firmware load	backspace	n/a	n/a
05	zero disk	erase	n/a	n/a
06	directory search	read status	n/a	n/a
07	read with no data or allocate	n/a	n/a	n/a
08	n/a	search for TM then read	n/a	n/a
0A	n/a	search for TM	n/a	n/a
0B	n/a	write TM then write data	n/a	n/a

The op code for label processing is:

80 label read
81 label write
83 console label write

If an error occurred, D-word 3 may contain an error code defined as abcd:

where ab = 80 - character count (MT)/EOF (CRD)
= 81 - parity error (MT,DK)/read error (CRD)
= 82 - timeout/seek error
= 83 - address error
= 84 - illegal rewind (MT)
= C0 - end-of-file (CRD)/no file
= 85 - end-of-tape (MT)
= 86 - write inhibit
= C7 - control card other than end (CRD)

For card (A9114) cd equals hardware status:

cd = 10 - early feed cell

cd = 08 - leading edge cell
   = 04 - trouble
   = 02 - no feed
   = 01 - ready

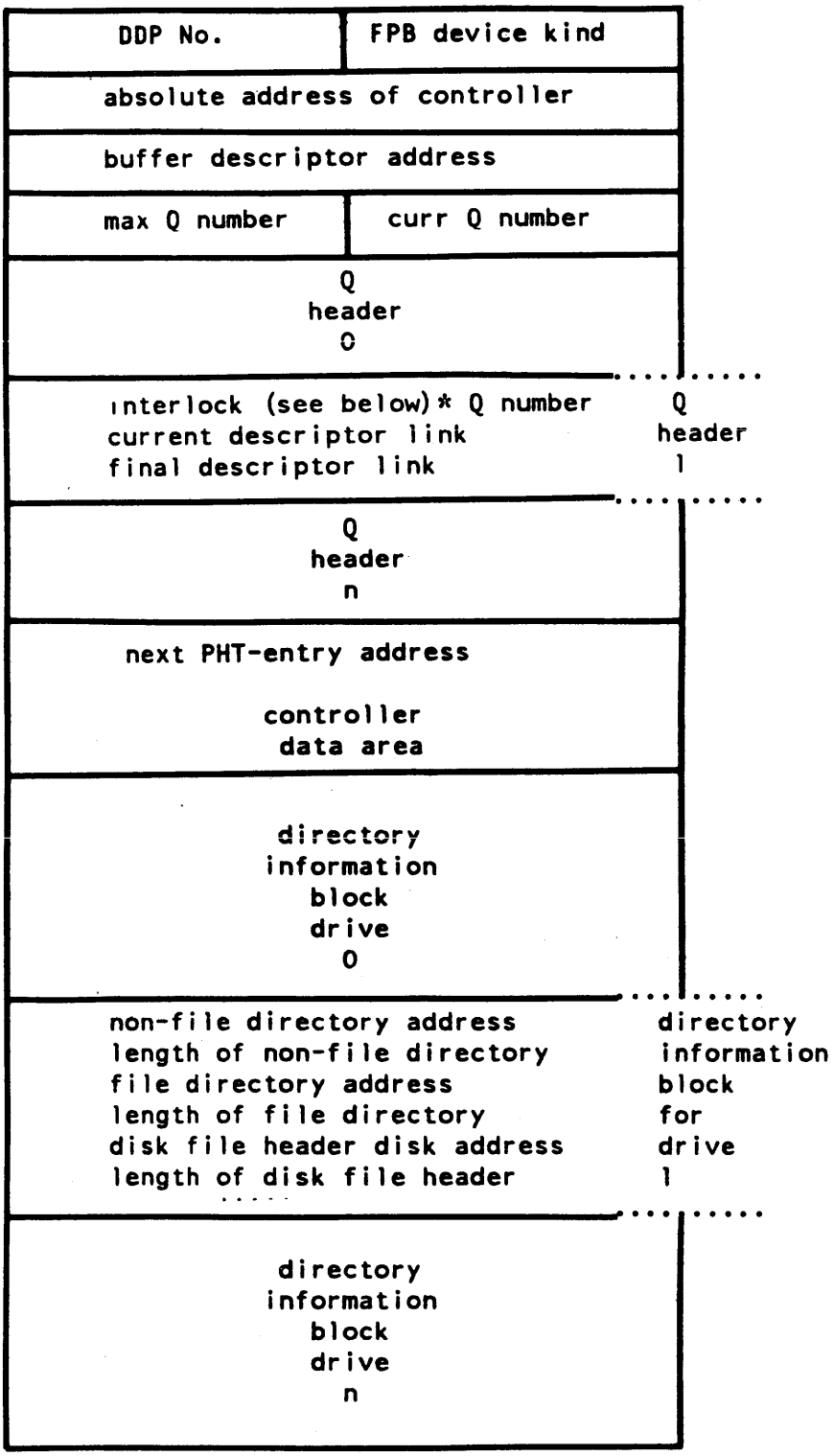
For line printers, D-word four should be converted into bits as follows:

$\overline{15}\ \overline{14}\ \overline{13}\ \overline{12}\quad \overline{11}\ \overline{10}\ \overline{9}\ \overline{8}\quad \overline{7}\ \overline{6}\ \overline{5}\ \overline{4}\quad \overline{3}\ \overline{2}\ \overline{1}\ \overline{0}$

bit 0 = 1 - execute specified format control after print
bit 1 = 1 - execute specified format control before print
bit 2 = 1 - execute specified format control only
bit 4 = 0 - vertical tab with ...
     bit 5 = 0 - top of form
     bit 5 = 1 - channel number in bits 8 - 15
bit 4 = 1 - line advance with ...
     bit 5 = 0 - advance 1 line.
     bit 5 = 1 - count in bits 8-15

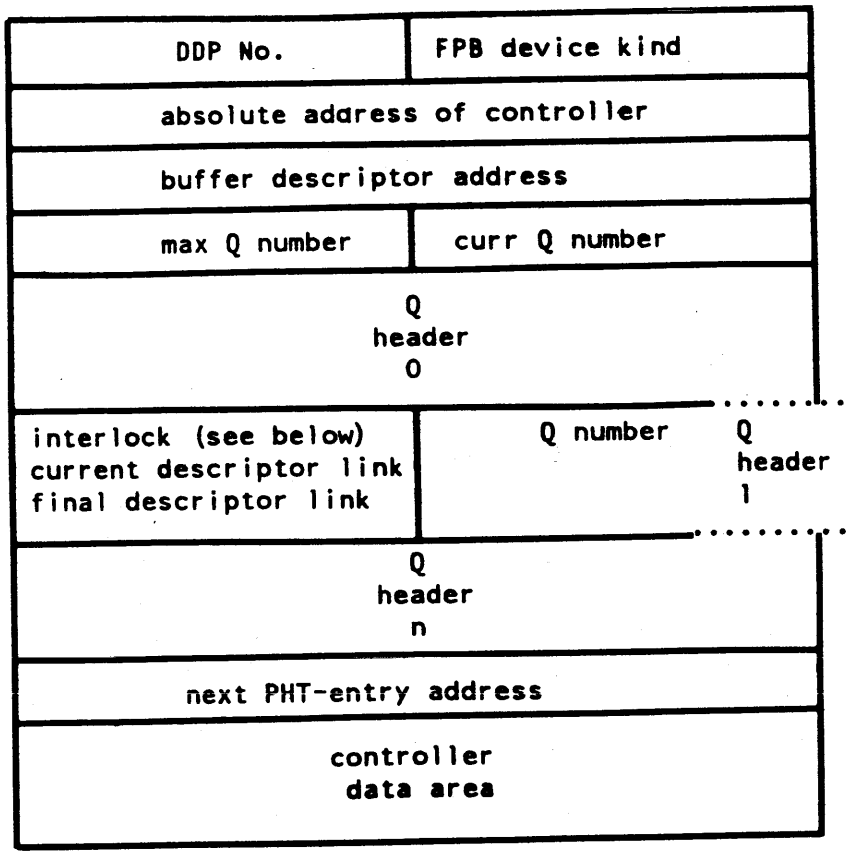
Note: channel number is 0-11 for channels 1-12

Table 5-1. PHT (Peripheral Handler Table) Disk



NOTE: One line = one D-word entry.
See Q header interlock table

Table 5-2. PHT (Peripheral Handler Table) Non Disk



note: one line = one D-word entry.

Q header interlock table (this description covers all PHTs)

bit 15 = 0	Q header
bit 14 = 1	Q not empty
bit 14 = 0	Q empty
bit 13 = 1	current status is ready
bit 13 = 0	current status is not ready
bit 12 = 1	last status is ready
bit 12 = 0	last status is not ready
bit 11 = 1	waiting help task
bit 10 = 1	CTM required
bit 9 = 1	timeout in progress
bit 8 = 1	timeout first part
bii 7 = 1	EOF
bit 6 = 1	device assigned by CTM
bits 5, 4, 3 are undefined	
bits 2, 1, 0 are Q number	

SECTION 6
ANALYSIS OF MCP TRAIL

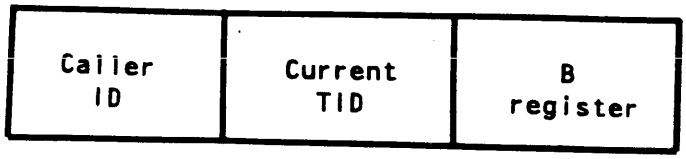
The objective is to examine the MCP trail up to the time of the clear/start.

The MCP trail consists of 32 entries with each entry being two D-words long. The trail is circular in fashion; that is, after entry 256 has been made, the next available entry is back at the starting point of the trail overlaying any previous entry. Two registers are used to help examine the MCP trail. Contents of register @0014@ point to the beginning of the trail. The content of register @0454@ is a relative pointer into in the trail which can identify where the next entry will be made:

Add	_____	OS register @0014@
to	_____	OS register @0454@
giving	_____	the next available entry.

Subtracting @02@ from this answer identifies the last entry into the trail.

The format of the trail is as follows:



Current TID = current task # = current mix # - 1

Caller IDs = 01, 02, 03, 04 or 05

B register is dependent upon caller ID.

Caller ID = 01 - branch-find B register = MAD

Micro program address descriptor (MAD) entries can be of two types:

1. Absolute memory address.

2. Offset and slice number.

The distinction between the two types of MADs is made by converting the B register to bits (bit numbering - right to left, starting with 0). If bit 15 = 1, then MAD = absolute memory address = 0, then MAD = offset and slice number.

The absolute memory address is the value represented by bits 14-0.

The offset and slice number can be evaluated by converting the B register into bytes. The leftmost byte equals the offset, and the rightmost byte equals the slice number. In addition, caller ID equals 02 minus MC timeout; and the B register equals PHT entry address.

By going to the PHT entry address and examining its value, the device port number and device kind experiencing the timeout can be identified. For example:

Caller ID = 03 - MC timeout B register = max Q #/current Q #

The maximum and current queue numbers are part of the PHT entry. The maximum queue number tells how many devices are on this port. The current queue number identifies which of these devices is in use. For example, a 100 TPI drive and a 18 MB fixed disk could be configured on the system (DKA, DKB, DFC, and DFD). The maximum queue number is 3 and, if the current queue number equals 2, then DFC is the unit experiencing the timeout. (Queue numbering is relative to zero.) Next:

Caller ID = 04 - MC-Q-loop. B register = Q header address

If a device should have an I/O queued because the unit is Not Ready, when the unit is made ready, the queue header address points to the memory address of the queued I/O descriptor. The MCP trail reflects this change of status with this entry. Next,

Caller ID = 05 - PH-IRQ B register = device status

The device status can be examined to determine the actual condition of a device from the MCP point of view. Selected devices are defined below. The device status should be converted to bits. Number the bits from right to left starting with 0.

Printer:	bit 0 - not ready	bit 5-7 - not used
	1 - end of page	8-12 - port # (0 relative)
	2-3 - not used	13-14 - not used
	4 - print complete	15 - data request, IOC ready to receive
Disk:	bit 0 - search good	bit 6 - parity error
	1 - write inhibit	7 - timeout
	2 - seek error	8-12 - port number (0 relative)
	3 - illegal address	13 - read after write check
	4 - file not open	14 - index mark
	5 - seek incomplete	15 - sector mark
SPO:	bit 0 - not ready	bit 5-7 - not used
	1 - input request	8-12 - port number (0 relative)
	2 - end-of-message	13-14 - not used
	3 - error	15 - data request
	4 - service too late	

Console as data entry:

bit 0 - carrier stall	bit 6 - forms ready
1 - over speed	7 - carrier buffer ready
2 - interrupt not honored	8-12 - port number 0 relative)
3 - ready	13 - not used
4 - keyboard ready	14 - end-of-paper
5 - printer ready	15 - data request, ready to receive data

DEK as entry station:

bit 0 - buffer full	bit 5 - not used
1 - parity error	6-7 - station address
2 - not used	8-12 - port # (0 relative)
3 - framing error	13-14 - not used
4 - service too late	15 - exception bit

Tape DMAC (PE):

bit 0 — not ready	bit 6 — tape mark
1 — busy or RWD	7 — CER
2 — file protect	8-12 — port (0 relative)
3 — EOT	13 — 1600/800 BPI
4 — BOT	14 — tape error
5 — blank tape timer	15 — op successful

Disk Pack:

bit 0 — link parity error	bit 5 — local
1 — time out error	6-7 — MTR use
2 — read after write error	8-12 — port (0 relative)
3 — search greater/equal	13-15 — MTR use
4 — search good	

Note

Link parity is a parity error on a data transfer from the B 9387 to the B 800.

Time out is a no response from the B 9387 to a command by the B 800.

The entries in the MCP trail for disk pack do not reflect drive related errors,such as a seek error. They only reflect errors in communication between the B 9387 and the B 800, results of search operations, results of read after write operations, and of the B 9387 going to a local state. These are all DDP oriented errors. If a disk pack drive is the suspected problem area, the system log and any related SPO messages must be used to determine any drive error related information.

SECTION 7
SLICE DESCRIPTOR TABLE OVERVIEW

The slice descriptor table (SDT) is an MCP table located in the upper portion of memory. It keeps track of the location of the following structures for the MCP:

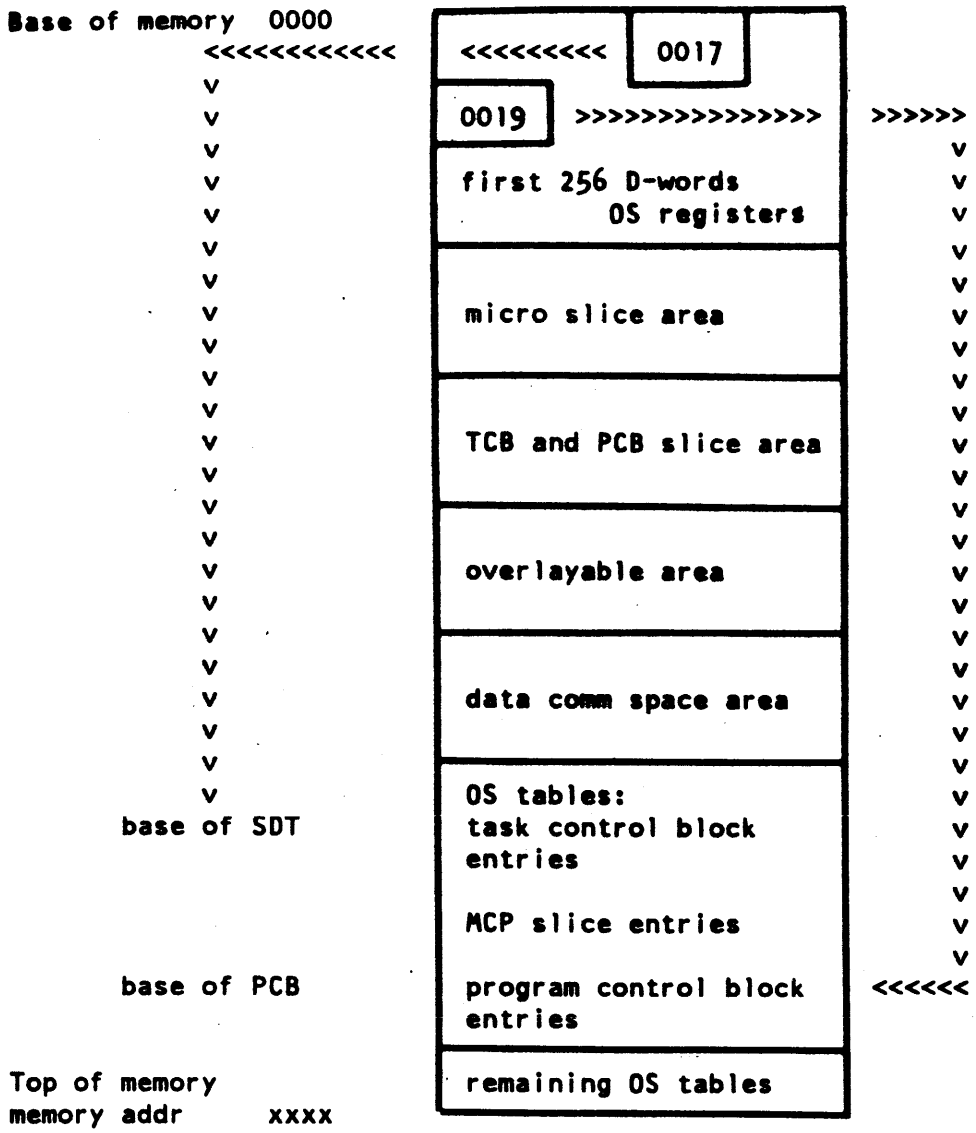
1. The task control blocks (TCB) for each mix number.
2. The micro slices of the MCP.
3. The program control blocks (PCB) for each user task.

The slice descriptor table is created at warmstart/restart time from information passed from warmstart to the OS-build module of the MCP. The SDT is created using either the warmstart configuration entered by the operator or a restart configuration entry located on the current system disk if the system is brought up by the restart or clear/start.

The SDT has an entry for each possible user task (nine entries for nine user tasks); seven entries for the MCP independent runners, a variable number of entries for all the MCP slices used in the system while running, and nine entries for each possible user program control block. There is a pointer to the absolute D-word location of the SDT base located in the operating system registers at location @0017@. There is a second pointer to the base of the program control block section of the SDT located in the OS registers at location @0019@.

The purpose of the slice descriptor table is to point to the current absolute position of the TCBs, MCP slices, and PCBs either in memory or disk. The memory management flags located in the first byte of each 10-byte entry (five D-words) in the SDT indicate that the slice, TCB, or PCB is either in memory or on disk. The various MCP virtual memory management routines use this table to maintain the current location of these structures.

The following illustrates the pointers to, location of, and internal information concerning the slice descriptor table (SDT).



Format of a slice descriptor table entry.

Byte	0	1	2	3	4	5	6	7	8	9
	mm flags	disk flag	disk address		length		memory base		pointers/ counters	
D-word	0		1		2		3		4	

length = 10 bytes / 5 D-words

MM flags	byte 0	Flags which determine type of slice and location (disk or memory).
Disk flag	Most significant bit of most significant digit of byte 1	Flag which, when set, indicates slice is on a disk pack (system disk is a pack).

Length	bytes 4 and 5	Slice length in D-words.
Memory base	bytes 6 and 7	The absolute D-word memory address of the base of the slice; if MM flags indicate that the slice is in memory.
Pointers/ counters	bytes 8 and 9	The base is a TCB or a device controller slice if TCB - most significant byte is slice number of program parameter block. Least significant byte is slice number of interpreter slice. If slice is not a TCB, both bytes combine to give a user count for the slice, if it is a device controller.

MEMORY MANAGEMENT FLAGS FOR SDT

The only structures described in the slice descriptor table are:

1. Micro slices.
2. Controller micro slices.
3. Ask control blocks.
4. Program control blocks.

The value of the memory management flags for these structures depends on whether the slice is in main memory, page memory, or absent from memory:

location in...	Micro slice	Controller	TCB	PCB
memory	@A8@	@B8@	@C2@	@E2@
page	@AC@	@BC@	@C6@	@E6@
absent	@28@	@38@	@42@	@62@

The detailed structure of the slice descriptor table (SDT) is:

1. The first nine entries are locations for up to nine task control blocks for user tasks. If an entry is not in use, it should consist of all zeros. If in use, a valid entry is located in the same relative position as the task mix number. (For example, position 1 equals mix number 1, position 9 equals mix number 9).

2. Seven entries are for the MCP independent runner tasks in mix number order. That is:

   mix 10 = super utility (SU)

   mix 11 = system logging (LOG)

   mix 12 = data comm space handler

   mix 13 = SCL loader

   mix 14 = help task

   mix 15 = working set bailiff

   mix 16 = data comm queue handler task

3. The total complement of basic slices used by the system is given. Some are permanently stored in the operating system nucleus; for example, the controllers for disk and SPO. The rest are brought into micro memory slice area as they are needed or, if they are controllers, when the device is opened for input/output. In any case, all possible slices, known or warmstarted, have positions assigned in the remaining area of the slice descriptor table.

4. The last 10 positions of the slice descriptor table are the program control blocks for super utility and up to nine user tasks. There is a separate pointer to the base of these 10 positions located in the operating system registers at @0019@. The first PCB entry in this area is for the system super utility, followed by the nine entries for the user PCBs. An all-zeros entry indicates the PCB is not in use.

SECTION 8
MCP SLICES

Table 8-1 is a list of the slice descriptor table entries. The length may change due to recompilation or patches.

Table 8-1. Table of MCP Slices

Slice Numbers Hex	Dec	name	Length (D Words) Hex	Dec	Remarks
0	0	TCB slice mix # 1	varies		
1	1	TCB slice mix # 2	varies		
2	2	TCB slice mix # 3	varies		
3	3	TCB slice mix # 4	varies		
4	4	TCB slice mix # 5	varies		
5	5	TCB slice mix # 6	varies		
6	6	TCB slice mix # 7	varies		
7	7	TCB slice mix # 8	varies		
8	8	TCB slice mix # 9	varies		
9	9	Super utility TCB	varies		
A	10	Logger TCB	varies		
B	11	DC-space-handler TCB	varies		
C	12	SCL/loader TCB	varies		
D	13	Help task TCB	002E	0046	
E	14	WSB task TCB	0023	0035	
F	15	DC queue handler TCB	varies		
10	16	BIL-SI			BIL/MPLII interpreter
11	17	COBOL-SI			COBOL/RPG interpreter
12	18	COBOL-INT-slice	0229	0553	
13	19	BIL-auten	0237	0553	
14	20	Open-close	01AE	0430	
15	21	Printer-open	04A5	1189	
16	22	Printer-close	01E4	0484	
17	23	Card-open	025E	0606	
18	24	Card-close	0121	0289	
19	25	Tape-open-1	026B	0619	
1A	26	Tape-close-1	0301	0769	
1B	27	Disk-open-1	0213	0531	
1C	28	Disk-close-1	0377	0902	
1D	29	Debug-COBOL-SI	1909	6409	
1E	30	Class-D-slice	1C2F	4815	MICR subsystem
1F	31	Date-time	02A3	0675	
20	32	Accept-slice	02A2	0674	
21	33	Alloc-slice	02CE	0718	

Table 8-1. Table of MCP Slices (Cont.)

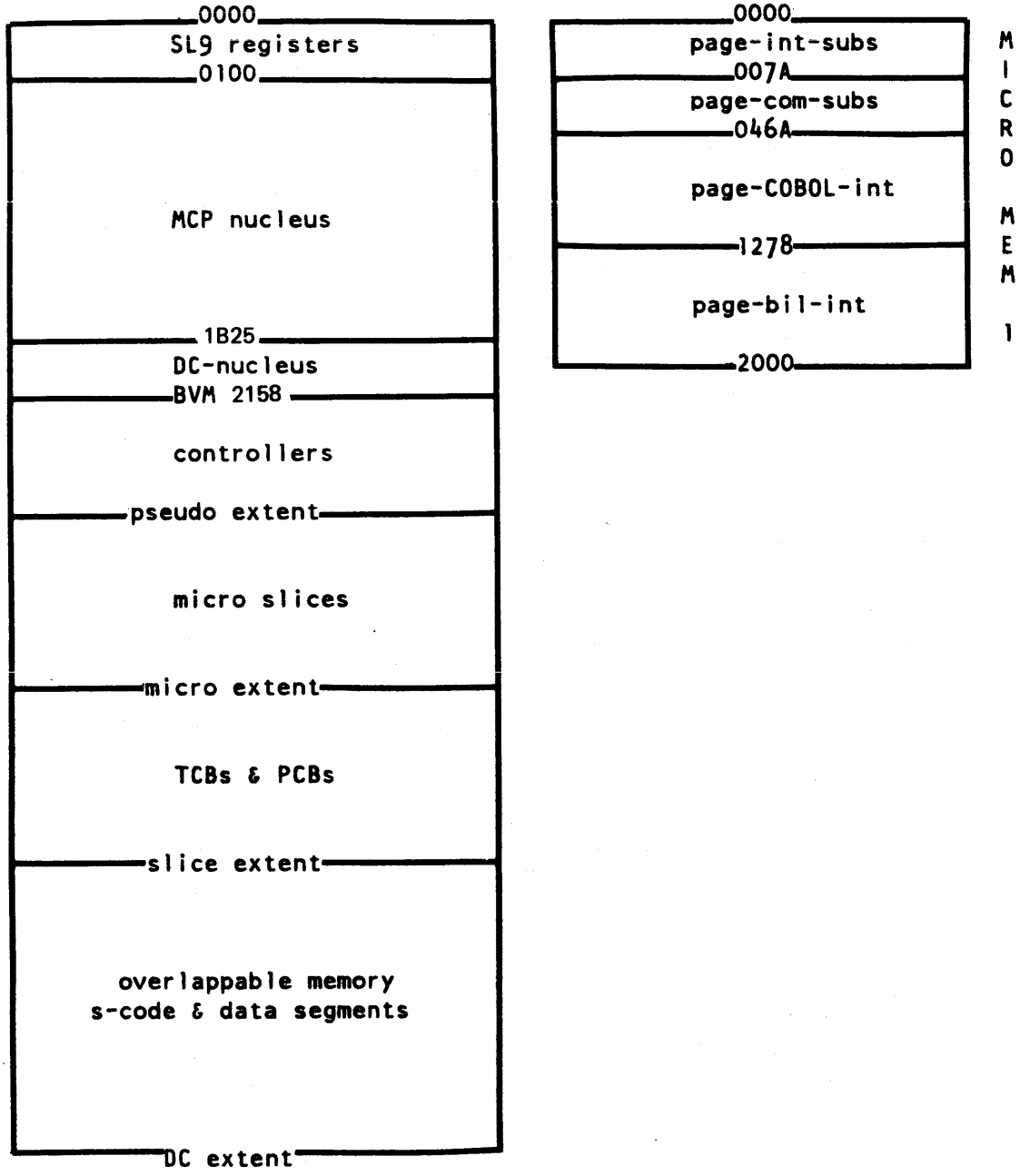
22	34	OS-to-SCL	0174	0372	
23	35	MH-slice	01C0	0448	
24	36	RTC-slice	0171	0369	-RTC controller
25	37	D/T-comm-slice	0141	0321	date/time request chg
26	38	DI-slice	0317	0791	MX function
27	39	Zip/display-slice	0298	0664	
28	40	EOJ-slice	025A	0602	
29	41	DC-slice			
2A	42	Loader-slice-#1	042A	1066	
2B	43	Loader-slice-#2	03D1	0977	
2C	44	Loader-slice-#3	02D1	0727	
2D	45	Loader-slice-#4	0322	0802	
2E	46	Loader-slice-#5	0123	0291	
2F	47	DC-space	00AA	0170	
30	48	DS-slice	01FF	0311	
31	49	RTC-date-fix	009B	0155	
32	50	Stop-slice	01A9	0425	
33	51	Go-slice	0154	0340	
34	52	Power-off	0206	0518	
35	53	OL-slice	0260	0621	
36	54	AD-slice	036C	1004	
37	55	DMAC-disk-slice	0178	0376	-disk controller
38	56	SPO-slice	01FD	0509	-spo controller
39	57	DC-ch-30	03D2	0978	
3A	58	DC-ch-31	032C	0812	
3B	59	DC-ch-32	03A9	0937	
3C	60	DC-ch-33	03AA	0938	
3D	61	DC-nucleus	0633	1587	
3E	62	DC-task-EOJ-slice	006B	0107	
3F	63	DC-loader	027F	0639	
40	64	Help-task-slice	03E1	0993	
41	65	Working-set-bailiff	02E4	0740	
42	66	Clear-start	0F92	3986	
43	67	Configurator	0083	0131	
44	68	OS-tracer	017C	0380	
45	69	DC-EOJ-slice	0238	0568	
46	70	Reconstruct	0165	0357	
47	71	RY-slice	0121	0289	
48	72	XD-slice	0397	0919	
49	73	DC-sh-slice	00AA	0170	
4A	74	DC-eh-slice	0238	0568	
4B	75	DC-LDR2-slice	01E2	0482	
4C	76	DC-LDR3-slice	0117	0279	
4D	77	DC-LDR4-slice	0312	0786	
4E	78	AD-disk	0000	0000	
4F	79	Indexed-slice	0B31	2865	
50	80	PE-table	0201	0513	
51	81	DC-no-DCOM-slc	0053	0083	
52	82	Disk-open-2	02DA	0730	
53	83	Disk-open-3	049A	1178	
54	84	Disk-close-3	0498	1176	
55	85	Tape-open-2	02A5	0677	

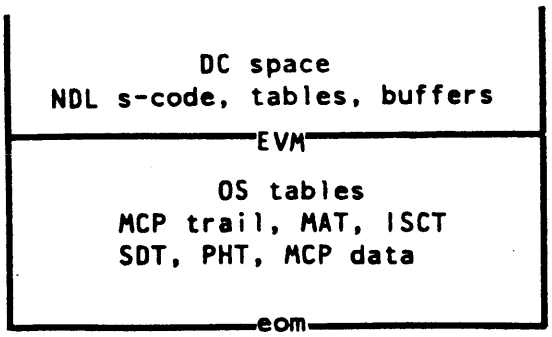
Table 8-1. Table of MCP Slices (Cont.)

56	86	Tape-open-3	0206	0694	
57	87	Tape-close-2	02B6	0694	
58	88	Tape-close-3	013A	0314	
59	89	EOJ-slice-2	0200	0512	
5A	90	EOJ-slice-3	021F	0543	
5B	91	Dealloc-slice	0277	0631	
5C	92	OS-susp-slice	0295	0661	
5D	93	Stream-slice	0251	0593	
5E	94	Index-open-1	027A	0634	
5F	95	Index-open-2	0237	0567	
60	96	Index-open-3	027B	0635	
61	97	Index-open-4	02F9	0761	
62	98	Index-close-1	0323	0803	
63	99	SPO-console	03C6	0966	
64	100	Interp-subs	000F	0015	
65	101	Console-slice	075D	1885	console controller
66	102	Index-open-5	024A	0586	
67	103	Sort-slice-1	0416	1046	
68	104	Sort-slice-2	0587	1415	
69	105	Sort-slice-3	0000	0000	
6A	106	DC-ch-34	023D	1411	
6B	107	DC-ch-redef4	0368	0872	
6C	108	DDE-SPO	049A	1178	-DDE SPO controller
6D	109	DDE-con	0600	1536	-DDE cons controller
6E	110	Log-task-slice	04DA	1242	
6F	111	SSC-slice	02D6	0726	
70	112	Disk-pack-ctlr	021B	0539	-disk pack controller
71	113	Large-scrn-SPO	01DE	0478	-TD830 SPO controller
72	114	FD-slice	020C	0524	
73	115	Int-mul-div	029D	0669	
74	116	DC-pre31-load	02C6	0710	
75	117	R/W-console	026C	0620	
76	118	Log-write-slice	0119	0281	
77	119	Pack-nuc-end	00A7	0167	
78	120	Any warmstarted controllers are loaded			
thru		in this area of the slice descriptor table.			
xx	xxx	These controllers are any device other than the disk and the SPO controllers			
xxx	xx	PCB entries			
+1	+1	first PCB	varies		SYS-SUPERUTL
+2	+2	second PCB	varies		
+3	+3	third PCB	varies		
+4	+4	fourth PCB	varies		
+5	+5	fifth PCB	varies		
+6	+6	sixth PCB	varies		
+7	+7	seventh PCB	varies		
+8	+8	eighth PCB	varies		
+9	+9	ninth PCB	varies		
+10	+10	tenth PCB	varies		

SECTION 9
B 800 MEMORY LAYOUT

The following is a description of the B 800 memory layout:





Related notes:

1. Beginning of virtual memory is 1B25 if data comm is not present and 2156 if data comm is present.

2. DC extent equals end of virtual memory (EVM) if data comm is not present.

3. The only absolute addresses are SL9 registers, MCP nucleus, DC nucleus, and page interpreters. All other addresses vary depending on the program mix and the hardware configuration.

4. Page memory and micro memory are both 8K words. When the MCP is in control of the system, page memory constitutes the first 8K words of main memory. When an interpreter is in control of the system micro memory 1 constitutes the first 8K words of main memory. Micro memory 1 is treated as read only memory and is never written to. Switching between the two memories is performed by a string of micro code as control is being transferred.

SECTION 10

MCP NUCLEUS

The MCP nucleus basic routines (by absolute memory address) are given in this section. They are:

010C General routines.
Read/write memory, common routines used by controllers, and common register manipulations.

01C3 Byte handling routines.
Manipulation of fixed format tables (such as PPB, FPB).

02AC Access/manipulate OS structures.
Binary multiple, divide, and modulo.
Accepts a slice number and returns an absolute memory address. Locates desired OS system buffers and manipulates TCB stack. Works with debug trail and link checking.

04D0 OS EPAR.
This is the central task switching routine of the system. The MAT is scanned for the highest priority task which can be run. If none exists, then service I/O interrupts and continues to scan. If a task is found, then the OS state alters to reflect the task and the CPU is switched.

05E7 Interpreter common subs.
Used when control is given up by the interpreter and back to the MCP.

05F6 Virtual memory.
Get/put micro slices.
Get/put pseudo slices.
Get/put space.
Expand/shrink slice.
Get/put segments.
DC get/put space.
Get/put DC nucleus.
Thrashing detection (08B2 - 08C2)
Garbage man (08C3 - 08F1)
Cleans up the available chain from slice extent to DC extent after all gets, puts, and detected thrashing conditions.

0A38 Working set bailiff.
Resident portion of WSB.
It cleans micro memory of all micro slices except controllers. This feature is used if there is insufficient memory space available for the IR WSB. After space has been made available, the IR WSB is loaded.

0A6E Master interrupt processor (MIP).
MIP interfaces with all device controllers and the peripheral handler. Control is transferred to the I/O controller if an interrupt needs service.

0AE1 Media controller (MC).
MC recognizes media insertion. It also facilitates device timeout for recovery from DDP/device limitations. The timeout logic in the various controllers is device/operation dependent. Controllers for devices that do not require timeout servicing contain their own timeout logic. MC initiates automatic retry of I/O after correction of a not ready condition has been encountered.

0C50 Peripherial handler (PH).
PH handles the I/O interface with read/write, open/close, virtual memory, and loader. It inserts the buffer descriptor into the queue and initiates the I/O if possible. It also removes the buffer descriptor from the queue and initiates the next descriptor. PH optimizes the seeks for disk. By comparing the disk addresses in the buffer descriptors queued, new I/Os are added to the queue in the correct slot in an attempt to keep the disk arm moving in one direction. PH halts I/Os if VM demands control to slide memory affecting buffers, FIBs, and so on. The I/O is resumed once the VM operation is complete.

0DCA SCL handler.
This routine expands the SCL-LDR-TCB slice by 268 bytes to create an I/O descriptor and buffer for SCL input. Once space has been obtained, the I/O is initiated. Preceding blanks are removed from the message until the first alpha character. SCL now scans through the legal SCL op table for a valid verb to obtain the op code. It also checks for communication from SYS-SUPERUTL and for MICR functions (GR, GK, NR). Based on the SCL entry, control passes to the appropriate slice: (DI-slice, OL-slice, RY-slice). After the function is complete, the SCL buffer area is removed by VM shrink and control is given back to MCH.

0F65 Master communicate handler (MCH).
It is the function of this routine to interrogate the verb of a CPA and cause the correct communicate class to be alerted. Minimal syntaxing of the verb is performed based on value range. Further decoding is done by the class decode.

17D9 Resident logger.
This routine moves SCL messages to the log message queue. It also transfers I/O counts from the logger TCB to the log files.

1A1F Clear/start.
Performs soft DPM read error, tests for parity errors, and writes memory contents to DMFILOO.

1AC9 Ninty D-words reserved for nucleus patch area.

1B24 End of nucleus.

1B23 Interp-trail
These two words are used by the interpreters to store info when they leave and return to the 2 MHZ page. On leaving, they update the first word with the operating system destination and zero out the second word. On returning, they write the page destination into the second word. If word two is zero at the time of a clear/start, execution was in the MCP. If word two is non-zero, execution was in the interpreter.

1B25 End of nucleus.

SECTION 11
DESCRIPTION OF MICRO SLICES

The following is a brief description of the slices pointed to in the slice descriptor table. Some functions are segmented into multiple slices. For example, tape open is performed by tape-open-1, tape-open-2, and tape-open-3. This is done to minimize the amount of micro memory required to perform a function, as only one of the slices of a multiple slice function is required to be in memory at any one time.

Hex/Dec

0A/10 Logger TCB
This is the task control block for the independent runner portion of log task. The data segments are:

0. Log message queue.
   Size is 572 D-words.
1. Log file FPB.
   Size is 32 D-words.
2. Log file FIB.
   This FIB is for the log file marked as active.
3. Log file 2 FIB.
   This FIB is for the log file marked as next active.
4. Log file work area.
   Size is 90 D-words.

0B/11 DC-space-handler TCB
This is the task control block for DC-space-handler slice and is also used by the data comm loader. The data segments are:

0. DC-loader FPB for NDLSYS.
   Size is 32 D-words.
1. DC-loader FIB for NDLSYS.
2. DC-loader FPB for NDL-interpreter for DCP 0.
   Size is 32 D-words.
3. DC-loader FIB for NDL-interpreter for DCP 0.
4. DC-loader FPB for NDL-interpreter for DCP 1.
   Size is 32 D-words.
5. DC-loader FIB for NDL-interpreter for DCP 1.
6. DC-loader space handler area.
7. DC-loader FC area.

0C/12 SCL-loader TCB
This is the task control block for the independent runner

portion of the SCL-loader. It is also used by many other slices of the MCP. The data segments are:

0. Configuration table.
1. Task detail table.
2. SPO output buffer.
3. Error message table.
4. Initiating message input buffer.
5. Loader FPB for code file.
6. Loader FIB for code file.
7. Loader FPB for virtual memory file.
8. Loader FIB for virtual memory file.
9. Six-B-tab buffer.
   Work buffer used by many slices.
10. Loader getseg-ppa2 buffer.
    Used by loader for DST, CST, CCB, and TCB working storage area.
11. Message-group-1.
    OS message table.
12. Message-group-2.
    Interpreter message table.
13. Message-group-3.
    Data com message table.
14. SCL work buffer.
15. Trace buffer.
16. Allocate-deallocate work buffer.
17. Open-close work buffer 1.
18. Open-close work buffer 2.
19. Indexed open work buffer.
20. Power off work buffer.
21. Through 29. Sort work buffers.
    One work buffer for each mix number 1 through 9.
30. Through 40. Indexed key work area.
    One area for each mix number 1-11 for key building.
41. System status work buffer.
42. Work area for sysconfig file.
43. FIB for sysconfig file.
44. FPB for sysconfig file.

0D/13 Help task TCB
Task control block for the independent runner help task. The data segments are:

0. HT-card-BD-SD.
   I/O buffer for help task.
1. HT-label-SD.
   Buffer used for label syntax.

0E/14 WSB task TCB
Task control block for the independent runner portion of working-set-bailiff. This TCB has no data segments.

0F/15 DC queue handler TCB
Task control block for the data comm result queue processor,

which is part of the DC-nucleus. This TCB has no data segments.

10/16 BIL-SI
This slice is the interpreter for BIL and MPLII programs. It is loaded at warmstart time to micro memory 1 (logical page) and is not overlayable. The slice descriptor shows an absolute memory in mml but no slice length.

11/17 COBOL-SI
This slice is the interpreter for COBOL and RPG programs. It is loaded at warmstart time to micro memory 1 (logical page) and is not overlayable. The slice descriptor shows an absolute memory in mml but no slice length.

12/18 COBOL-INT-slice
Interprets COBOL "compare for class" and RPG "move sign" constructs. This slice was part of COBOL-SI prior to 3.1 release.

13/19 BIL-auten
processes MPLII data comm authenticator communicates.

14/20 Open-close
This slice is the common entry point to all open and close communicates. It checks the communicate for syntax and decodes it.

15/21 Printer-open
Processes open communicates for line printers, serial printers, console, and keyboard files.
Provides:
1. Device allocation.
2. FIB construction.
3. Priming of the device.

16/22 Printer-close
Processes close communicates for line printers, serial printers, console, and keyboard files.
Provides:
1. Completion of outstanding physical I/Os and label writing.
2. FIB removal.
3. Release of hardware device (not done if half close).

17/23 Card-open
Processes open communicates for all card devices.
Provides:
1. Device allocation.
2. Construction of the FIB.
3. Priming of the device.

Hex/Dec

18/24 Card-close
Processes close communicates for all card devices.
Provides:
1. Completion of outstanding physical I/Os and label writing.
2. Removal of the FIB.
3. Release of hardware device (not done if half close).

19/25 Tape-open-1
First phase of tape open communicate processing.
Provides:
1. Location of FPB and configuration table.
2. Decode of FPB parmeters.

1A/26 Tape-close-1
First phase of tape close communicate processing.
Provides:
1. Completion of outstanding physical I/Os.
2. Decode of close parmeters.
3. Label writing.
4. Handling of change reel.

1B/27 Disk-open-1
First phase of disk open communicate processing for all types of disk files except indexed.
Provides:
1. Syntax of open verb.
2. Routines for error handling during open.

1C/28 Disk-close-1
First disk close communicate processing phase for all types of disk files except indexed.
Provides:
1. Completion of outstanding physical I/Os.
2. Decoding of close parameters.

1D/29 Debug-COBOL-SI
This slice is used by engineering for development. The final release MCP file does not contain any code for this slice.

1E/30 Class-D-slice
PSL interpreter.

1F/31 Date-time
Processes operator requests to change either the date and time that was entered at warmstart or a prior DT message.

20/32 Accept-slice
Processes AX messages for SYS-SUPERUTL and application programs.

Hex/Dec

21/33 Alloc-slice
Gains disk space from the available table and updates the disk file header.

22/34 OS-to-SCL
Formats SPO messages for fatal errors to a task (DS or DP messages).

23/35 MH-slice
Formats all SPO messages except DS or DP and clear/start.

24/36 RTC-slice
Controller for real time clock.

25/37 D/T-comm-slice
Processes date and time request communicates from application programs.

26/38 DI-slice
Processes operator requests for MX.

27/39 Zip/display-slice
Processes zip and display request communicates from application programs.

28/40 EOJ-slice
First phase of end-of-job processing.
Provides:
1. Interface to DS or DP SCL commands.
2. Processing of terminate communicates.
3. Processing of abort program loads.

29/41 DC-slice
This slice attempts to get a data comm buffer, move data from a SPO buffer into it, and enqueue the message to the MCS queue.

2A/42 Loader-slice-1
First phase of processing a load program request.
Provides:
1. Determination if request is part of load utility or SYS-SUPERUTL.
2. Opening of code file.
3. Updating of slice descriptor table.

2B/43 Loader-slice-2
Second phase of processing a program load request.
Provides:
1. Gains in disk space for virtual memory file.
2. Updating of task detail table.
3. Updating of mix attribute table.

Hex/Dec

2C/44 Loader-slice-3
Third phase of processing a program load request. This slice builds the task control block.

2D/45 Loader-slice-4
Fourth phase of processing a program load request.
Provides:

1. Determination if a program control block exists.
2. Building of a program control block if one does not exist.
3. Link task control block to program control block.

2E/46 Loader-slice-5
Fifth phase of processing a program load request.
Provides:

1. Error recovery for load failures.
2. Updating of the task detail table in the case of a zip.
3. Determination if a debug interpreter is needed.

2F/47 DC-space
This slice is set to the size of the data comm area in memory, calculated when a NDLSYS program is loaded.
It contains the data comm buffers, NDL S-code, and NDL tables.
There is no micro code in this slice.

30/48 DS-slice
Processes DS, DP, and PR SCL commands.

31/49 RTC-date-fix
Updates date at end-of-day (2400 hours).

32/50 Stop-slice
Processes requests to stop an executing program.

33/51 Go-slice
Processes requests to restart a program that is suspended by:

1. ST command.
2. Pause communicate.
3. No user disk condition.
4. Special forms request.
5. No file condition.
6. Duplicate file condition.

34/52 Power-off
Processes requests for an orderly shut down of an entire disk.

Hex/Dec

35/53 OL-slice

Processes requests by:

1. OL SCL command.
2. Configuration table maintenance to print out status of a peripheral it has just recognized.
3. Help task to print out status of a peripheral that has gone Not Ready while in use.

36/54 AD-slice

Processes assignment of a peripheral to a program by an operator in the following cases:

1. Unlabeled input file.
2. Special forms.
3. No file condition.
4. Duplicate file condition.
5. Printer unavailable.
6. No user disk.

37/55 DMAC-disk-slice

Disk controller for fixed, cartridge, and super mini units.

38/56 SPO-slice

Controller for TC4201 SPO.

39/57 DC-CH-30

Processes data comm communicates with a verb value of 30. This slice is entered from the master communicate handler. The adverb values are:

Communicate	Adverb
queue	00
queue.depth	01
set.input.limit	02
set.queue.limit	03
exchange.reference	04
fetch.message	05
get.message.space	06
release.message.space	07
read.header	08
write.header	09
read.text	0A
write.text	0B
copy.text	0C
continue.station	0D
continue.task	0E
route.input	0F
route.output	10
allow.input	11
disallow.input	12
allow.output	13
disallow.output	14
set.output.limit	15

Hex/Dec

3A/58 DC-ch-31

Processes data comm communicates with a verb value of 31. This slice is entered from the master communicate handler. Their adverb values are:

Communicate	Adverb
line.count	00
station.count	01
modem.count	02
terminal.count	03
subnet.count	04
line.number	05
station.number	06
queue.nuumber	07
line.description	08
station.description	09
modem.description	0A
terminal.description	0B
subnet.description	0C
line.stations	0D
subnet.stations	0E
line.status	0F
station.status	10
task.name	11
task.number	12
recall	13
clear	14

3B/59 DC-CH-32

Processes data comm communicates with a verb value of 32. This slice is entered from the master communicate handler. The adverb values are:

Communicate	Adverb
redefine.line	00
redefine.station	01

3C/60 DC-ch-33

Processes data comm communicates with a verb value of 33. This slice is entered from the master communicate handler. The adverb values are:

Communicate	Adverb
enable.input	00
disable.input	01
enable.output	02
disable.output	03
receive	04
send	05
accept	06

Hex/Dec

3D/61 DC-nucleus

This is a collection of data comm routines invoked from the data comm communicate slices. This slice is assembled at an absolute address. Part of the DC-nucleus is the PMLC (programmable multi line control), which is a controller for the DCP. The job of the PMLC is to service all interrupts from any DCP. The four types of interrupts are:

1. 7PM or 8PM parity error.
2. SPM parity error.
3. Result queue needs service.
4. DC system needs memory space.

3E/62 DC-task-EOJ-slice

This:

1. Detachs the current task from all subnet queues.
2. Detachs the current task from all station queues.
3. Sends a detach message to the MCS indicating the task being detached.

3F/63 DC-loader

First phase of loading a data comm subsystem.

Provides:

1. Get space for the DC queue handler TCB.
2. Get space for the DC space handler TCB.
3. Opening of the NDLSYS file.
4. Size calculation of DC-space-slice and allocation of memory for it.

40/64 Help-task-slice

This slice is an independent runner that interfaces with:

1. All I/O controllers because of a not ready condition while in use.
2. OL-slice.
3. MH-slice.
4. Virtual memory.
5. Reconstruct.

Provides:

1. Printing of Not Ready message for all devices when required.
2. Configuration table maintenance.

41/65 Working-set-bailiff

The working set bailiff is composed of two modules. One is resident in the MCP-nucleus and the other is a micro slice and an independent runner.

The functions of the independent runner module are:

1. Evict task run structure from memory to backing store for one of the following reasons:
   1) Operator stops the task.
   2) Virtual memory routine senses there is insufficient memory available to fullfill a request for memory allocation.

3) Thrashing is detected.
4) A program is loaded.

2. Restore task's runstructure from backing store to memory for the following reasons:
   1) Operator "GOs" a stopped task.
   2) SCL has information for the task (AD, AX).
   3) The task can be run and the current task has released required system resources.

One task is evicted at a time until memory requirements are satisfied. The priority with which tasks are evicted is to evict the lowest priority task that is:

1. Stopped.
2. Long-waited.
3. Runnable.

One task is restored at a time until one does not fit. The priority with which tasks are restored is:

1. Task receiving SCL information.
2. Highest priority runnable task.

42/66 Clear-start

Provides:

1. Access to system disk sector 31 which contains the restart information.
2. Display of the Clear/Start message.
3. Loading of WS-1 to memory and then passing control to it.
4. Called by power-off slice to display the PO message and re-enter warmstart if the system disk was powered off.

43/67 Configurator

Processes requests by some utilities for the names of all disks on-line.

44/68 OS-tracer

Invoked by a "GT" SCL command which gives a trace of all communicates on a line printer. The trace listing has 10 columns of D-words. The first column is the current task's mix number, the second column is the absolute address of the current task's communicate parameter area, and the last eight columns contain the communicate.

45/69 DC-EOJ-slice

This:

1. Puts PMLC in load/clear mode.
2. Zeroes PMLC controller address in PHT.
3. Zeroes DC registers.
4. Removes DC-queue handler's TCB from memory.
5. Removes DC-space handler's TCB from memory.
6. Removes DC-space from memory.

7. Removes DC-nucleus from memory.
8. Removes all DC micro slices from memory.

46/74 Reconstruct
Called by help task when an unacceptable integrity flag is encountered. It:
1. Locates pointers to file list, DFH, available table.
2. Steps through DFH:
   1) If codefile found, resets user count and deallocates second area if applicable.
   2) If data file found, resets user count. This includes VM and DM files except for DMFILOO.
   3) If MCP file found, resets user count if not systems disk.
   4) If reserved entry found, deallocates all areas indicated.
3. Rewrites label to indicate good integrity.

47/71 RY-slice
Processed RY and SV SCL commands.

48/72 XD-slice
Processed XD SCL commands to remove bad sectors from disk.

49/73 DC-SH-slice
This slice gets space from the data comm reserve buffer pool and links this space to the available buffer pool.

4A/74 DC-EH-slice
This slice handles errors reported by the PMLC controller and prints them on the SPO.
Example: SPM PARITY ERROR.

4B/75 DC-LDR-2
This:
1. Loads NDLSYS file.
2. Calls DC-pre31-load if necessary.

4C/75 DC-LDR-3
This:
1. Half-closes NDLSYS file.
2. Updates configuration table.

4D/76 DC-LDR-4
This:
1. Opens NDLDCP file.
2. Loads NDLDCP file.
3. Half closes NDLDCP file.

Hex/Dec

4E/77 AD-disk
Process request to assign a peripheral to a program by an operator in the following cases:

1. Unlabeled input file.
2. Special forms.
3. No file condition.
4. Duplicate file.
5. Printer unavailable.
6. No user disk.

4F/79 Indexed-slice
Processes communicates for indexed files.

50/80 PE-table
This slice has read-write tables necessary for tape DMAC. They are loaded into tape DMAC DDP at warmstart time. The tables are:

1. Binary to binary.
2. EBCDIC to ASCII.
3. ASCII to EBCDIC.

51/81 DC-no-DCOM-SLC
This slice is called as a result of a task data comm communicate being received by the MCP when the data comm subsystem is not present. If this condition exists, a 91 is placed in the CD status key and control is passed back to the task.

52/82 Disk-open-2
Provides:

1. Location of FPB and configuration table.
2. Start of construction of FIB.
3. Search of disk directory for the file name in the case of an old file or search of the available table for an available disk area in the case of a new file.

53/83 Disk-open-3
This:

1. Builds disk file header.
2. Calculates number of areas and their sizes.
3. Completes construction of the FIB.

54/84 Disk-close-2
Provides:

1. Updated DFH.
2. Deallocation of disk space.

Hex/Dec

55/85 Tape-open-2
This:
1. Loads tape controller in memory
2. Searchs tape(s) for file.
3. Starts construction of FIB.

56/86 Tape-open-3
Provides:
1. Completion of FIB construction.
2. Processing of label handling.

57/87 Tape-close-2
This:
1. Rewinds tape.
2. Purges label.
3. Marks tape saved if closed with lock.
4. Desolves FIB.

58/88 Tape-close-3
This slice is called by EOJ or tape-close-1 to close a half closed file.

59/89 EOJ-slice-2
This slice deallocates the virtual memory file or changes it to a dump file if the task was DP'd.

5A/90 EOJ-slice-3
This slice dissolves the TCB and PCB, if the user count is zero, and closes the codefile.

5B/91 Dealloc-slice
This slice returns unused disk space to the available table.

5C/92 OS-susp-slice
Processes task suspensions by the operating system.

5D/93 Stream-slice
Processes stream access communicates.

5E/94 Index-open-1
This slice is the first phase of indexed file open communicate processing.
This:
1. Checks FPB syntax.
2. Reads in KFPB.
3. Fills-in fields in extended FPB.

5F/95 Index-open-2
This slice builds the DFH for a new key-file.

60/96 Index-open-3
This slice builds the FIB.

Hex/Dec

61/97 Index-open-4
This slice reconstructs the rough table if an old file is being opened.

62/98 Index-close-1
This slice is called from disk-close-1 if the file to be closed is a indexed file and one the following conditions exists:

1. The file is being locked.
   1) New file - build KFPB.
   2) Old file - merge if opened I/O or extend and merge bit is set.
2. The close adverb is remove.
   1) New file - build KFPB.
   2) Old file - merge if opened I/O or extend and merge bit is set.
   3) Old and new file - the old file should already be purged, the new file is now locked.
3. The close adverb is release and the file is old.
   1) No name is inserted in KFPB.
   2) Merge if opened I/O or extend and merge bit is set.
4. The file is being half-closed.
   1) New file - build KFPB.
   2) Old file - establish link to data file.
   3) No merge is allowed.

Close with purge or release and file is new does not invoke this slice.

63/99 SPO-console
Controller for 120 cps console/SPO.

64/100 Interp-subs
This slice is the interface for the interpreters to relinquish control of the machine to the operating system and through which control is returned back to them. It is loaded to micro memory 1 (logical page) at warmstart time and is not overlayable. The slice descriptor shows an absolute memory address in MM1 but the slice length is invalid.

65/101 Console-slice
Controller for 120 cps console files.

66/102 Index-open-5
This slice completes the open if an old file is being accessed output sequential.

67/103 Sort-slice-1
This slice is called by SORTINTRINS, an MPLII program, with a micro library call. Tags are built from user-specified

keys and are written to a 5760-byte work area. When this work area is full, these tags are sorted into ascending or descending order and then written to a work file named BS.FILE. These blocks of sorted data are called strings. This routine loops until the input file is at EOF and then returns back to SORTINTRINS.

68/104 Sort-slice-2
This slice is called by SORTINTRINS, an MPLII program, with a micro library call. The strings in work file BS.FILE, created by sort slice 1, are continually merged by way of a four-way table merge until there is only one string left. From this tag file a full record data file, tag file, or key file is generated depending on the type of sort requested. If a merge of two or more files is requested, the entire merge is done by SORTINTRINS.

69/105 Sort-slice-3
This slice is for future use. No code exists for it in the MCP file.

6A/106 DC-ch-34
Processes data comm communicates with a verb value of 34. This slice is entered from the master communicate handler. The adverb values are:

Communicate	Adverb
DCP.reload	00
DCP.program.names	01
DCP.program.count	02
DCP.description	03
DCP.program.terminals	04

6B/107 DC-CH-redef2
This slice checks for consistency of parameters after a redefine.

6C/109 DDE-SPO
Controller for the DDE SPO.

6D/109 DDE-con
Controller for DDE console files.

6E/110 Log-task-slice
This slice maintains the log files for both SPO and maintenance entries.

6F/111 SSC-slice
Processed requests by privileged users for a system status (MPL construct).

Hex/Dec

70/112 Disk-pack-ctlr
Controller for disk pack.

71/113 Large-scrn-SPO
Controller for the TD 830 SPO.

72/114 FD-slice
Processes SCL command to define the format of output to a serial printer.

73/115 Int-mul-div
This slice processes decimal arithmetic for the interpreters.

74/116 DC-pre31-load
This slice loads a NDLSYS file that was compiled prior to 3.1 NDL compiler release.

SECTION 12
B 800 CMSMCP FILE

The B 800 CMSMCP file is a single area file which has all the microcode necessary to operate the system, with the exception of the NDL data comm processor micro code which is loaded into the data comm processor if an MCS is executed. The NDL data comm micro code is handled as a separate system file.

Internal structure of the B 800 CMSMCP file is broken down into three portions. The first portion is a one-sector available table which is used in the plant to mark areas of the MCP file that they have removed for testing and analysis purposes. This area should never contain any entries on a field level MCP file. The next portion of the file is a 24-sector slice descriptor table describing the various MCP slices located in the third portion of the file. Each sector of this slice descriptor table contains 11 entries of 16 bytes each. The format of the B 800 MCP slice descriptor is given in table 12-1. This table is used by the warmstart routine to build the slice descriptor table (SDT) in memory and by patch in altering the disk slices. Note that there are two slice descriptor tables; one in the MCP file and one in memory. It is important not to confuse them.

Table 12-1. Field Contents

Byte Offset Within Entry		Data Code	Notes
0 - 3	OS slice-ID	as	(1)
4	micro slice number	bi	(2)
5	slice descriptor offset in sector	bi	(3)
6 - 7	sector no. of table for this slice entry	bi	(4)
8 - 9	offset in MCP file of the start of the slice	bi	(5)
10 - 11	length of the slice in D-words	bi	
12 - 13	location to load in memory	bi	(6)
14 - 15	date slice assembled (mm/dd)	de	(7)

as=ASCII bi=BINARY de=DECIMAL

related notes:

1. The OS slice ID is the first four characters of the slice name given by the assembler programmers when the slice is declared to the assembler. For example, the first slice (slice 0) ID is OS-S. This is the first four characters of the slice name OS-SUBS, the name given to the system registers and basic subroutines slice. Similarly, the second slice (slice 1) is identified as WS-1 which is the first four characters of the slice name WS-11 used as a section of the warmstart function.

Table 12-1. Field Contents (Cont.)

2. The micro slice number is the relative slice number of this descriptor from the start of the slice descriptor table in the MCP file (starting at zero).

3. The slice descriptor offset in sector is the base zero relative location of this slice descriptor in this sector.

4. This is the sector number within the MCP file slice descriptor table in which this slice entry occurs.

5. Offset within the MCP file of the start of the data contained in this micro slice, in sectors, starting at base zero.

6. With the one exception of OS-SUBS, this field, if not zero, indicates the absolute memory address in D-words at which this micro slice is loaded. If zero, the slice can be loaded in the slice area of memory (between base of the base of virtual memory and slice extent) at loader discretion. This field is zero for OS-SUBS because this slice starts at absolute memory address zero.

7. Date slice assembled indicates the month and the day of the month when this slice was last assembled for this version of the MCP.

The third portion of the MCP file consists of the disk copies of all micro slices used in the system. It includes the interpreter for BIL/MPLII, COBOL, pocket select language (PSL), and data comm interfacing to the NDL system, as all the MCP slices needed to manage the system. The slices are formatted in three ways:

packed - 90 micros per sector
unpacked - 45 micros per sector
data slices - tables or ASCII data

A packed micro slice is one which is either part of the operating system nucleus (OS-SUBS, OS-NUCLEUS, DC-NUCLEUS) or is a free-standing piece of micro code which does not run under the operating system (warmstart, clear/start, and patch slices, for example). These slices, since they exist as separate entities or as part of the fixed nucleus, do not need an internal addresses "fixed up" when they are loaded into the system and are therefore packed 90 micros to a sector.

Unpacked slices, however, are those which are loaded into main memory at different addresses depending on the system operating state at the time the slice is needed. Since these slices must "call" and "jump" both within themselves and to the routines within the fixed address area of the operating system nucleus, each executable micro is followed by a flag word which indicates to the loader (the disk DMAC control) whether the preceding micro instruction needs its "address part" fixed to point to a address relative to the base of where this slice is located in memory or whether it should be left undisturbed. It is left undisturbed when it is either a non-address type micro or it refers to an absolute address within the OS nucleus. A micro requiring "fix up" is suffixed by a D-word containing @8000@. Otherwise, it is followed by a word containing @0000@.

A data slice is one that contains only ASCII data in the form of a look-up table or a table of operating system messages. Since these tables are accessed only in a look-up manner and do not contain direct executable micro code, they are packed with 180 bytes of data or table per sector.

APPENDIX A

BMAR	memory address register
BVM	beginning of virtual memory
CPA	communicate parameter area
CPS	characters per second
D-word	two bytes
DC	data communications
DDP	device dependent port
DST	data segment table
EVM	end of virtual memory
FC	fetch communicate
FIB	file information block
FPB	file parameter block
I/O	input/output
ISCT	interrupt scan control table
ISN	interpreter segment number
MA	memory address
MAD	micro address descriptor
MAT	mix attribute table
MB	megabyte
MC	media controller
MCH	master communicate handler
MCP	master control program
MM	memory management
MPCR	micro program count register
OS	operating system
PCA	program current address
PCB	program control block
PHT	peripheral handler table
PSN	program segment number
SDT	slice descriptor table
TCB	task control block
TDT	task detail table
TID	task ID
TOS	top of stack
TPI	tracks per inch
VM	virtual memory

APPENDIX B

MCP LOGIC ERRORS WORKSHEET

This should be used as a checkoff sheet when documenting an FTR.

BMAR @0083@ = _____

MPCR @0084@ = _____

ERROR @0085@ = _____

CURRENT SLICE @000C@ = _____

CURRENT TASK TID @000A@ = _____

HARDWARE CONFIGURATION: _____

_____

_____

_____

_____

_____

_____

_____

_____

Review:

- SPO log
- mix worksheet
- slice memory worksheet
- peripheral I/O descriptor worksheet
- MCP trail

MIX WORKSHEET

PART 1

ma = memory address

MAT ANALYSIS
OS register @000E@ = _____. Proceed to this memory address and extract the MAT.

C FFFF ____ ____ ____ ____ ____ ____

____ ____ ____ ____ ____ ____ ____ ____ ____

B FFFF ____ ____ ____ ____ ____ ____ ____ ____ ____

A FFFF ____ ____ ____ ____ ____ ____ ____ ____ ____ FFFF

TDT ANALYSIS
OS register @0017@ = _____ locates the SDT-base ma

Add @3C@ giving _____ locates the SCL-loader SDT entry

add @03@ giving _____ locates the TCB ma
Proceed to this memory address.

Contents = _____ locates the start of the TCB

add @01@ giving _____ locates the DST in the SCL TCB
Proceed to this memory address.

Contents = _____ ma of data segment 0 of SCL-LDR

add @04@ giving _____ ma of data segment 1 of SCL-LDR

add @03@ giving _____ locates the start of TDT
Proceed to this memory address.

Contents = _____ locates the start of TDT

Proceed to this ma and mark off sets of 23 D-words on the hex dump D-words in the table.

- ____-____ ____ ____ ____-____ ____ ____ ____ ____ ____
1 5 (ASCII) 8 9 (ASCII) 14

- ____-____ ____ ____ ____-____ ____ ____ ____ ____ ____

- ____-____ ____ ____ ____-____ ____ ____ ____ ____ ____

- ____-____ ____ ____ ____-____ ____ ____ ____ ____ ____

- ____-____ ____ ____ ____-____ ____ ____ ____ ____ ____

- ____-____ ____ ____ ____-____ ____ ____ ____ ____ ____

- ____-____ ____ ____ ____-____ ____ ____ ____ ____ ____

- ____-____ ____ ____ ____-____ ____ ____ ____ ____ ____

- ____-____ ____ ____ ____-____ ____ ____ ____ ____ ____

MIX WORKSHEET

PART 2

Summarize the mix contents in the tables below.

Independent runners:

TASK #	PRIORITY	TASK ID	STATUS
F	C	data comm	
E	C	working set bailiff	
D	C	help / CTM	
C	C	SCL loader	
B	C	data comm space handle	
A	C	log	
9	B	super utility	

Identify the status for each of the independent runners above.

User programs:

TASK #	PRIORITY	TASK ID	STATUS
8			
7			
6			
5			
4			
3			
2			
1			
0			

Identify the priority, task name, and status for each task #.

CURRENT TASK WORKSHEET

OS register @000A@ = _____ task number
add @01@ giving _____ mix number
program name _____ from mix worksheet
OS register @00B6@ = _____ line-count in hex
OS register @00AD@ = _____ current s-opcode
OS register @000B@ = _____ current TCB ma
Proceed to this location and extract 7 D-words

_____	_____	_____	_____	_____	_____	_____ (TCB)
1	2	3	4	5	6	7
PSN/ISN	DST BASE	DST LIMIT			MCH-TOS RECOVER	MCH ACTIVE VERB

To find the CPA, add MCH active verb (D-word 7) _____
to the TCB base (found at @000B@) _____
giving the CPA address _____
The contents of the D-word at this address is _____
This is the communicate verb (refer to MCP manual section 5)

To extract the COBOL/RPG perform stack from a hex dump,

Add MCH-TOS recover (D-word 6 of TCB) _____

to the TCB base (found in part 1 at @000B@) _____

giving _____

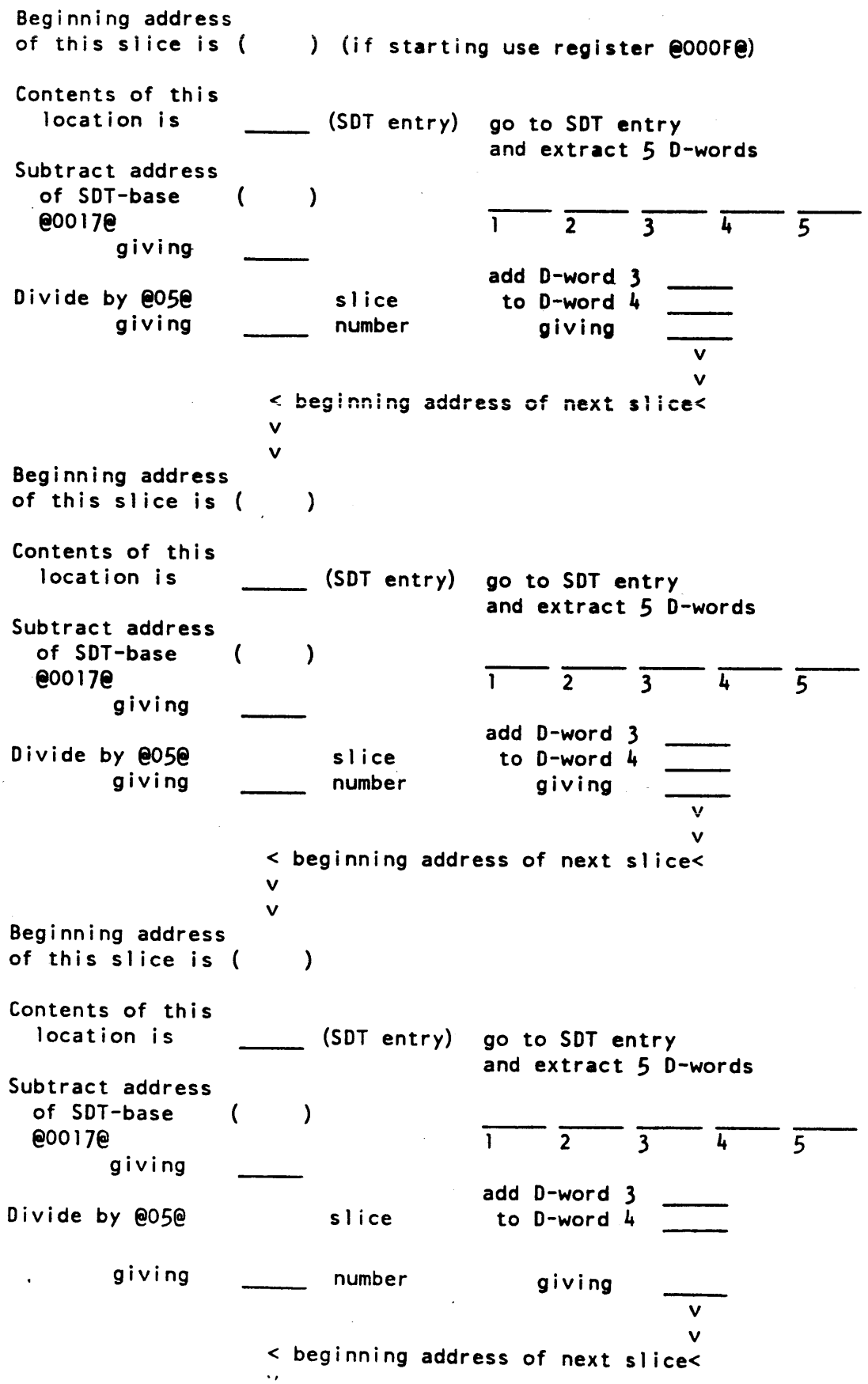
subtract @01@ giving _____

Go to this resulting address in the dump and mark off one D-word. It represents the boundary between the interpreter's part of the stack and the MCP's part.

Proceed to contents of D-word 3 and mark off sets of 4 D-words

LINE-COUNT	BYTE OFFSET	SEGMENT NUMBER	"K"	LINE-COUNT	BYTE OFFSET	SEGMENT NUMBER	"K"
_____	_____	_____	_____	_____	_____	_____	_____
1	2	3	4	1	2	3	4
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____

SLICE MEMORY MAPPING WORKSHEET



PERIPHERAL I/O DESCRIPTOR WORKSHEET

OS register @0043@ = _____ ma of ISCT.
Proceed to this ma and extract 12 D-words.

1	2	3	4	5	6	ports
7	8	9	10	11	12	ports

Selected device _____

Select the ma corresponding to the port on which this device is located.

Add @02@ giving _____ ma of the current I/O

contents = _____ ma of I/O descriptor
Proceed to this ma and extract 4 D-words.

1	2	3	4	
1	2	3	4	(queued I/O descriptor)
1	2	3	4	(queued I/O descriptor)
1	2	3	4	(queued I/O descriptor)

D-word 1 contains the interlock and op code.
D-word 2 contains the next I/O descriptor ma or an ma pointing back to the PHT.

Selected device _____

Select the ma corresponding to the port on which this device is located.

Add @02@ giving _____ ma of the current I/O

contents = _____ ma of I/O descriptor
Proceed to this ma and extract 4 D-words.

1	2	3	4	
1	2	3	4	(queued I/O descriptor)
1	2	3	4	(queued I/O descriptor)
1	2	3	4	(queued I/O descriptor)

D-word 1 contains the interlock and op code.
D-word 2 contains the next I/O descriptor ma or an ma pointing back to the PHT.

Documentation Evaluation Form

Title: B 800 Systems MCP Memory Dump Analysis
User's Guide

Form No: 1118452
Date: January, 1981

Burroughs Corporation is interested in receiving your comments and suggestions regarding this manual. Comments will be utilized in ensuing revisions to improve this manual.

Please check type of Suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments:

From:

Name _____

Title _____

Company _____

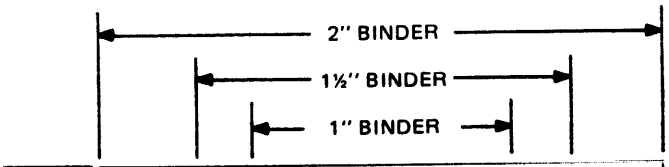
Address _____

_____

Phone Number _____ Date _____

Remove form and mail to:

Documentation Dept, TIO - East
Burroughs Corporation
Box CB7
Malvern, PA 19355



B 800 Systems
MCP Memory Dump Analysis
USER'S GUIDE

1118452

Printed in U.S.A.

Printed in U.S.A. February 1981 1118452