Burroughs

B 900/CP 9500 Systems

MEMORY DUMP

USER'S GUIDE

(Relative to 3.04.23 Release)

PRICED ITEM

Printed in U.S. America

November 1982

1118478

The names, places, and/or events depicted herein are not intended to correspond to any individual, group or association existing, living, or otherwise. Any similarity or likeness of the names, places, and/or events with the names of any individual, living or otherwise, or that of any group or association is purely coincidental and unintentional.

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

This edition also includes the information released under the following:

PCN 1118478-001 (November 26, 1981)

Correspondence regarding this publication should be forwarded using the Remarks form at the back of the manual, or may be addressed directly to TIO East Documentation, Burroughs Corporation, P.O. Box CB7, Malvern, Pennsylvania, 19355, U.S. America.

LIST OF EFFECTIVE PAGES

Page	Issue
iii thru xiv	Original
1-1, 1-2	Original
2-1 thru 2-5	Original
2-6	Blank
3-1 thru 3-6	Original
4-1 thru 4-3	Original
4-4	Blank
5-1 thru 5-12	Original
6-1 thru 6-6	Original
7-1 thru 7-13	Original
7-14	Blank
8-1 thru 8-23	Original
8-24	Blank
9-1 thru 9-26	Original
10-1 thru 10-14	Original
11-1 thru 11-22	Original
12-1 thru 12-10	Original
A-1 thru A-8	Original
B-1 thru B-5	Original
B-6	Blank
C-1 thru C-3	Original
C-4	Blank
D-1, D-2	Original
E-1 thru E-9	Original
E-10	Blank
F-1 thru F-5	Original
F-6	Blank

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

TABLE OF CONTENTS (CONT.)

LIST OF ILLUSTRATIONS

LIST OF TABLES

INTRODUCTION

This manual is a guide which will assist Burroughs personnel in identifying and resolving certain problems with the B 900/CP 9500 system. It gives both a system overview and a detailed discussion of the contents of a system memory dump, probably the single most effective tool available for the task of analyzing problems.

This manual is divided into 12 sections and 6 appendices, A through E.

Section 1 provides an overview of the methods by which a system dump file can be created.

Section 2 explains the operation of the SYSANALYZER program which is used to convert the numbers and symbols from a hexadecimal memory dump into a format that is more easily read.

Sections 3, 4, and 5 give an overview of system operation. Section 3 is a software overview, section 4 discusses interprocessor communication, and section 5 discusses the theory of operation for the system software.

Sections 6 through 12 discuss the actual memory dump as it appears after it has been converted using SYSANALYZER.

Section 6 is titled Heading Information. This section discusses the general information contained in the first four pages of a memory dump.

The next six sections of the manual discuss the major divisions of the memory dump. They are discussed in this manual in the order in which they appear in the dump. They are:

Section 7 - Virtual Memory

Section 8 - Operating System

Section 9 - Data Communications

Section 10 - Task Processor

Section 11 - Disk Management

Section 12 - Buffer Memory

The appendices supply detailed information which is frequently needed as reference.

Appendix A lists operating system error messages.

Appendix B gives task processor error messages.

Appendix C is a list of, and an explanation of, SYSANALYZER error messages.

Appendix D gives techniques for user task analysis.

Appendix E is a guide for trouble-shooting system failures. This is to assist a field engineer in making initial efforts to correct problems.

Appendix F is glossary of terms.

It is assumed that the support personnel using this manual are familiar with CMS concepts and the implementation of these concepts on the B900/CP 9500 system. It should be understood that this document is not a substitute for a properly trained software support person.

Related Documents

The following manuals may be of assistance in understanding system operations:

CMS Software Operation Guide, form 2015228-001

CMS Data Communications Reference Manual, form 1090909

CMS Master Control Program (MCP) Reference Manual, form 2007555-001

SECTION 1
MEMORY DUMPS

INTRODUCTION

A system memory dump is a file created on disk containing the contents of the system memory. The system memory dump, in conjunction with system log files, are the most essential tools to aid in the analysis of system failures. It is very important that a memory dump be taken each time an unexplained system failure occurs. It is equally important that the dump be taken properly and contain useful information. This section details when a memory dump file creation occurs. Printing of the dump file is performed by the SYSANALYZER program, and is covered in Section 2 of this manual.

CREATION OF DUMP FILES

System dumps are produced in three distinct ways. They may be created automatically by the operating system when the system detects a failure (such as Task Processor/Data Comm Processor Logically Not Ready, or a CLEARSTART takes place); the memory dump may be requested by the operator using the GT MD intrinsic, explained in Section 9 of the CMS Software Operations Guide (SOG), form 2015228; and thirdly, the memory dump may be manually created by using the ROM/PRAM dump facility. This option is to be used when a system hang occurs. (System hang is defined as a condition where the system is unable to communicate with the operator.) It is also used when the system fails to automatically create a dump. Operation of the ROM/PRAM dump facility is explained in Section 9 of the CMS SOG.

DUMPS CREATED BY OPERATING SYSTEM

Dumps produced by the operating system are written into a previously-assigned disk file named SYSDMFILEnn. This file is assigned space by the monitor startup procedure of the OS. The decimal number nn is determined by startup in the following steps.

Startup attempts to open, initialize, and close the file SYSDMFILE00; if a duplicate file condition occurs on the file close attempt, the file just created is purged and an attempt is made to create SYSDMFILE01. This procedure is repeated until a successful close occurs or no file number is found to be available. The maximum number of system dump files allowed on the system disk is six (SYSDMFILE00 through SYSDMFILE05). This procedure ensures that previous valid dump files are not destroyed by subsequent warmstarts. If startup is unable to find sufficient disk space, an appropriate warning message, "SYSDMFILE NOT CREATED MAX NO EXCEEDED," is displayed on the Operator Display Terminal, and the system continues the warmstart. The normal operator action is to first reorganize the system disk so that sufficient space is available, and then re-warmstart the system to create a valid dump file. The OS monitor dump action can only write into a dump file created at the last warmstart by the OS monitor startup action.

NOTE

An unused system dump file is removed automatically as part of the system disk power-off procedures. If the system is not powered off in an orderly manner, or if the system hangs, an empty dump file will remain on disk.

DUMPS CREATED BY GT MD

The GT MD intrinsic will dump the system memory to the current dump file while the system continues normal operation. This can be useful in gauging system performance, and for certain types of failures that do not cause fatal error conditions, such as with programs that do not appear to be executing.

ROM DUMP FILES

A system can be dumped to disk by use of the hardware ROM/PRAM dump facility. This ROM dump must be reformatted by a utility, 'ROMCONVERT,' to be identical in layout to a dump file created by the operating system. In this form it can be analyzed by SYSANALYZER. Operating instructions for ROMCONVERT are contained in the CMS SOG.

SECTION 2
SYSANALYZER PROGRAM

INTRODUCTION

SYSANALYZER is a program which selects and prints data from a system dump file. It also performs some basic dump analysis by highlighting suspected system errors. This section describes how to use the SYSANALYZER program, and includes the format of the printed output.

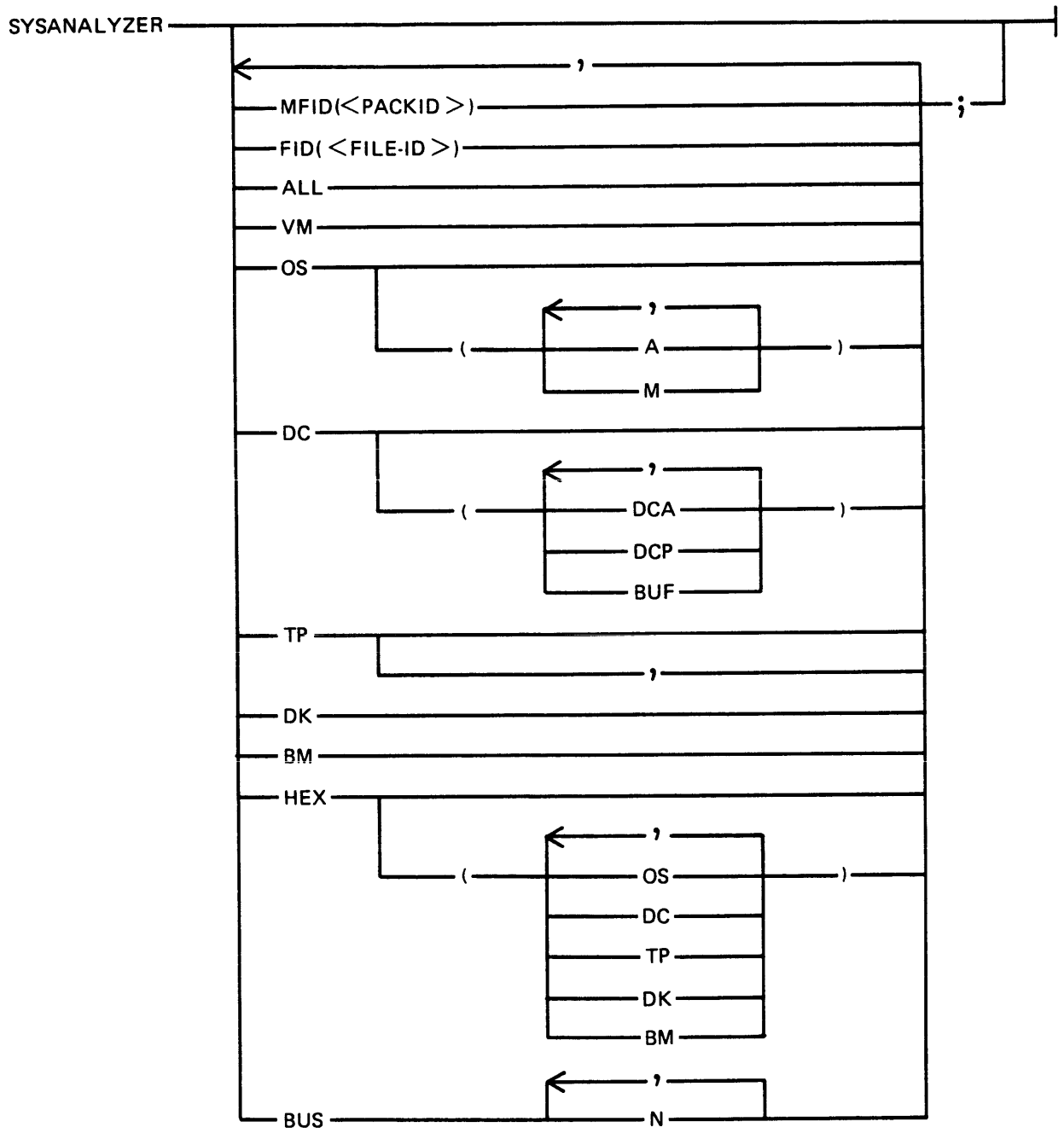
DUMP FILE REQUIREMENTS

The SYSANALYZER program requires as input either a system-created dump file or a ROM dump file which has been converted into the format required by SYSANALYZER. This conversion process is performed by the ROMCONVERT utility.

The 3.04 version of SYSANALYZER also requires that the dump file being analyzed must have been created on a system executing under 3.04 MCP control.

SYSANALYZER SYNTAX

Parameter values are supplied to SYSANALYZER through a single BOJ initiating message. The syntax of this message is depicted in the railroad diagram below. The options may be entered in any order; the analysis takes place in the option sequence depicted in the diagram.



ED2644

CAUTION

The options ALL and HEX are advised for analysis requested by the user. ALL produces a formatted full-system analysis and a hexadecimal dump; HEX produces a hexadecimal dump. The user is advised to ignore the more specific analysis options, except when otherwise requested.

The following points should be noted:

1. If the user enters the "FID( )" option and/or the "MFID( )" with no other options following, only the header page and dump file information are output by the analyzer program.
2. If the initiating message contains no options (consists only of the string SYSANALYZER), the system analyzer uses default values equivalent to "MFID(<system pack>), FID (SYSDMFIL00), ALL;" If all options are omitted, the semicolon is not included in the message.
3. If the DK option is selected, a Hex dump of the disk processor will automatically follow the formatted version.

Option	Meaning
MFID( )	The <packid> specified is the pack on which dump file resides.
FID( )	The <file-id> specified is that of the dump file to be analyzed.
ALL	Perform analysis of: Virtual Memory (see VM) Operating System (see OS) Data Comm Processor (see DC) Task Processor (see TP) Disk Processor (see DK) Buffer Memory (see BM)
VM	Perform analysis of virtual memory.
OS	Perform analysis of operating system. If no suboptions are specified, use all suboptions. Suboptions of OS: (A) Activity management. (M) Memory associated with OS processor.
DC	Perform an analysis of each data comm processor. If no suboptions are specified, use all suboptions. To restrict analysis to individual DCPs, use the "BUS" option. Suboptions of DC: (DCA) Include analysis of data-comm-related parts of operating system. (DCP) Include analysis of data-comm-related parts of disk processor. (BUF) Include analysis of data-comm-related parts of buffer memory.
TP	Perform an analysis of each task processor. To restrict analysis to individual task processors, use the "BUS" option.
DK	Perform analysis of disk processor.
BM	Perform analysis of buffer memory
VM	Perform analysis of virtual memory.
HEX	Print an unformatted hexadecimal dump of the memory or memories specified. If no suboptions are specified, print a full hex dump of the system. Suboptions for HEX: (OS) Print a hex dump of memory associated with the operating system. (DC) Print a hex dump of memory associated with the data comm processor. (TP) Print a hex dump of memory associated with the task processor(s). (DK) Print a hex dump of memory associated with the disk processor. (BM) Print a hex dump of buffer memory.
BUS(n)	Each "n" is a hexadecimal character that corresponds to a given physical bus address. When the BUS option is specified, only the specified bus addresses are considered to be in the dump file.

The BUS(n) option allows analysis of the specific task processor(s) or data comm processor(s) that the user wishes to examine. When this option is not used, all task processors are analyzed as a result of TP, and all data comm processors are analyzed as a result of DC.

The OS processor memory must be available if the DC(DCA) or VM option is specified. To do a complete analysis of a DCP when the BUS (n) options is used, the bus address of the OS processor must be specified, as well as the bus address of the given DCP.

The BUS (n) option may be used together with the ALL option. This causes a comprehensive analysis, but only of the bus addresses specified.

SYSANALYZER EXAMPLE

The following initiating messages result in the same analysis. The actual sequence of analysis is represented by the order of the options in the second message.

BOJ Message 1: SYSANALYZER FID(THISFIL), MFID(THATPCK), ALL;

BOJ Message 2: SYSANALYZER MFID(THATPCK), FID(THISFIL), ALL;

SYSANALYZER CONVENTIONS

The reader should be aware of the following conventions which apply to the printed output:

FORMATS OF TITLES

Titles boxed in asterisks are titles of analysis options, suboptions, and headings of individual data structures.

ERROR MESSAGE FORMAT

Error messages are printed on the extreme left hand portion of the output listing, and are preceded and trailed by three dollar signs.

BYTE REVERSAL FORMAT

Data declared to be two-byte is stored in memory byte-reversed by the BDS processor. In order to provide a consistent approach to output, the following rules concerning byte reversal have been followed:

1. All data read from the dump file is printed in byte-reversed form (as it was found in the dump file).

2. Addresses that are calculated by the analyzer and printed in front of a dump area (in the same columns as data titles) are printed in a byte-forward manner.

PRINT FORMAT

The formatted analysis uses a global print routine to produce the printed output. The 120-character-wide print line is logically divided into a 40-character and an 80-character area. The 40-character area is used to display titles (and addresses) right-justified preceding the appropriate data. The 80-character area is used to print the data. The data is always printed as follows:

<two bytes> blank <two bytes> blank ... <two bytes>.

If the number of bytes is odd, the last <two bytes> are printed as <<single byte> blank>. The routine will continue on to succeeding lines, as necessary, if more than 32 bytes are to be printed. This routine is not used when data is printed in a table format.

NOTE

Disk addresses are NOT byte-reversed.

ROMANALYZER

A ROM-generated program may be analyzed directly by the program ROMANALYZER at those times when it is not possible to run ROMCONVERT. ROMANALYZER is explained in detail in the System Operation Guide.

SECTION 3
SOFTWARE OVERVIEW

The system firmware comprises all necessary modules to provide full CMS support. This includes:

1. Operating system (OS).
2. Data storage subsystem.
3. Task processor control routines and language interpreter.
4. System-dependent initialization routines.
5. Data communication control, messsage handling and line protocols.

Figure 3-1 illustrates the interrelationship of the above modules. Initialization routines are not included in the figure, as they are a stand-alone function.

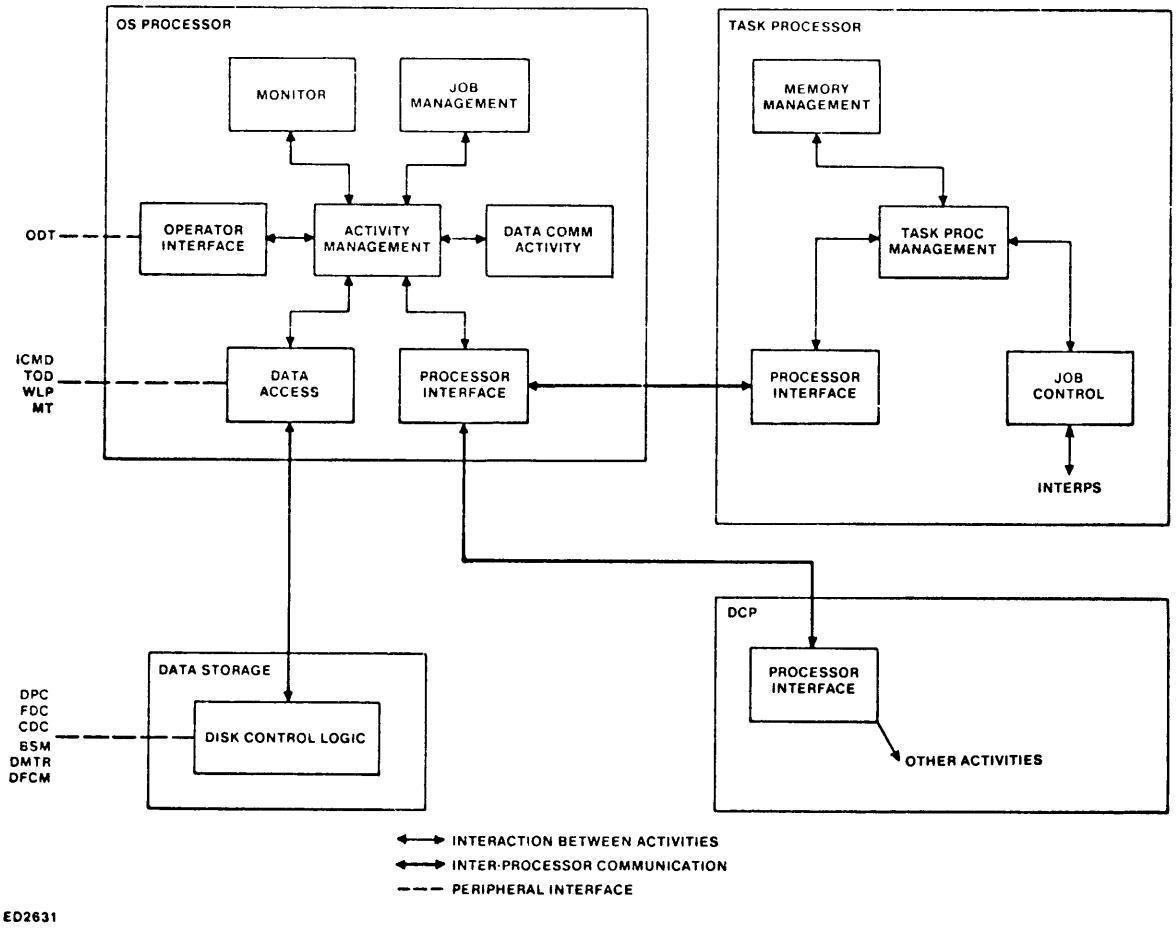


Figure 3-1. Software Functional Block Diagram

OPERATING SYSTEM (OS)

The operating system resides in one processor/memory set. This processor maintains an interface to all I/O devices except the data storage subsystem and the data communication lines. The operating system is divided into the following modules:

1. Operator Interface (OI).
2. Data Access (DA).
3. Job Management (JM).
4. Processor Interface (PI).
5. Monitor (MN).
6. Activity Management (AM).
7. Data Comm Activity (DCA).

OPERATOR INTERFACE (OI)

The operator interface module provides a communications interface between the system and a variety of operating processes; an example of such an interface is that between an operator and a user program. Included in the duties of this module are:

1. Collecting information.
2. Syntaxing messages.
3. Formatting messages.
4. Routing messages.
5. Providing system information to the operator.
6. Handling all I/O to the Operator Display Terminal (ODT).

DATA ACCESS (DA)

The data access module provides the logical/physical interface for I/O devices (line printer, ICMD), and the logical interface to the data storage subsystem. Included in the duties of this module are file management and buffer memory management.

JOB MANAGEMENT (JM)

The job management module provides an orderly system environment for user jobs. It tracks the state of each job from beginning to end, and synchronizes the occurrence of multiple commands to the same job.

PROCESSOR INTERFACE (PI)

The processor interface module, in conjunction with the corresponding module in other subsystem processors, provides the processor-to-processor communication link. Through the use of memory mailboxes, messages are transferred between processors.

MONITOR

The monitor module provides the capability to record, analyze, and report system events.

ACTIVITY MANAGEMENT (AM)

The activity management module provides the environment needed for intermodule cooperation. This environment supplies the interfaces needed to access system resources and mechanisms which enable the OS modules to communicate with one another. It could be loosely regarded as the interpreter of the remainder of the system.

DATA COMMUNICATION ACTIVITY (DCA)

The DCA module comprises four major submodules:

1. Data Comm Communicate Handler (DCCH).
2. Data Comm Loader (DCL).
3. Result function.
4. End-of-job function.

The data comm communicate handler (DCCH) provides the logic necessary to perform the functions indicated by a data comm communicate. These functions can be issued by an MCS or by a data communications user program. The primary function of the DCCH is to validate parameters, enforce user-defined limits, provide message routing to/from the data communication programs, and control the message interface to the DCP.

The data comm loader primes the necessary control and routing tables to be used by the other DCA modules, and loads the DCP microcode and NDL tables.

The result function provides message handling for messages sent from the DCP. This module routes messages and control information to the assigned programs.

The end-of-job function performs the necessary housekeeping and deallocation of resources when a program terminates.

DATA STORAGE SUBSYSTEM

The data storage subsystem resides in a processor/memory pair separate from the operating system. It works in conjunction with the operating system (OS) to provide the system with a highly efficient mechanism for controlling high speed storage media. Logical as well as physical control routines are incorporated into this subsystem. In addition, the data storage subsystem controls the display and maintenance routine (DMTR) module. All disk subsystems, other than the industry-compatible mini disk (ICMD), are connected to the data storage and maintenance processor (DS&M).

THE TASK PROCESSOR (TP)

User programs are executed in a separate processor known as a task processor. This processor is managed by the OS processor and requires the OS processor to initiate and control all its I/O requests. Jobs are assigned to the task processor by the job management module of the OS processor. The complete set of necessary firmware is known as the Interpreter Control Program or ICP. This set of firmware falls into two categories: control/support logic and language interpreters. The control/support logic can be further characterized as follows:

1. Operating system communication (PI).
2. Local memory management (PM, VM).
3. Job Control (JC).
4. Processor Management (PM).

The MPLII interpreter is required for execution of system utility programs. The COBOL/RPG interpreter is a user option that is specified in the WARMSTART information area of SYSCONFIG.

DATA COMM PROCESSOR (DCP)

The DCP firmware is produced by using the NDL compiler and the NPC900 program, and resides in the DCP. The DCP is composed of three major modules:

1. Scheduler.
2. Host control.
3. Line manager.

The internal organization of the DCP is shown in figure 3-2.

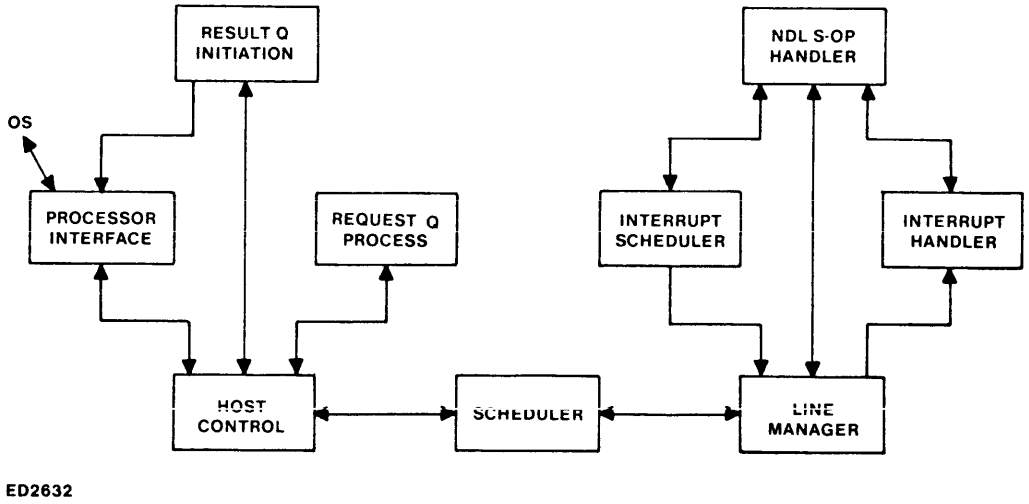


Figure 3-2. DCP Internal Organization

SCHEDULER

The scheduler module assigns processing time to the line manager and host control on a predetermined basis.

HOST CONTROL

The host control module is responsible for all external interfaces and provides support for:

1. Request queue processing.
2. Result queue initiation.
3. Processor interface.

REQUEST QUEUE PROCESS

The request queue process submodule is primarily responsible for routing messages from the DCCH portion of the OS data comm activity module to the directed station. Additionally, it initiates station and line functions as requested.

RESULT QUEUE INITIATION

The result queue initiation submodule takes complete messages, station functions, or line functions, and places them on the data comm activity (DCA) result queue, indicating that a complete message has been accumulated from a data comm line or that the requested station or line function has been handled. It then signals the DCA via PI.SEND.

LINE MANAGER

The line manager module determines if a given data comm line is currently active, and yields control to the appropriate NDL process (s-op handler, interrupt scheduler, or interrupt handler) at the appropriate time. If the NDL process module is in control and an interrupt or a pause s-op is detected, the line manager module transfers control to the scheduler module. When the expected interrupt occurs, the line manager returns control to the NDL process.

THE NDL PROCESS

The NDL process is logically partitioned into three sections: s-op handling, interrupt scheduling, and interrupt handling.

The s-op handler performs logical functions and initiates control of the physical data comm line. When code being processed by the s-op handler requires or anticipates an external event, control is relinquished to the interrupt scheduler.

The interrupt scheduler conditions the necessary hardware and logic functions to expect the event.

The interrupt handler is invoked when the event (interrupt) occurs on the data comm line adapter. The interrupt handler performs implicit functions such as parity generation and checking, format validation, and translation. The interrupt handler invokes the NDL s-op handler to return the results of the last event that occurred.

SECTION 4
INTERPROCESSOR COMMUNICATION (PI)

This section is intended to describe the mechanisms used in interprocessor communication. The specific uses made within each processor are described in section 10 of this document.

Interprocessor communication utilizes the processor interconnect bus hardware via the interface control (not covered in this manual). The interested reader is referred to the B 900/CP 9500 Field Engineering Technical Manual Volume 3 (form 1129418).

The interface between the DS&M and the OS does not use the mechanisms described in this section. A special interrupt mechanism exists between the DS&M and the Data Access activity of the OS.

MAILBOXES

The task processor and the DCP both have two mailboxes. The "output mailbox" is used for communications to the OS processor, and the "input mailbox" is used for communications from the OS processor.

Each mailbox carries traffic in one direction, but has a standard layout of bytes and their functions:

Field	Bytes	Use
Text	16	Contains the actual text passed. For a CMS communicate this will be the communicate parameter area (CPA).
Request function	1	This is the action id or name that this message was directed to.
Request processor	1	This is the bus address that this message was directed to.
Result processor	1	This is the bus address that this message originated from.
Result function	1	This is the action id or name that this message originated from.
System mix number	1	Where appropriate, this will contain the task id of the job initiating this communication.
Reserved	1	
Flags	1	These flags define the status of the mailbox. The values are as follows: MSB 7 = Request message 6 = Unused data comm processor 5 = Task processor error 4 = Task processor rejected 3 = Operating system accepted 2 = Given to operating system 1 = Task processor accepted LSB 0 = Given to task processor

The request and result function fields contain an action name or action id dependent on the original request. In general, a response to a previous communication is directed to an action id, and an original request is directed to an action name. For CMS communicates, this id is @FF@.

The flags are used to specify current ownership of the mailbox. The OS processor always uses the mailbox in the other processors.

See Section 10 of this manual for further details of mailbox functions.

SENDING A MESSAGE

OS TO TP

The procedure is as follows:

OS 1. Check status of input mailbox of desired TP. If not ok then wait.

OS 2. When ok, insert text. Reset OS accepted. Set given TP.

OS 3. Interrupt TP. Wait.

TP 4. If handler not available, mark mailbox as TP rejected, interrupt OS; OS will reset TP rejected, set OS accepted. Wait, then retry.

TP 5. If handler is available, will set TP accepted, initiate appropriate function, and interrupt OS; OS will mark box as OS accepted, and finish the send function.

TP TO OS

The flow is almost a reverse of the above:

TP 1. Set up output mailbox (if owned by TP).

TP 2. Set given OS, interrupt OS.

OS 3. Copy message in, set given TP, interrupt TP.

TP 4. Set TP accepted, inform caller.

OS TO DCP

OS 1. If OS doesn't own mailbox, wait.

OS 2. Copy message into mailbox.

OS 3. Set given TP, request message, reset other flags.

OS 4. Interrupt DCP.

DCP 5. Set given OS, reset request message, interrupt OS.

OS 6. Set OS accepted.

DCP TO OS

DCP 1. If output mailbox is not owned by the DCP, look for something else to do.

DCP 2. Else, set up mailbox, clear flags.

DCP 3. Set given OS, interrupt OS, forget it.

OS 4. Copy in, set given TP. Continue.

TRANSFERRING DATA

Data transfer is not a PI function; however, interprocessor communication is usually needed before such a transfer can take place.

Data is transferred via the bus under the control of the OS processor. During transfers from disk to any memory, the requesting processor is responsible for freezing the appropriate memory area, and is expected to leave it alone until informed otherwise by the OS processor. This mechanism is transparent for data file I/O as buffer memory is an OS-managed entity, but is not transparent for code or data segment I/O.

Note that any processor may attempt a remote access. Each processor is required to adhere to the restrictions imposed by the OS processor.

SECTION 5
SYSTEM SOFTWARE THEORY OF OPERATION

OS OVERVIEW

The OS may be considered to operate using three levels:

Activities

Actions

The kernel (nucleus)

ACTIVITIES

Activities are groups of actions and are committed to specific functions, specifically:

OI -Operator Interface

JM -Job Management

DA -Data Access

DC -Data Comm

MN -Monitor

AM -Activity Management

ACTIONS

An action is a region of code which may be given control of the OS processor. In practice, an action is a procedure that is executed by the Activity Management procedure "INVOKE." The different types of actions are defined as follows:

AVAILABLE ACTION: A region of code that is called upon to perform some specific functions.

EXECUTING ACTION: An instance where it was decided to use a specific available action.

CRITICAL ACTION: An action that cannot be invoked if it is already executing.

CONTINUOUS ACTION: An action that never yields the processor voluntarily.

KERNEL

The kernel, or nucleus, is a microcoded area that contains commonly used routines. Some of these routines have responsibility for interactivity communication. With the exception of the monitor activity, no activity may directly contact another, it must call a specific nucleus routine to perform the switch. (See figure 5-1.)

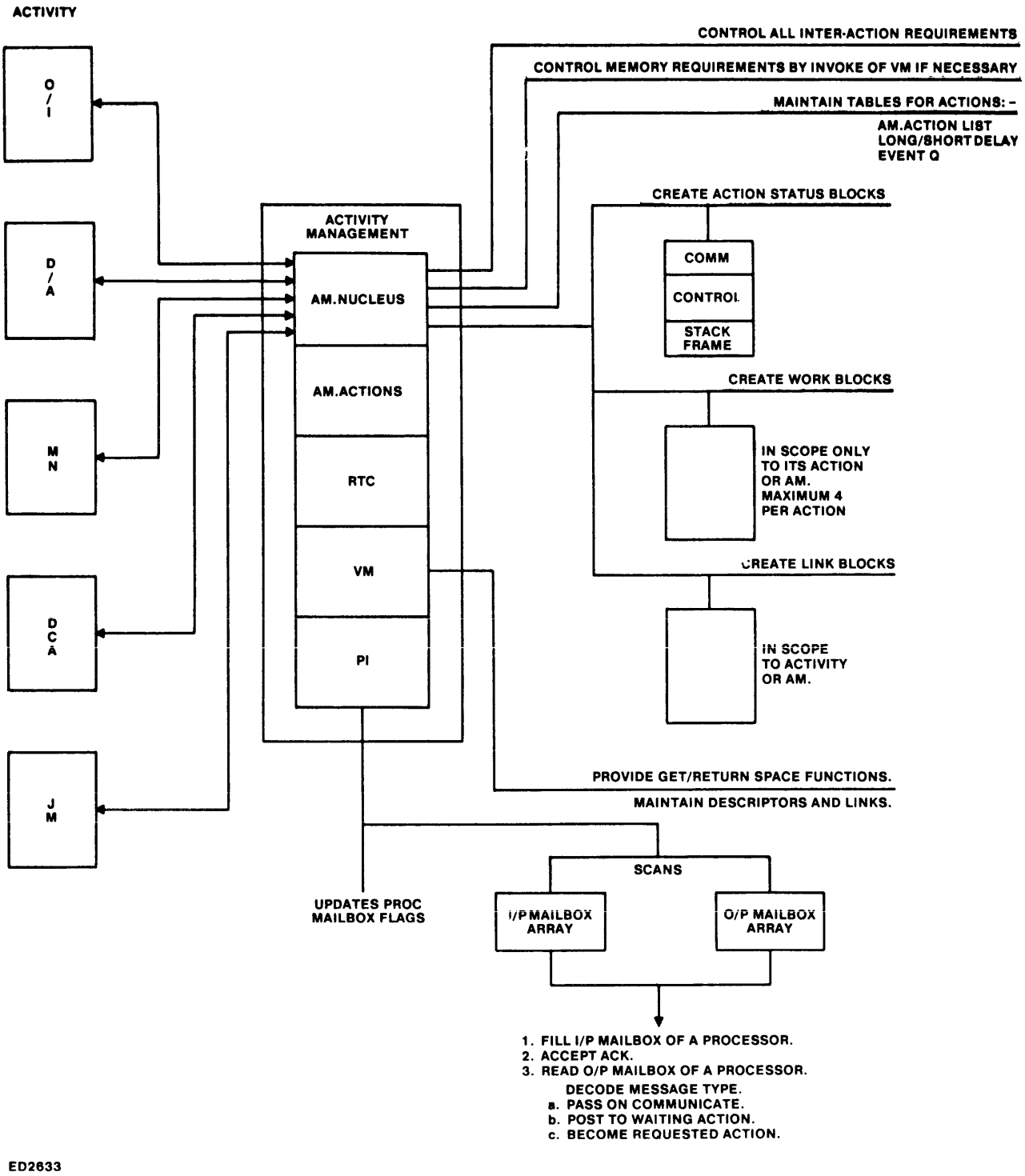


Figure 5-1. OS Overview

The nucleus is considered part of the AM (Activity Management) activity, and is permanently resident along with the other activity management functions: AM actions, real time clock (RTC), virtual memory (VM), and processor interface (PI) at the low end of page 0.

Routines within it maintain tables, lists, and queues on behalf of AM, and also control the invocation, scheduling, and deallocation of actions. These routines will apply available memory for use by actions, invoking if necessary the more complex VM routines to make the space.

An area known as a "status block" is created for every action. The size of a status block is specific to a given action. As shown in figure 5-2, each status block is divided into three logical sections: comm area, control area, and stackframe. Of these, only the size of the stackframe may vary from action to action. The status block, as its name implies, is used to hold the status of the action when it becomes suspended, as well as the communication parameters it receives or wishes to send to another action or micro routine.

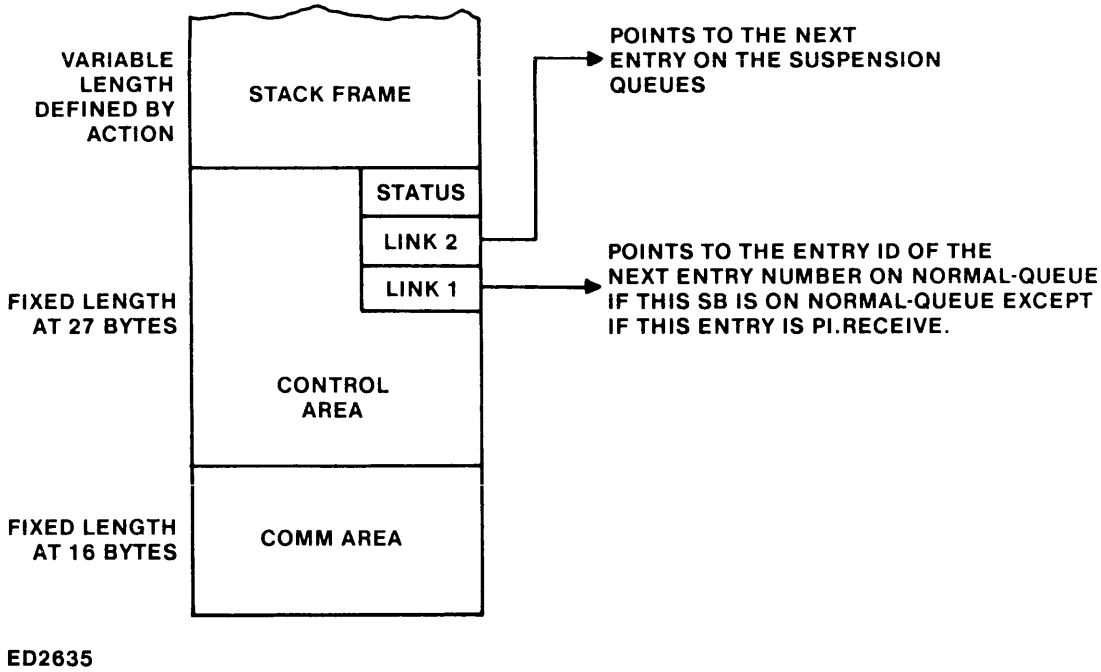


Figure 5-2. Status Block Layout

One to four workblocks may be associated with an action. They are accessible only to the action and activity management, and are used to hold the data pertinent to the action. It is possible for a workblock to hold data which may be used to supplement comm area data when one action calls another.

A link block is a work block which an action or AM has deemed necessary to be in scope to the whole of an activity. Before an action may manipulate the contents of a link block, its status must be returned to that of a work block.

VM

VM is responsible for making space if required, maintaining descriptors and links for these, and returning space.

PROCESSOR INTERFACE (PI)

PI controls interprocessor communication. It scans two arrays, one for input, one for output. On detection of a processor's readiness to send or receive a communication, it will:

1. Read that processor's output mailbox.

2. Write to that processor's input mailbox if a message is available.

Message transfers in either direction are monitored by a system of acknowledgements.

AM_ACTION_LIST POINTERS

In order to keep track of the location of the status blocks, a list of pointers to them is kept resident in memory. The maintenance of this list is the responsibility of routines within the nucleus. The list consists of 100 two-byte entries @00@-@64@. Each entry is the absolute memory address of its associated status block. The first fifteen entries are for resident actions (one continuous, and fourteen critical). Refer to figure 5-3.

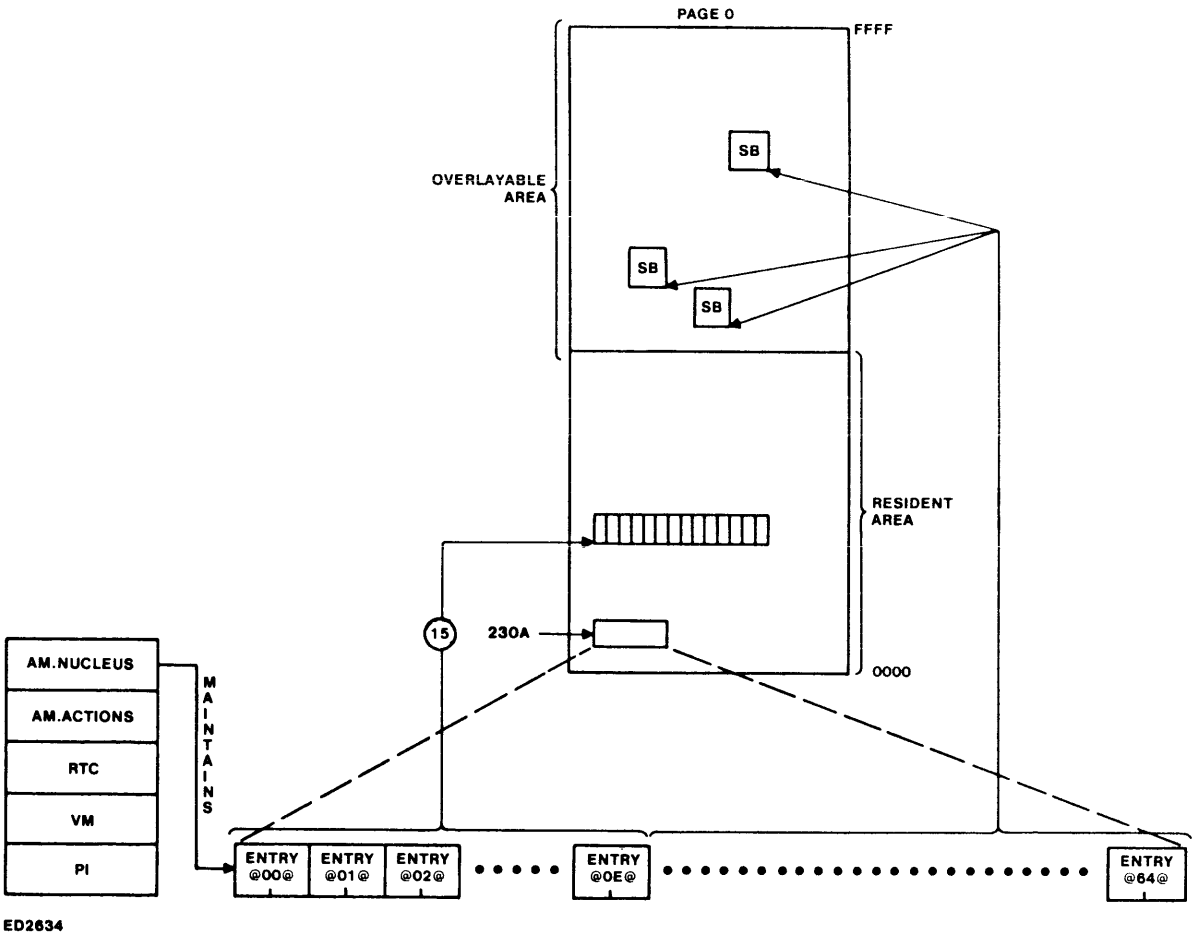


Figure 5-3. Action.List Pointers

The continuous action may, in fact, be any of four which are designated "continuous." Only one may run at a time, so they share a status block. This status block is permanently located in a fixed location within AM resident area. The continuous action will always run to completion without yielding the processor to another action. Entries @01@ thru @0D@ are for the "critical" actions; @0E@ is a critical action reserved for future use. These are actions of which only one invocation may exist at any time. These actions may yield the processor.

Entries @0F@ thru @64@ are assigned on an as-required basis. They will always be in overlayable memory. However, all status blocks with their associated work and link blocks, will always be located on page 0.

STATUS BLOCK LINKING

OS operation consists of the serial running of actions, with interspersed nucleus code. Within the nucleus is "scheduler" code, which determines the order in which actions are given control. Scheduler scans through the status blocks to find one which indicates by its status byte that the associated action is ready to run.

All current actions will be linked together in a circular list, only critical actions which are active will be included. The linking mechanism is the "LINK1" byte, containing the "entry number" of the next action on the list. (Refer to figure 5-4.)

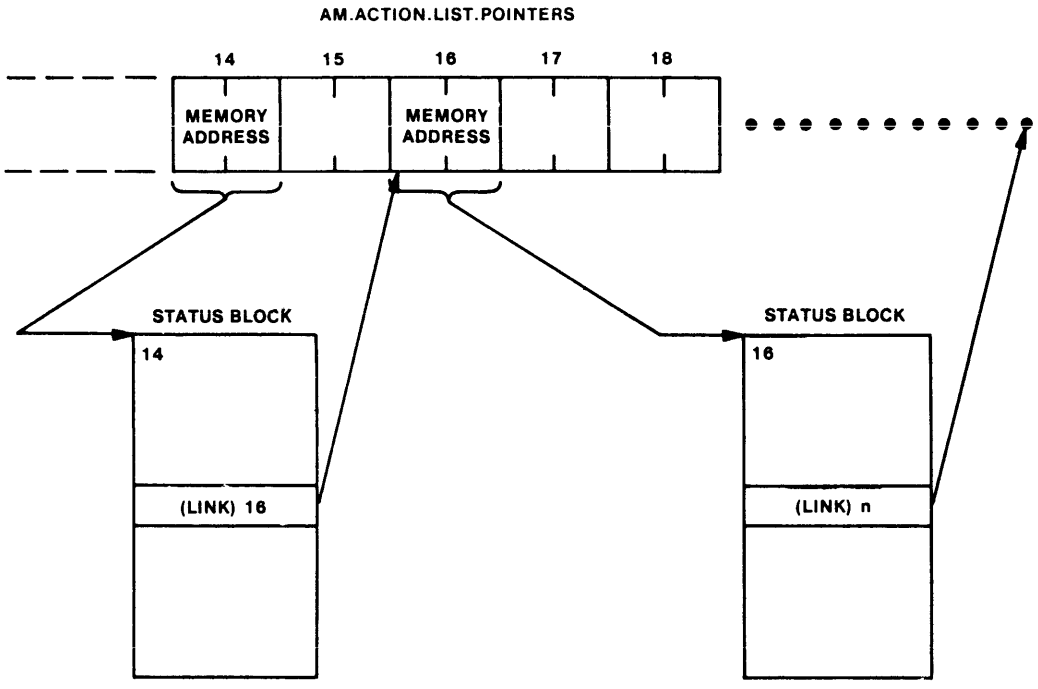


Figure 5-4. Status Block Linking

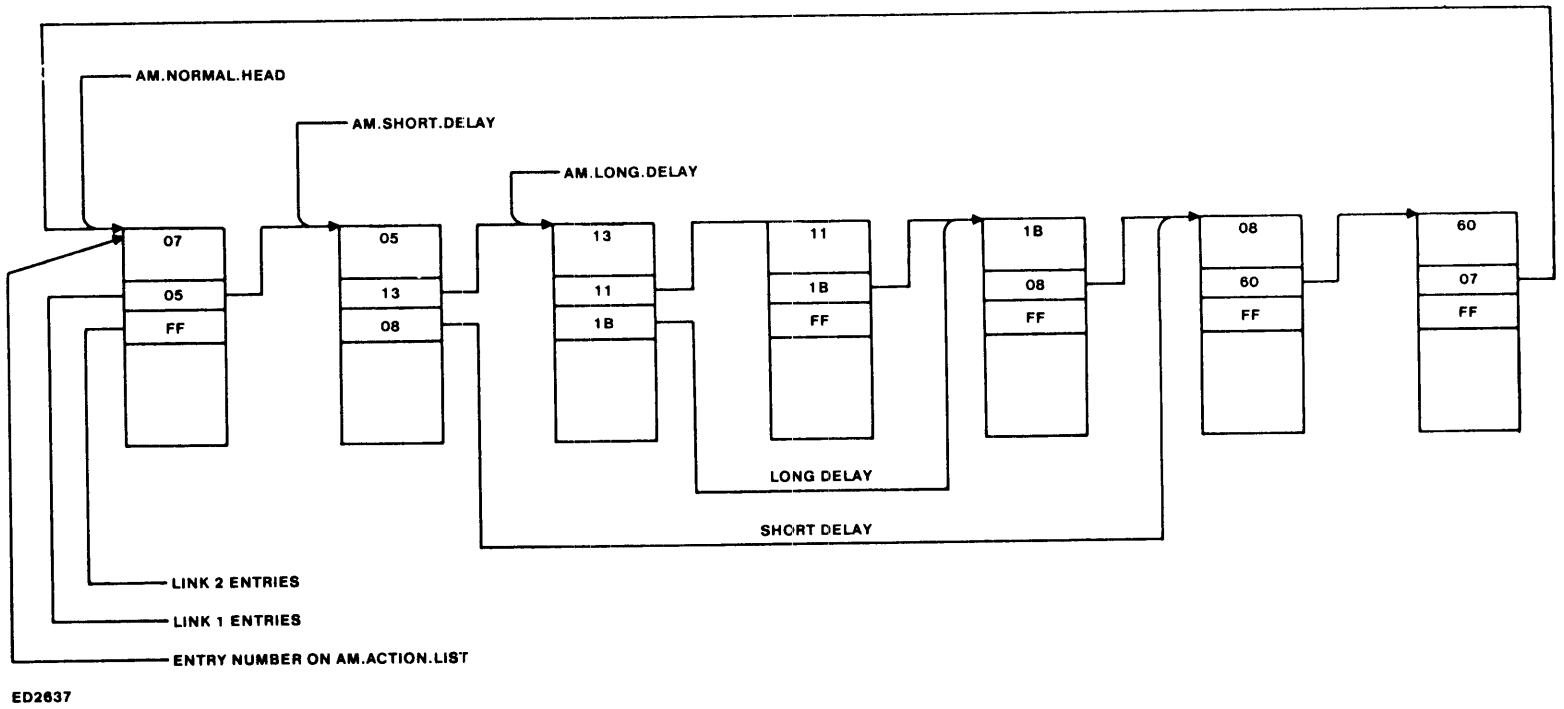


Figure 5-5. Normal and Suspension Queue Links

It may be that an action wishes to invoke a critical action which is already running, or wait for some system event to occur. In these cases, a queueing mechanism is used, and the entry number or the next action in the queue will be in the status block "LINK2" byte. There are a large number of possible queues, as can be seen from a look at the dump. The above description is only intended to describe the linking mechanism for each queue, as shown in figures 5-5 and 5-6.

The entry number, then, is used as an index into the AM.ACTION.LIST pointer table which contains the absolute memory locations of the status blocks of current actions.

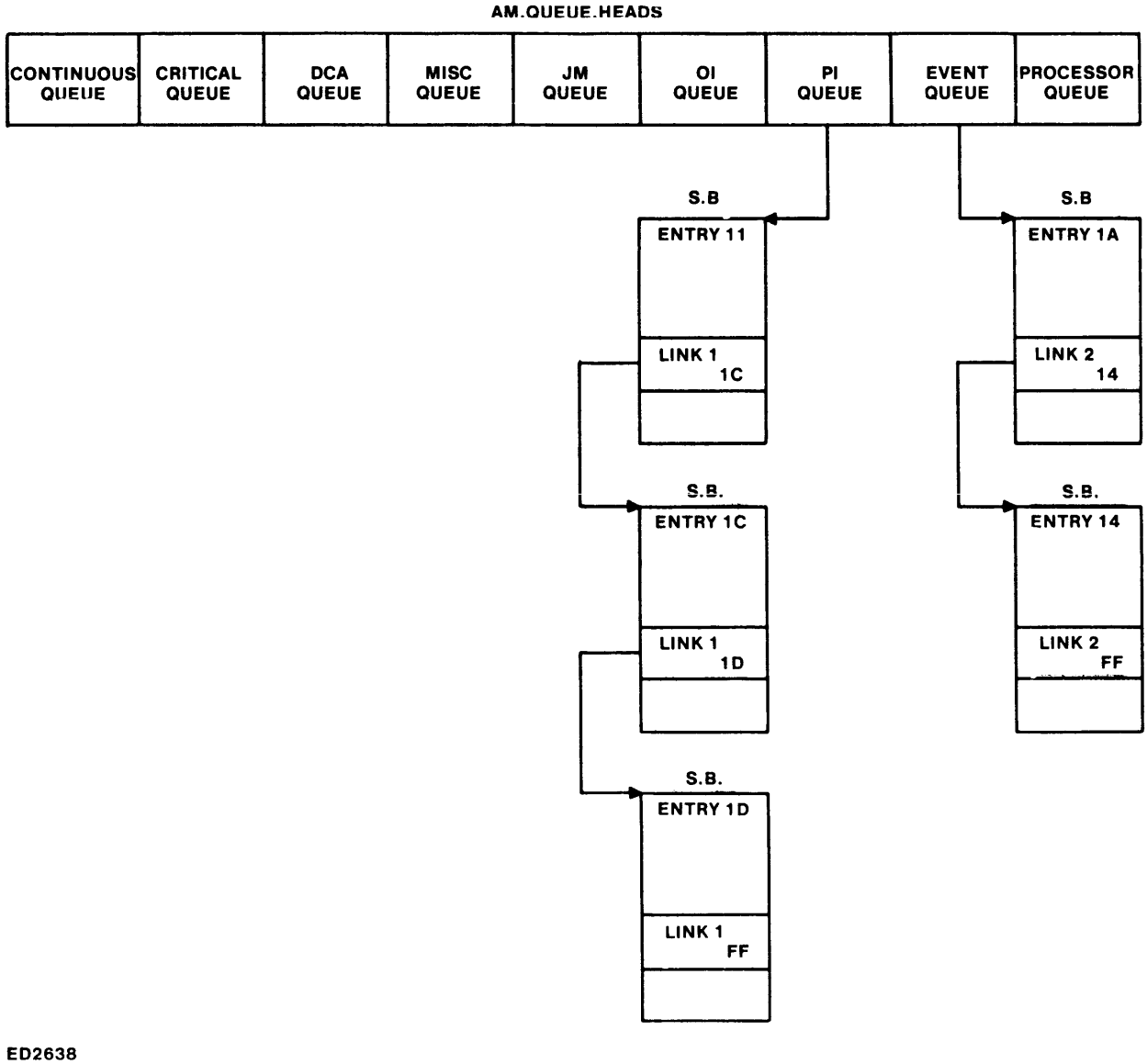


Figure 5-6. AM.QUEUE.HEADS

LINKING

Structures within overlayable memory are linked together in two distinct lists, depending on their form. Although the linking system is basically similar, there are some differences as outlined below.

AVAILABLE

Each available block of memory contains within it a descriptor. This descriptor is five bytes long and is the first five bytes in the block. It contains type (available), size (physical), and absolute address of next available descriptor. The chain starts from the available head pointer which points to the available table. This table contains one entry for each page of memory; thus, available memory linking is on a per-page basis. The entry in the table for each page points to the descriptor of the first piece of available memory on that page. The last available area on a page is linked to @FFFF@.

WORK BLOCK/LINK BLOCK

Each work or link block is immediately preceded by a three-byte descriptor indicating type and size. The main information about the block is held in a 10-byte descriptor. This descriptor also holds a pointer to the next descriptor in the chain. The head of the chain is pointed to by the "WB.LB Pointer." (Refer to figure 5-7.)

It should be noted that the size may be indicated differently in the 10-byte descriptor and the three-byte descriptor. The three-byte descriptor contains "physical size," while the 10-byte descriptor contains "logical size." The discrepancy occurs as follows:

1. Area of 200 bytes needed.
2. Thirteen bytes must be added for descriptors; look for 213 bytes.
3. Find available of 217 bytes.
4. Since surplus bytes <5 (minimum requirement for "available" descriptor), use this area.
5. Create and mark 3-byte descriptor with size = 207 bytes (physical).
6. Create and mark 10-byte descriptor with size = 203 bytes (logical) and pointer to start of area (not to 3-byte descriptor).

NOTE

Such a three-byte descriptor always immediately precedes the area it describes. The 10-byte descriptor may be anywhere.

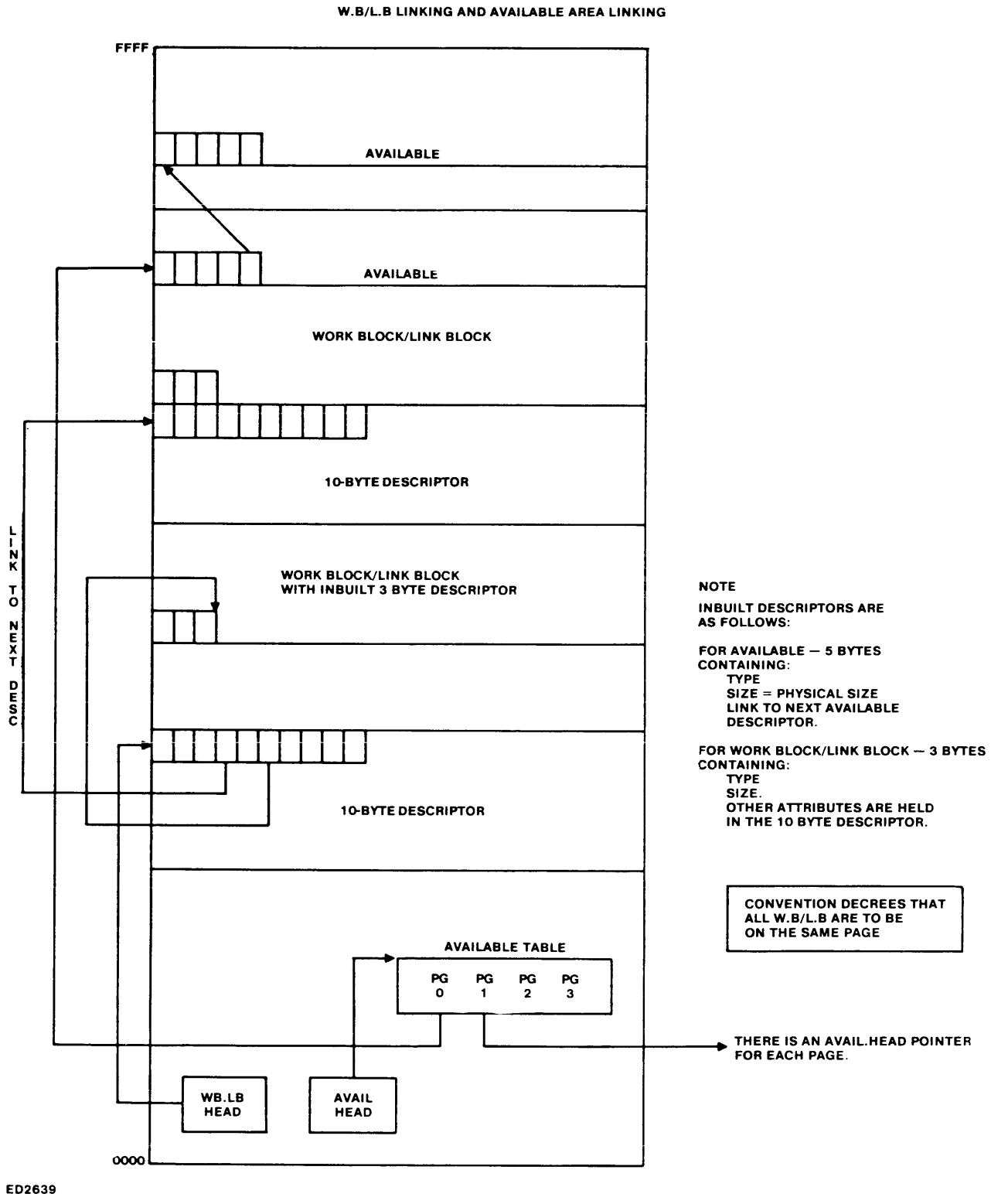


Figure 5-7. W.B./L.B. and Available Area Linking

SEGMENT TABLE

All OS data segments must reside on page 0. Code segments may be on any page. A segment table exists in the resident area for determining the presence and location of either structure. (Refer to figure 5-8.) This table contains 136 entries which for any given release will be in the same order. It is indexed into by the segment number. Before accessing a segment, a test is performed on its descriptor to determine if that segment is present. If so, its location in memory, held within the descriptor, is valid. If not, the disk address and size will indicate its location on disk. An absent segment will have a memory address of @FFFF@.

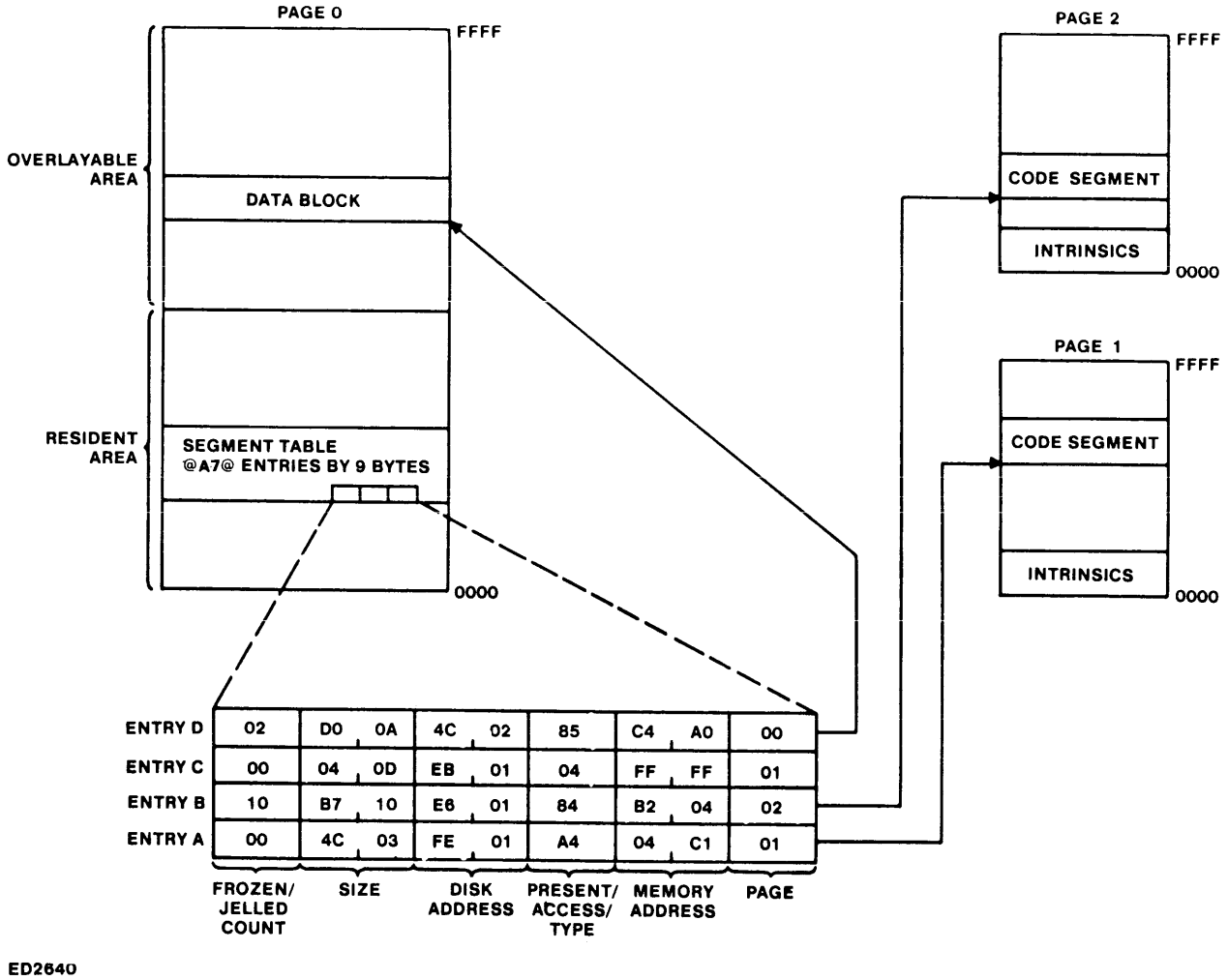


Figure 5-8. Segment Table Block Diagram

TASK PROCESSOR

Communications between the OS and the TP take place through PI, except for movements of data which are initiated by a PI message but are performed by actions internal to OS or TP.

Messages received from the input mailbox of the TP go to one of four functions. These functions are interpreter load, monitor, command handler, or data move.

The interpreter load function is responsible for loading the interpreters at warmstart time. The monitor function handles polling from the OS. The command handler function breaks down its own parameters to determine the precise request being made by the OS. Data move is responsible for the movement of data between the OS and the TP.

Responses to OS communications are returned to the OS by way of PI. The action in the OS to which the response is to be routed is part of the data contained in the output mailbox of the TP.

The TP does not run with the same action structures as the OS. Only the interface mechanisms are common to both. The rest of the TP may be considered a completely separate entity. Figure 5-9 is a block diagram of the TP, showing the OS-to-TP interface, the TP functions, and the TP-to-OS interface, via PI.

A task record is maintained for each TP task. Only two tasks are considered to be "resident" and have task records permanently assigned These tasks are named "OS REQUEST TASK" and "TERMINATE TASKS TASK." Entry numbers indicating TP-relevant tasks are found in the JOB QUEUE. A task is a pointer to a system task, and may be converted to a user mix number by use of the mix table.

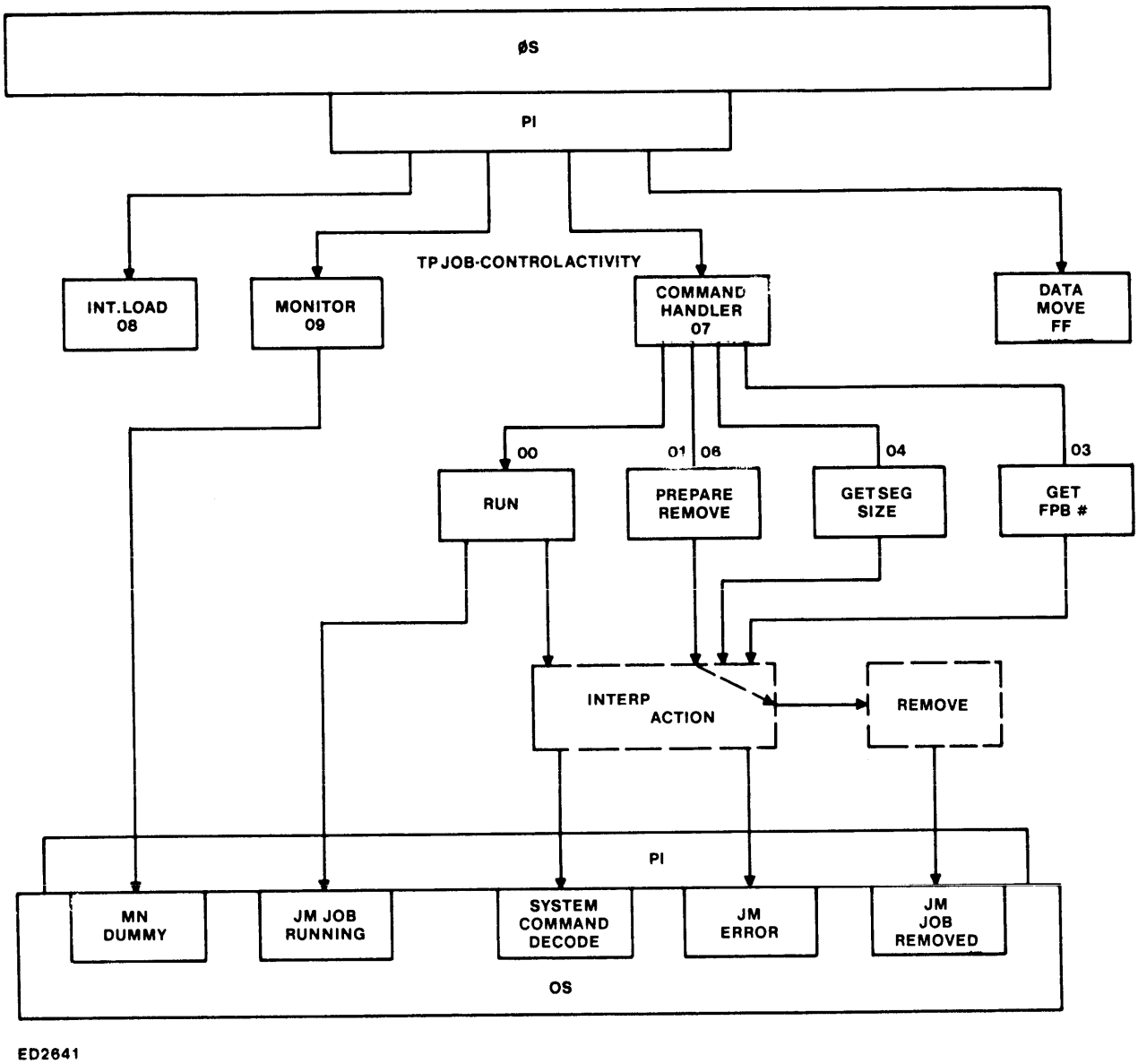


Figure 5-9. Task Processor Job Control Activity

SECTION 6
HEADING INFORMATION

INTRODUCTION

This section presents a description of the first four pages encountered in a system dump. The first page is the title page, which identifies the dump file being analyzed, and indicates when the dump was created. The second page, titled "Dump File Parameters," contains data detailing the structures found within the dump. Page three is a brief section containing dump file map definitions, and page four is the dump file map. The dump file map provides the status of the processors and associated memories at the time the dump was taken. The relevant fields in initial dump analysis are discussed in the following paragraphs in the order in which they appear in the dump.

TITLE PAGE

The title page contains information relevant to the dump file being analyzed. The initiating message to the SYSANALYZER program is printed first. This will indicate which portions of the dump were requested to be printed. The fields "dump file creation date" and "dump file creation time" can be used to determine if the dump was created by the ROM/PRAM routine or by the system. If these fields contain a valid date and time, the dump was created by the system. This could have occurred either automatically, due to a system failure, or by the operator's use of the GT MD intrinsic. If the date is given as 11 11 11 and the time is given as 24.00.00, the dump was most likely created by the ROM/PRAM dump routine. It is also possible that the failure occurred prior to the input of the date. On a remote SPO system it is not mandatory to enter in the date and time; therefore, the dump could contain 11 11 11 as the date and 24.00.00 as the time. Refer to figure 6-1 for an example of a title page.

```
BOJ MESSAGE
                    ALL,HEX
                    DUMP FILE PACK ID     0000000
                         DUMP FILE ID     SYSDMFILE00
DUMP FILE CREATION DATE                   82 08 13
DUMP FILE CREATION TIME                   14 12 05
                    ANALYSIS DATE         82 08 13
                    ANALYSIS TIME         14 13 04
                  REVISION LEVEL          03.04.10
```

Figure 6-1. Title Page Messages

DUMP FILE PARAMETERS

This area contains pointers to various structures within the operating system, most of which are used in a hexadecimal dump analysis. Some of the fields in the dump file parameters sections are irrelevant for an initial dump analysis. Those that are most important are:

Version String
System Error Value
Frozen Processor
Frozen State

The dump file parameter fields are discussed in the following paragraphs in the order in which they appear in the dump file. Refer to figure 6-2 for a sample dump file parameter page.

ANALYZER LEVEL

The analyzer level is a one-byte field which is checked by SYSANALYZER. If the level in the dump file does not match the SYSANALYZER level, an analysis of the dump will not be performed.

VERSION STRING

This field identifies the level of the MCP in use when the dump file was created.

SYSTEM ERROR VALUE

This field gives the first two digits (byte) of the clearstart number if the dump was created by a clearstart (refer to Appendix A for clearstart numbers and their meanings); otherwise, the field contains @FF@.

FROZEN PROCESSOR

This field gives the bus address of a processor that has been frozen by the operating system. Other sections of the dump may show why the processor was frozen.

FROZEN STATE

The contents of this field indicate the state of the frozen processor identified in the preceding field, "frozen processor." The values for the processor state are found under "Dump File Map Definitions." It should be noted that a fatal failure in a task or DCP processor always results in an attempt to produce a system dump file.

PROCESSOR TYPES

The processor types field is a 16-byte array. Each byte represents a zero-relative bus address. An unused entry contains @FF@. Valid entries are described under Dump File Map Definitions.

REMAINING DUMP FILE PARAMETERS

The remaining dump file parameters are used for in-depth analysis, generally in conjunction with a hex dump.

```
********************
DUMP FILE PARAMETERS
********************

                    ANALYZER LEVEL  03
                    VERSION STRING  3033 3034 3233 3033 3230 3130 3030      030423032010000
                 SYSTEM ERROR VALUE  FF
                   FROZEN PROCESSOR  FF
                       FROZEN STATE  FF
                    PROCESSOR TYPES  FF01 0304 0303 0305 02FF FFFF FFFF FFFF
                         DATE & TIME  0813 8219 1412 0572
                ACTION LIST POINTER  0A23
                  NUMBER OF ACTIONS  64
                  COMM BASE POINTER  0F5C
                 CURRENT ID POINTER  115C
                     VM BASE PAGE 0  1384
                VM BASE NON PAGE 0   4803
              WAITING HEAD POINTER   3024
               NORMAL HEAD POINTER   2F24
                QUEUE HEAD POINTER   3124
      NUMBER OF CRITICAL QUEUES      0F
              LAST QUEUE POINTER     7E24
      NUMBER OF SCHEDULE QUEUES      0D
              LONG DELAY POINTER     3B5C
             SHORT DELAY POINTER     3C5C
              WB/LB HEAD POINTER     EA23
                WB/LB ENTRY SIZE     0A
           AVAILABLE HEAD POINTER    EC23
             AVAILABLE ENTRY SIZE    05
            SEGMENT TABLE POINTER    C35E
            SEGMENT TABLE ENTRIES    A7
               SEGMENT ENTRY SIZE    09
     INTERRUPT HANDLER POINTER       B924
        INTERRUPT HANDLER SIZE       0E
                   MAX USER JOBS     20
                DC DATA POINTER      D06A
             DC DATA END ADDRESS     566B
  BUFFER QUEUE ADDRESS POINTER       4D6B
              DC POINTER POINTER     496B
    HISTORY COMM TABLE POINTER       8F46
                      PI POINTER     1328
                           ID WS     9D
                           ID JM     9F
                ID DA PROG NAMTAB    A0
              ID DA INTERP NAMTAB    A1
                ID DA MISC NAMTAB    A2
                      ID DA LWAIT    A3
                           ID OI     A4
                           ID SM     A5
                  DC LINE POINTER    A6
        NUMBER OF EVENT QUEUES       21
              PORT TABLE POINTER     EF64
      PERIPHERAL TABLE POINTER       3A65
                  DA.CT.POINTER      4267
                     ID DC LIST      51
               DA BUFFER POINTER     DF66
             NUMBER OF PI QUEUES     0F
                     ODT SEG PTR     926F
               PERIPHERAL LOCKS      0857
```

Figure 6-2. Dump File Parameters

DUMP FILE MAP DEFINITIONS

The type key may indicate the type of area that is dumped, or it may indicate termination. The type key values are as follows:

@01@: Page of memory

@02@: Data block/work block/linked block area. Data block is part of the operating system's virtual memory file; work block/linked block are defined in Section 3.

@03@: Terminated with no errors

@04@: Terminated due to parity errors

The validity flags and their meanings are:

@FF@ Incomplete: The area that follows is the last area dumped, and is probably incomplete because of a system or disk crash.

@00@ Valid: The area that follows has been completely dumped with no errors

@01@ Invalid: The area that follows had disk errors during dump. Go on to the next id sector. The same area will be dumped again. Only one retry will be done before the dump is terminated and a type key of "terminated with parity errors" is written.

The processor state values are outlined in figure 6-3.

DUMP FILE MAP

This table provides the status of all the processors and memories on the system at the time the dump was created. Using the fields bus address, processor type, and last address, the configuration of the system can be ascertained. The "Last Address" field is used to calculate the amount of memory assigned to each processor. A normal 64K page will have the value @F0FF@ (byte-reversed), and for depopulated boards, the last address should be on a 16K boundary. If the last address is any value other than previously stated, it means that an error was detected when attempting to dump that page. The error being detected is at the address contained in "Last Address." The amount of buffer memory assigned can be ascertained by a processor type of 05 (BM). In a ROM/PRAM dump, buffer memory is not evident. A dump file map is shown in figure 6-4; see figure 6-5 for an example of a processor status decode section.

```
**************************
DUMP FILE MAP DEFINITIONS
**************************

                              TYPE KEY
                                        01 - PAGE OF MEMORY
                                        02 - DATA BLOCK
                                        03 - TERMINATED WITH NO ERRORS
                                        04 - TERMINATED DUE TO PARITY ERRORS

                            VALIDITY FLAGS
                                        00 - VALID
                                        01 - INVALID
                                        FF - INCOMPLETE

                            PROCESSOR STATE
                                        00 - FROZEN   / REGISTER STATE SAVED
                                        01 - FROZEN   / REGISTER STATE UNSAVED
                                        02 - UNFROZEN / SOFTWARE ERROR
                                        FF - O.K.
```

Figure 6-3. Dump File Map Definitions

****
DUMP FILE MAP
****

KEY	FLAGS	BUS ADDRESS	PAGE ADDRESS	PROCESSOR STATE	PROCESSOR STATUS	PROCESSOR TYPE	LAST ADDRESS	DISK RECORD
***	*****	*******	*******	*********	*********	*********	*******	******
01	00	01	00	FF	810211B1	OS 01	FOFF	0003
01	00	01	01	FF	810211B1	OS 01	FOFF	0171
01	00	01	02	FF	810211B1	OS 01	FOFF	02DF
01	00	01	03	FF	810211B1	OS 01	FOFF	044D
01	00	02	00	FF	81008FA2	TP 03	FOFF	05BB
01	00	02	01	FF	81008FA2	TP 03	FOFF	0729
01	00	02	02	FF	81008FA2	TP 03	FOFF	0897
01	00	02	03	FF	81008FA2	TP 03	FOFF	0A05
01	00	03	00	FF	81008F73	DK 04	FOFF	0B73
01	00	04	00	FF	81008F64	TP 03	FOFF	0CE1
01	00	04	01	FF	81008F64	TP 03	FOFF	0E4F
01	00	04	02	FF	81008F64	TP 03	FOFF	0FBD
01	00	04	03	FF	81008F64	TP 03	FOFF	112B
01	00	05	00	FF	81028F05	TP 03	FOFF	1299
01	00	05	01	FF	81028F05	TP 03	FOFF	1407
01	00	05	02	FF	81028F05	TP 03	FOFF	1575
01	00	05	03	FF	81028F05	TP 03	FOFF	16E3
01	00	06	00	FF	81008F06	TP 03	FOFF	1851
01	00	06	01	FF	81008F06	TP 03	FOFF	19BF
01	00	06	02	FF	81008F06	TP 03	FOFF	1B2D
01	00	06	03	FF	81008F06	TP 03	FOFF	1C9B
01	00	07	00	FF	2100DF17	BM 05	FOFF	1E09
01	00	07	01	FF	2100DF17	BM 05	FF3F	1F77
01	00	08	00	FF	A1008F18	DC 02	FOFF	1FD4
02	00	00					FFFF	2142
03	00	00					FFFF	22D4

Figure 6-4. Dump File Map

```
**********************
PROCESSOR STATUS DECODE
**********************

PROCESSOR ON BUS 01   NPRO CONNECTED      RAM STATE          REMOTE ACCESS      RAM FETCH
                      OS SUBSYSTEM        IOS CONNECTED      CONTROLS I/O
PROCESSOR ON BUS 02   NPRO CONNECTED      RAM STATE          LOCAL ACCESS       RAM FETCH
                      OS SUBSYSTEM        IOS CONNECTED
PROCESSOR ON BUS 03   NPRO CONNECTED      RAM STATE          LOCAL ACCESS       RAM FETCH
                      DISK SUBSYSTEM      IOS CONNECTED      CONTROLS I/O
PROCESSOR ON BUS 04   NPRO CONNECTED      RAM STATE          LOCAL ACCESS       RAM FETCH
                      DISK SUBSYSTEM      IOS CONNECTED
PROCESSOR ON BUS 05   NPRO CONNECTED      RAM STATE          LOCAL ACCESS       RAM FETCH
PROCESSOR ON BUS 06   NPRO CONNECTED      RAM STATE          LOCAL ACCESS       RAM FETCH
PROCESSOR ON BUS 07   FROZEN              RAM STATE          LOCAL ACCESS       READ WITH CLEAR
                      RAM FETCH           CONTROLS I/O
PROCESSOR ON BUS 08   NPRO CONNECTED      FROZEN             RAM STATE          LOCAL ACCESS
                      RAM FETCH           CONTROLS I/O
```

Figure 6-5. Processor Status Decode

SECTION 7
VIRTUAL MEMORY

INTRODUCTION

This section of the manual provides a field-by-field description of the Virtual Memory portion of a dump. This area of the dump is composed of six tables, which are maintained by the MCP, indicating the status of the system at the time the dump was taken. The tables are:

DA_PROG_NAMTAB - Program Name Table
DA_INTERP_NAMTAB - Interpreter Name Table
DA_MISC_NAMTAB - Miscellaneous Name Table
MIX TABLE
PORT TABLE
PERIPHERAL TABLE (PHT) - Peripheral Handling Table

Under heavy system activity, it is possible for some of these tables to be swapped out to disk (backing store). When this condition occurs, these tables are located towards the end of the SYSMCP file. In a system dump, both "Copy in Memory" and "Copy on Backing Store" are printed unless the tables have been swapped out of memory. When both are present, Copy in Memory is the most recent version and is the one which should be analyzed. Copy on Backing Store is never printed in a ROM dump. The reason for this is that the links which point to the location on backing store are cleared in the process of taking a ROM dump.

Initial analysis of the Virtual Memory section consists of scanning the MIX TABLE, PORT TABLE, and the Peripheral Table (PHT).

An examination of the MIX table shows the number of programs in the mix and their status. More specifically, the "MIX PROCESSOR" field gives the bus address of the task processor executing a specific program. The Port Table identifies the devices on the "soft" ports. The PHT indicates the physical status of all peripherals connected to the system. This table should be used in conjunction with the Device Configuration Table (DA_CT), which gives the logical status of the peripherals.

A detailed description of the virtual memory tables and their fields as they appear in a dump is contained in the subparagraphs which follow.

DA_PROG_NAMTAB

This table is a list of all open code files. A sample DA_PROG_NAMTAB is supplied in figure 7-1.

DA_INTERP_NAMTAB

This table is a list of all loaded interpreters. A sample DA_INTERP_NAMTAB is supplied in figure 7-2.

DA_MISC_NAMTAB

This table is a list of all opened miscellaneous files: for example, log files, SYSLANGUAGE, SYSDM-FILEnn, etc. Refer to figure 7-3 for an example of the DA_MISC_NAMTAB.

MIX TABLE

The mix table is an internal array record of each job. It has one entry for each program in the system. The job management activity of the operating system controls the maintenance of this table. The following fields are used by Job Management in performing its functions. These fields appear in the dump analysis in front of the first mix table entry break down.

Sample Mix Table entries are outlined in figures 7-4 and 7-5.

MAX JOBS	This field indicates the maximum number of jobs (in hex) allowed in the mix.
NULL MIX RUNNING	This field is set true (@FF@) if the current job requires a null mix in which to run.
JOB COUNT	This field indicates the number of jobs in the mix, including SUPER-UTILITY.
MCS COUNT	Contains the number of MCSs running in the mix.
TEMPORARY ID	This field temporarily holds values during the loading of a job.
NEXT MIX NUMBER	This field is the mix number of the next job to be loaded.
COUNT WAIT JOBS	This field contains the number of jobs waiting to be assigned a processor to run in.
RESTART ID	This field contains the action id of the restart action.
INTERPRETER ARRAY 0,1	This field contains the file information block ids of the interpreter.
SUPER FUNCTION	This field contains the current SUPER-UTILITY function, such as PD. If the field is blank, the SUPER-UTILITY is idle.

The following fields are contained in a mix table record:

MIX TABLE ENTRY/TASK ID	This field contains an index into the mix table. It points to the mix table entry for this job (itself).
PROGRAM FIB ID	This field contains the id of the file information block for the program (code) file. It is used in opening and closing the program file. The left hand byte is the entry number in the DA_PROG_NAMTAB.

VM FIB ID

This field contains the id of the file information block for the virtual memory file. It is used in opening the VM file. The left hand byte is the entry number in the virtual list and should be the same as the task id.

INTERPRETER FIB ID

This field contains the id of the file information block for the interpreter file. It is used in opening and closing the interpreter file. The left hand byte is an index into the DA_INTERP_NAMTAB.

MIX NUMBER

This field contains the mix number assigned to the task at load time. This is the number the user uses to refer to the job.

Note that this is in hex.

MIX FLAGS

These reflect the state of a job.

Any reasonable combination of these flags is permitted.

@8000@ : EOJ terminating

@4000@ : DP terminating

@2000@ : DS terminating

@1000@ : Terminating (accompanied by one of the above)

@0800@ : BOJ/EOJ message suppress

@0400@ : Data comm job

@0200@ : MCS

@0100@ : Multicopy

@0080@ : Thrashing

@0040@ : Null mix required

@0020@ : Paused

@0010@ : Terminate with lock

@00008@ : EX pending

@00004@ : PR pending

@00002@ : ST pending

@00001@ : GO pending

@00000@ : None of the above

Mix Flags 2

If this field = 01 then the task is a remote SPO handler.

MIX STATE

This reflects the current state of the job. possible values are:

@01@ : Loading

@02@ : Waiting (not assigned to a TP)

@03@ : Awaiting run

@04@ : Executing

@05@ : Awaiting remove

@06@ : Stopped

@07@ : Unloading

@08@ : Awaiting suspend

@09@ : Suspended

MIX PROCESSOR

This field indicates the processor which is running this job.

MIX INTERPRETER

This field holds the index into the interpreter associated with this job (index into interpreter array).

MIX PRIORITY

The low order digit of this field indicates the priority of the job as follows:

1 : Priority C
2 : Priority B
3 : Priority A
4 : Illegal priority

	A @C@ in the high order digit indicates that this job may open any file; @8@ in this digit means this job can open any file but system.
MIX TCB SIZE	This field indicates the size of the task control block. It is used by Job Management when passing the job to the task processor to run.
MIX PARENT POINTER	This field contains the task id of the parent job if this job was started by a zip with pause.
REAL STORE	Holds the working set estimate specified at job execution (if given).
SPO SIZE	Reserved for future use.

PORT TABLE

This table lists the number of devices associated with a port and the number of users for the device and the actions executing for the port.

The "index" is the index into the DA_CT table. DEVICES gives the number of devices associated with this port, thus the number of entries in the DA_CT, index 00 point to first entry in DA_CT. If DEVICES = 2, two entries will appear in DA_CT, both controlled by the controller indicated in port table TYPE column.

NOTE

Variations from above are devices that do not appear in DA_CT. They are:

```
Entry 00 = PI
Entry 01 = DP
Entry 02 = TOD
Entry 03 = ODT
```

Refer to figure 7-6 for a sample Port Table.

PERIPHERAL TABLE (PHT)

This table contains device status information. The individual fields of an entry are explained below.

The first grouping of four data items contains information about the disk file header locks. These locks prevent concurrent access to the disk file headers for a particular file during Open, Close, or Disk Area Allocation. Each lock is three parts: a SYSMEM flag, task byte, and a FILETAG.

Refer to figure 7-7 for an example of a PHT entry. An explanation of the individual entries follows:

PERIPHERAL TABLE (Cont.)

NUMBER PERIPHERAL LOCK

This byte contains the number of locks. This value is used by the dump analyzer to compute the sizes of the individual lists that comprise the locks. This value is the number of bytes in the SYSMEM flag and task byte lists and is one-fourth the size of the lock byte list.

SYSMEM FLAGS

These flags are used by OPEN SYSMEM LOCK as an indication that all directory activity for a unit being locked has ceased. When Open wants to OPEN SYSMEM LOCK, the SYSMEM flag is set (@FF@) for each lock that is not available (an available lock has its corresponding four-byte FILETAG entry in the lock bytes binary zeroed out). The Open delays until all of thes SYSMEM flags have been reset. It then continues with the Open.

TASK BYTES

The task bytes contain the internal task number of the job that is currently holding a lock. When a lock is freed, the lock maintenance code nsures that the task id of the lock holder matches the task id of the lock freer. If they do not match, a clearstart results.

LOCK BYTES

The lock bytes contain the FILETAGs for the files that have currently locked headers. When Open, Close, or Allocate attempt to unlock or lock a file header, the FILETAG is passed from the FIB.

ENTRY/DA_CT INDEX

The contents of this field is an index into the configuration table, DA_CT.

FORCE STATE STATUS

This field indicates whether the device is physically ready, physically not ready, or unavailable :

@00@ : Physically not ready
@01@ : Physically ready
@FF@ : Not available

STATUS

This field indicates the status of a device. The analyzer only lists this field for the devices that are marked physically not ready. The flag values are:

(MSB) 0 : Device post needed
1 : AVR needed
2 : Last status noticed
3 : Illegal PO flag

UNIT/PORT TABLE INDEX

This field is an index into the port table, as follows. The righthand digit is the port number and the lefthand digit is the device number within that port.

PERIPHERAL TABLE (CONT.)

QUEUE HEAD, TAIL, & CURRENT — These fields point to the queue of I/O's that are outstanding for this device. If the queue contains any entries, they are analyzed here.

QUEUE STATUS — This field reflects the status of this I/O.

@00@ : New
@01@ : In process - by the controller
@02@ : Complete
@03@ : Complete and dequeued (delinked from queue but still in buffer memory)

AM.ACTION.LIST INDEX — This field contains the action id of the OS action associated with this I/O.

RETURN STATUS — This is a three-byte field which is the I/O status returned for the last operation performed. (This field is only valid with a queue status of 2 or 3.) This field is decoded as follows:

MSB byte 1 : @00@ Controller error (internal to DP)

@01@ Command successful

@02@ Command unsuccessful

@03@ Device error, aborted command

@04@ Not used

@05@ Descriptor error

byte 2 : @00@ No error

@01@ Seek timeout

@02@ Head off cylinder

@03@ Sequence error (internal to DP)

@04@ Parity

@05@ Sector not found

@06@ Illegal address

@07@ Status word error(internal to DP)

PERIPHERAL TABLE (CONT.)

@08@ Data error (in compare)

@09@ Write inhibited

byte 3 : Retry count

OP CODE — This field contains the op code of the command the device was executing. The following is a list of acceptable I/O commands:

0	Read data	9,10	Search less than or equal
1	Write data	11,12	Search greater than or equal
3,4	Search less than	13	Initialize
5,6	Search greater than	14	Lock door
7,8	Search equal	15	Unlock door

During a search operation the 2 bit is used to condition read operations. If set, the hit sector will be read; if reset, it will not be read.

BUFFER ADDRESS — This is a 6-byte field. The first two bytes indicate size. The second two bytes indicate BUS/PAGE, and the last two bytes indicate offset.

SIZE PARAMETER — This is a 2-byte field indicating total buffer size.

DEVICE-DEPENDENT PARAMETERS

For Disk:

Byte 1-3 Sector address
4 Key length
5 Key displacement
6 Key record length
7-9 MTR status

For Tape Descriptor Format:

The tape I/O descriptor differs from the disk I/O descriptor format in the Return Status fields, the CPCODE, and the Device Dependent Parameters.

RETURN STATUS

Byte 1 - Command Status

@00@ - I/O Controller Error
@01@ - I/O Completed ok
@03@ - I/O Device Error

PERIPHERAL TABLE (CONT.)

@04@ - Tape Mark Read (end of file)
@05@ - I/O Descriptor Format Error
@06@ - End of Tape Sensed
@07@ - I/O Complete with a Short Block

Byte 2 - Error Status
@0C@ - No Error
@01@ - Not Ready Error
@02@ - Parity Error
@03@ - Tape Hard Error
@04@ - Attempt to Backspace over BOT
@05@ - Data Error
@06@ - Attempt to Backspace over Tape Mark
@07@ - Device Timeout
@08@ - Attempt to Write to Write Disable Tape
@09@ - Gap Timeout
@0A@ - Service Late Error

Byte 3 - Retry Count

OPCODES
@00@ - Read
@01@ - Write
@02@ - Write Tape Mark
@03@ - Skip to Tape Mark
@04@ - Read to Tape Mark
@05@ - Back Space One Record
@06@ - Skip One Record
@07@ - Rewind With Wait
@08@ - Rewind NC Wait
@09@ - Fixed Length Erase
@0A@ - Backspace TC Tape Mark
@0B@ - Stream Dump

Device Dependent Parameters
Bytes 0,1 - Amount of Bytes short if a Read Encounters a short block.

```
***************
VIRTUAL MEMORY
***************


***************
DA PROG NAMTAB
***************

                COPY ON BACKING STORE   PACKID     FILENAME       ENTRY-NUMBER
                **** ** ******* *****   *******    ********       ************
                       COPY IN MEMORY   PACKID     FILENAME       ENTRY-NUMBER
                       **** ** ******   *******    ********       ************
                                        0000000    SYS-SUPERUTL   00
```

Figure 7-1. DA_PROG_NAMTAB

```
*****************
DA INTERP NAMTAB
*****************

                COPY ON BACKING STORE   PACKID     FILENAME       ENTRY-NUMBER
                **** ** ******* *****   *******    ********       ************
                                        0000000    SYSMPLII       00
                                        0000000    SYSCOBOL       01
                       COPY IN MEMORY   PACKID     FILENAME       ENTRY-NUMBER
                       **** ** ******   *******    ********       ************
                                        0000000    SYSMPLII       00
                                        0000000    SYSCOBOL       01
```

Figure 7-2. DA_INTERP_NAMTAB

```
***************
DA MISC NAMTAB
***************

                COPY ON BACKING STORE   PACKID     FILENAME       ENTRY-NUMBER
                **** ** ******* *****   *******    ********       ************
                                        0000000    SYSLANGUAGE    00
                                        0000000    SYS-LOG-02     02
                                        0000000    SYS-LOG-03     03
                       COPY IN MEMORY   PACKID     FILENAME       ENTRY-NUMBER
                       **** ** ******   *******    ********       ************
                                        0000000    SYSLANGUAGE    00
                                        0000000    SYS-LOG-01     01
                                        0000000    SYS-LOG-02     02
                                        0000000    SYSDMFILE00    03
```

Figure 7-3. DA_MISC_NAMTAB

```
*********
MIX TABLE
*********

                COPY ON BACKING STORE
                **** ** ******* *****
                          MAX JOBS   FF
                NULL MIX RUNNING     FF
                          JOB COUNT  FF
                          MCS COUNT  FF
                     TEMPORARY ID    FF
                   NEXT MIX NUMBER   FF
                   COUNT WAIT JOBS   FF
                        RESTART ID   FF
              INTERPRETER ARRAY 0    FFFF
              INTERPRETER ARRAY 1    FFFF
                   SUPER FUNCTION

                   COPY IN MEMORY
                   **** ** ******
                          MAX JOBS   1A
                NULL MIX RUNNING     00
                          JOB COUNT  01
                          MCS COUNT  00
                     TEMPORARY ID    0C
                   NEXT MIX NUMBER   0D
                   COUNT WAIT JOBS   00
                        RESTART ID   0B
              INTERPRETER ARRAY 0    0301
              INTERPRETER ARRAY 1    0300
                   SUPER FUNCTION
```

Figure 7-4. Mix Table - Copy on Backing Store

```
MIX TABLE ENTRY / TASK ID  00
            PROGRAM FIB ID 0100          0000000/SYS-SUPERUTL
                VM FIB ID  0400
       INTERPRETER FIB ID  0300          0000000/SYSMPLII
               MIX NUMBER  01
                MIX FLAGS  0000
               MIX FLAGS2  00
                MIX STATE  04            EXECUTING
            MIX PROCESSOR  02
          MIX INTERPRETER  01
             MIX PRIORITY  C2            PRIORITY 8          CAN OPEN ANYTHING
             MIX TCB SIZE  7F02
       MIX PARENT POINTER  00
               REAL STORE  0000
                 SPO SIZE  0000
```

Figure 7-5. Mix Table - Copy in Memory

****
PORT-TABLE
****

ENTRY	INDEX	DEVICES	USER COUNT	ACTION TYPE	AM_ACTION_LIST INDEX
*****	*****	*******	*****	******	**********
0000	FF	00	00	00	FF
0001	00	06	00	03 DA DP CONTROL	03
0002	FE	00	00	00	FF
0003	FF	00	00	00	FF
0004	06	01	00	36 DA LP CNTRL	FF
0005	FF	00	00	00	FF
0006	FF	00	00	00	FF
0007	07	04	00	73 DA MT CNTRL	FF
0008	FF	00	00	00	FF
0009	FF	00	00	00	FF
000A	FF	00	00	00	FF
000B	FF	00	00	00	FF
000C	FF	00	00	00	FF
000D	FF	00	00	00	FF
000E	00	00	00	00	FF

Figure 7-6. Port Table

```
**********************
PERIPHERAL TABLE  PHT
**********************

              NUMBER PERIPHERAL LOCKS  0A
                        SYSMEM FLAGS  0000 0000 0000 0000 0000
                          TASK BYTES  0000 0000 0000 0000 0000
                          LOCK BYTES  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
                                      0000 0000 0000 0000

                   ENTRY/DA CT INDEX  00
                  FORCE STATE STATUS  00                  DEVICE PHYSICALLY NOT READY
                              STATUS  00
                UNIT/PORT TABLE INDEX  03
                          QUEUE HEAD  FFFF FFFF
                          QUEUE TAIL  FFFF FFFF
                       QUEUE CURRENT  FFFF FFFF

                   ENTRY/DA CT INDEX  01
                  FORCE STATE STATUS  00                  DEVICE PHYSICALLY NOT READY
                              STATUS  00
                UNIT/PORT TABLE INDEX  03
                          QUEUE HEAD  FFFF FFFF
                          QUEUE TAIL  FFFF FFFF
                       QUEUE CURRENT  FFFF FFFF

                   ENTRY/DA CT INDEX  02
                  FORCE STATE STATUS  00                  DEVICE PHYSICALLY NOT READY
                              STATUS  00
                UNIT/PORT TABLE INDEX  04
                          QUEUE HEAD  FFFF FFFF
                          QUEUE TAIL  FFFF FFFF
                       QUEUE CURRENT  FFFF FFFF

                   ENTRY/DA CT INDEX  03
                  FORCE STATE STATUS  00                  DEVICE PHYSICALLY NOT READY
                              STATUS  00
                UNIT/PORT TABLE INDEX  04
                          QUEUE HEAD  FFFF FFFF
                          QUEUE TAIL  FFFF FFFF
                       QUEUE CURRENT  FFFF FFFF

                   ENTRY/DA CT INDEX  04
                  FORCE STATE STATUS  01                  DEVICE PHYSICALLY READY
                              STATUS  04
                UNIT/PORT TABLE INDEX  05
                          QUEUE HEAD  0001 25A8
                          QUEUE TAIL  0001 25A8
                       QUEUE CURRENT  0001 25A8

                       QUEUE STATUS  01
                 AM_ACTION_LIST INDEX  18
                       RETURN STATUS  0100 00             COMMAND COMPLETE    NO ERROR
                        FORWARD LINK  FFFF FFFF
                       BACKWARD LINK  FFFF FFFF
                             OP CODE  01                  WRITE
                      BUFFER ADDRESS  F1FF 0001 0000
                      SIZE PARAMETER  F1FF
            DEVICE DEPENDENT PARAMETERS  0243 D202 0004 FFFF FF
```

Figure 7-7. Peripheral Table

SECTION 8
OPERATING SYSTEM

INTRODUCTION

The Operating System section of the dump examines the structures within the MCP. This portion of the dump file provides a history of what the MCP was doing before the dump was taken, and shows what section of the MCP (if any) detected an error.

The Operating System portion of the dump file is subdivided into the following sections:

ACTIVITIES
AM.ACTION.LIST
AM.QUEUE.HEAD
SEGMENT TABLE
HISTORY COMMUNICATE TABLE
PI DATA
DEVICE CONFIGURATION TABLE (DA_CT)
WB.LB.HEAD - Work Block/Link Block Head
FIB - File Information Block
AVAILABLE TABLE - Available Table Map
OVERLAYABLE MEMORY - Overlayable Memory Map
AVAILABLE MEMORIES - Available Memory Map
ODT DATA SEGMENT

INITIAL DUMP ANALYSIS

In an initial analysis of the OS section of the dump, these are the sections to be considered:

ACTIVITIES
AM.ACTION.LIST
HISTORY COMMUNICATE TABLE
PI DATA
DA_CT

The remaining sections are explained in the order in which they appear in the dump.

ACTIVITIES

This is the area in which the processor register values are saved when a failure is detected. If a clearstart occurs, the clearstart number appears in the ERROR NUMBER field. (Refer to Appendix A for an explanation of these error numbers.) This information gives a general idea of what caused the clearstart.

The fields ACTION ID and ACTION NAME contain the ID and the name of the action that is in error. If these fields contain @FF@ and the other register values are @00@, then the OS processor did not detect an error and therefore was not the reason for the dump to occur.

NOTE

If an action was marked "IN ERROR," it does not necessarily mean that this action had failed. It may mean, instead, that this was the action that detected the error. For example, this action may have picked up bad data from some other area of the MCP, or read an area of corrupted memory.

The MAX register is the next entry to be checked. This register is explained further in figure 8-1. It contains the bus address and the page of the processor that the OS processor was accessing at the time of the failure. If the OS processor detects a memory error, that error may be in any of the processors attached to the system. This is because any processor can access any other processor via the Processor Interface Bus.

The IC Status Word fields should be interrogated next. These words identify the status of the processor and the type of failure which occurred. Refer to the IC Memory Map, figure 8-2.

The IC FIRST ERROR REGISTER identifies the memory address within a page of memory that the processor was addressing at the time of the failure. In the case of a memory parity error, this register contains the address of the failing memory.

AM.ACTION.LIST

This area interprets the status blocks of all the actions that were active prior to the dump being taken. If the problem was a clearstart, the ACTION ID and the ACTION NAME found in the STATE SAVE AREA identifies the action in error. The status block of this action contains pertinent information. More specifically, the Caller Field within the status block contains the action number of the action that invoked it. This may help to identify the reason for the failure.

The LINK1 and LINK2 fields identify the sequence of events leading up to the clearstart.

The last field to interpret in the initial analysis of this section is the TASK ID/MIX TABLE INDEX. This field identifies the task in the MIX Table on whose behalf the action was running. A @20@ in this field indicates that the action was invoked by a system resource.

HISTORY COMMUNICATE TABLE

This table identifies the last 10 communicates performed. The table is cyclic, the highest entry number is the item which occurred most recently.

This information is useful as it indicates the type of I/O the system was performing before the clear-start occurred.

For example, if heavy disk I/O had been occurring, it is possible the failure is in the disk area.

PI DATA

In this section of the dump file, the PI DEAD PROCESSORS field indicates any "dead" processors on the system. The field is byte-reversed and bit-oriented.

For example, if a task processor on bus address 5 had been frozen for some reason, this field contains 1000. When this is byte-reversed it becomes 0010, the fifth bit being set. This indicates the processor on bus address 5 is frozen.

DA_CT

This table provides the logical status of the peripherals on the system. This information is used in conjunction with the Peripheral Handling Table to determine both the physical and logical status of the peripherals.

The other fields in the Operating System portion of the dump file are used for more in-depth analysis of errors. The method of using them is explained in detail in the remainder of this section.

Byte	Description	Length In Bytes	Note	Byte Reversed?
1	Clearstart/TP/DCP/OS Error No.	1	1	
2	AM ACTION LIST entry No.	1	1	
3	Action Name in error	1	1,2	
4-5	M1 Register	2		Yes
6-7	WR Register	2		Yes
8-9	M2 Register	2		Yes
10-11	B32 Register	2		Yes
12-13	MAX Register	2	3	Yes
14-15	MXA Register	2		Yes
16-17	MXB Register	2		Yes
18-19	B1FL Register	2		Yes
20	B0 Register	1		
21	REQ Register	1		
22-29	XY Register	8		Yes
30-35	J,K,L Registers	6		Yes
36	AD Register	1		
37-38	Top of stack (UMR3)	2		Yes
39-40	Top of X stack (UMRX3)	2		Yes
41-42	2nd on stack (UMR4)	2		Yes
43-44	2nd on X stack (UMRX4)	2		Yes
45-46	3rd on stack (UMR5)	2		Yes
47-48	3rd on X stack (UMRX5)	2		Yes
49-50	Bottom of stack (UMR6)	2		Yes
51-52	Bottom of X stack (UMRX6)	2		Yes
53	IC Error Status	1	3	
54	IC Status 1	1	3	
55	IC Status 2	1	3	
56	IC Status 3	1	3	
57	IC Status 4	1	3,4	
58	IC Status 5	1	3,4	
59	MA Lines Least Significant Byte	1	4,5	
60	MA Lines Most Significant Byte	1	4,5	
61	XMA Lines Least Significant Byte	1	4,5	
62	XMA Lines Most Significant Byte	1	4,5	

Notes:

1. Task Processor Error Numbers are defined in Appendix B.
   Clearstart Numbers are defined in Appendix A.
2. This field will contain a 00 or FF for Task Processor entries.
   The AM ACTION LIST names are listed in table 8-1.
3. See figure 8-2.
4. This field will contain FF on 1 MHz. processors.
5. This field is known as the FIRST ERROR Register.

Figure 8-1. Format of Processor State Save Area

NAME	PAGE	ADDRESS	MSB 7	6	5	4	3	2	1	0 LSB
IC STATUS 1	0	FFFB	PROC AVAIL	2ND ERROR	FREEZE	ROM / RAM	REQ 1	REQ 0	0	PG1 STATUS
ERROR STATUS	0	FFFC	2ND ERROR	ERROR	READ PARITY	WRITE PARITY	BOUND ERROR	MEM LIMIT	DATA / MKRD	WRITE / READ
IC STATUS 3	0	FFFD	LOCAL=1 / REMOTE	READ WITH CLEAR	ROM / RAM	IC ENABLE	BUS ADDRESS 8	BUS ADDRESS 4	BUS ADDRESS 2	BUS ADDRESS 1
IC STATUS 2	0	FFFE	POSS OS PROC	POSS DISK PROC	IOS AVAIL	CONTRL OF PERIPH	BUS ADDRESS OF THIS IC 8	BUS ADDRESS OF THIS IC 4	BUS ADDRESS OF THIS IC 2	BUS ADDRESS OF THIS IC 1
IC STATUS 4	1	FFF9	2ND ERROR	BUS CNTRL FREEZE	FRZ2 SET	MOM FRZ (NOT USED)	REFRESH FREEZE	I-O FREEZE	CLEAR FREEZE	CMND FREEZE
IC STATUS 5	1	FFFA	?	MTR PARITY	BUS REFRESH	MANDA-TORY REFRESH	OPPOR-TUNISTIC REFRESH	BUS DEST FREEZE	SYNC FREEZE	BUS REG FREEZE
MA LSB	1	FFFC	MA 7	MA 6	MA 5	MA 4	MA 3	MA 2	MA 1	MA 0
MA MSB	1	FFFD	MA 15	MA 14	MA 13	MA 12	MA 11	MA 10	MA 9	MA 8
XMA LSB	1	FFFE	0	0	0	0	XMA 3	XMA 2	XMA 1	XMA 0
XMA MSB	1	FFFF	XMA 15	XMA 14	XMA 13	0	XMA 11	XMA 10	XMA 9	XMA 8

MAX

ROM	RD WITH CLEAR	IC EN	0	BUS ADDRESS 8	BUS ADDRESS 4	BUS ADDRESS 2	BUS ADDRESS 1					PAGE SELECT 8	PAGE SELECT 4	PAGE SELECT 2	PAGE SELECT 1

UMARX

0	0	IC EN	0	0	0	0	0	0	0	0	0	PAGE SELECT 8	PAGE SELECT 4	PAGE SELECT 2	PAGE SELECT 1

ED2713

Figure 8-2. IC Memory Map

ACTIVITIES

STATE SAVE AREA

For each processor on a B 900/B 5900 system, there is an area in the dump known as the State Save Area. This area is used to store the contents of the NPRO registers and IC registers at the time a failure is detected within that processor. For initial dump analysis, this area is one of the most important areas for the purpose of identifying the cause of a failure. The important fields in this area are listed in the introduction to this section.

If no error is detected by a specific processor, the contents of these registers are: OS, DCP, and Disk Processor: the first three bytes are @FF@, the register values are 0. Task Processor: the State Save Area address is 0, the area is not printed.

The format of the State Save Area is detailed in figure 8-1. An example of the State Save Area is presented in figure 8-3. This information is also contained in the maintenance log file. Refer to B 900/ B 5900 F. E. Handbook, form 1127297, for the format of maintenance entries.

AM.ACTION.LIST

This structure is a list of status blocks associated with actions awaiting processor time or an external event. The information contained in a status block is as follows:

ENTRY NUMBER	This information is formulated by the system dump analyzer. It is the index into the action list. This index into the action list is the action id.
STATUS BLOCK ADDRESS	This information is formulated by the system dump analyzer. It is the OS memory address of the status block.
COMM AREA	This is the area which is used to communicate information to other actions.
TASK ID MIX TABLE INDEX	This field contains the task id of the job on whose behalf this action is executing. @20@ denotes that the action is on behalf of a system function, not a task function.
PARAMS	This field contains the parameters which were passed to activity management by the action. If the status of this action is "WAITING REPLY," the second byte of this field will contain the Action ID of the action invoked by this action.
TIME	This field contains the time of day at which the action was started (hour, minute, second, hundredths).
NAME	This field contains the name of the action which owns the status block. Table 8-1 is a complete

```
********************************
****** OPERATING SYSTEM ******
********************************


**********
ACTIVITIES
**********


                STATE SAVE AREA
      STATE SAVE AREA ADDRESS  C724
                 ERROR NUMBER  FF
                    ACTION ID  FF
                  ACTION NAME  FF                    END OF LIST EXCEEDED
                           M1  0000
                           WR  0000
                           M2  0000
                          B32  0000
                          MAX  0000
                          MXA  0000
                          MXB  0000
                         BTFL  0000
                           BO  00
                          REG  00
                           XY  0000 0000 0000 0000
                            J  0000
                            K  0000
                            L  0000
                           AD  00
                  TOP OF STACK  0000
                 TOP OF XSTACK  0000
                  2ND ON STACK  0000
                 2ND ON XSTACK  0000
                  3RD ON STACK  0000
                 3RD ON XSTACK  0000
               BOTTOM OF STACK  0000
              BOTTOM OF XSTACK  0000
            IC - ERROR STATUS   00
          IC - STATUS WORD 1    00
          IC - STATUS WORD 2    00
          IC - STATUS WORD 3    00
          IC - STATUS WORD 4    FF
          IC - STATUS WORD 5    FF
       IC - FIRST ERROR REG     FFFF FFFF
```

Figure 8-3. Sample of Beginning of Operating System Portion of a Dump

AM.ACTION.LIST ENTRIES (Cont.)

list of possible values and the action they denote. The analyzer prints the appropriate id.

CALLER — This field contains the action id (index into AM.ACTION.LIST) of the action that invoked this action. @FF@ = invoked without wait, @00@ = TP invoked.

SB_STACK_OFFSET — This field contains the offset of the current stackframe (see glossary).

This is a pointer from beginning of "comm area" to a location in the stackframe. This may either be as an offset, or a memory address. This may be determined by the value, since stackframes are required to be kept small. Typically, if the value is displayed as nn00 (byte reversed = 00nn) this would be an offset. If an action is in use, or recently used, one would see an address. Conventionally, on yield this value would be returned to an offset.

LINK1 — This field contains the scheduling link. That is, the action id (index into AM.ACTION.LIST) of the next action on the scheduling list.

LINK2 — This field contains the suspension link.

STATUS — This field contains the status of the action. The possible contents of this field are shown below. The field is also interpreted by SYSANALYZER.

Value	Status	Value	Status
@00@	Available		
@01@	Waiting reply (waiting for an invoked action)	@0B@	Waiting Conditional Post (waiting for a task processor to finish)
@02@	Waiting critical action	@0C@	Lonely wait (waiting to become the only action running; used with "GT MD".)
@03@	Waiting VM action		
@04@	Waiting VM reply		
@05@	Waiting post (waiting for an action in OS to post a reply)	@0D@	Event wait
@06@	Long delayed	@0E@	Waiting mailbox
@07@	Short delayed	@0F@	Waiting processor (waiting for an interrupt from a task processor)
@08@	Waiting interrupt	@10@	Ready
@09@	$$$ IN ERROR $$$	X@10@	Invalid

Table 8-1. Status Block Action Name Values

Value	Action	Value	Action	Value	Action
01	AM_VM	1A	AM_UNIMPLCOMM	33	DA_DUMPTOVM
02	AM_RTC	1B	DA_RD	34	DA_SYSPRINT
03	DA_DP_CONTROL	1C	DA_AD	35	DA_FILESEND
04	DA_OSVMIO	1D	DA_CL	36	DA_LPCNTL
05	DA_ERRLOGGING	1E	UNIMP_SCL	37	DA_M_DK_CNTL
06	MN_LOG_INIT	1F	DA_OL	38	MN_TRAP_INIT
07	MN_PROC_POLL	20	DA_PG	39	MN_TRACE_INIT
08	DC_RSLT_FN	21	DA_PO	3A	MN_TRACE_STOP
09	DC_LOAD	22	DA_RY	3B	MN_PROC_STATS
0A	DC_EOJ	23	DA_SF	3C	MN_SWITCH
0B	JM_RESTART	24	DA_SN	3D	MN_MEM_DUMP
0C	AM_SYSTEM_ERR	25	DA_SV	3E	DC_VERB30
0D	MN_QUICK_LOG	26	DA_RTN_DEVICE	3F	DC_VERB31
0E	FUTURE USE	27	DA_CMS_START	40	DC_VERB32
0F	PI_SEND	28	DA_CMS_RWRITE	41	DC_VERB33
10	PI_RECEIVE	29	DA_CMS_READ	42	DC_VERB34
11	MN_LOG_IR	2A	DA_CMS_WRITE	43	DC_SPO
12	JM_EX	2B	DA_CMS_DELETE	44	JM_MIXACCESS
13	OI_ACCEPT	2C	DA_CMS_STRMIO	45	OI_SCL_LOGON
14	OI_ZIPDISPLAY	2D	DA_CMS_OPEN	46	DA_NAME_FIBID
15	OI_SYSMESSAGE	2E	DA_CMS_CLOSE	47	JM_JOB_RUNNING
16	OI_AX	2F	DA_SYSOPEN	48	JM_JOB_REMVED
17	OI_REQUESTS	30	DA_SYSIO	49	JM_GO
18	OI_ROUTING	31	DA_SYSCLOSE	4A	JM_PR
19	OI_SYSSTATUS	32	DA_ADDAREA	4B	JM_STOP

Table 8-1. Status Block Action Name Values (Cont.)

Value	Action	Value	Action	Value	Action
4C	JM_DS_DP	5E	JM_SUP_ACCEPT	70	DA_RUF_GEN
4D	JM_THRASHING	5F	JM_EOJ	71	DA_DEV_STATS
4E	JM_DC_JOBREM	60	JM_MIX	72	DC_DCP_ERROR
4F	DC_VERB3F	61	OI_DCI_INT	73	DA_MT_CNTRL
50	AM_WAIT	62	OI_ODT_DCI	74	DA_MT_AVR
51	DC_JOB_LOG	63	DA_GET_DC_MEM	75	DA_CT_CNTRL
52	JM_TP_ERROR	64	DA_SO	76	DA_CT_AVR
53	AM_TIMER	65	DA_TO	77	DA_CHNG_REEL
54	DA_CMS_TST_ST	66	PI_DEAD_PROC	78	JM_SPO_SIZE
55	JM_PAUSE	67	OI_SCL_READQ	79	DA_INIT_TRK
56	AM_DT	68	OI_SCL_RUN	7A	AM_CMPLX_WAIT
57	AM_DATE_TIME	69	OI_SCL_LOGOFF	7B	DC_TSTEVNT
58	DA_RES_AVR	6A	DA_REDEF_READ	7C	OI_TEST_SCL_Q
59	AM_CMS_ERROR	6B	DA_RL	88	PI_REVIVE
5A	DA_LP_AVR	6C	DA_AP	89	DA_IO_LINKER
5B	DA_DK_AVR	6D	MN_LOG_LG_LS	8A	MN_HOOKUP
5C	DA_ICMD_AVR	6E	DA_VF	8B	MN_DUMMY
5D	DA_HSP_CNTRL	6F	DA_MERGE		

AM.ACTION.LIST ENTRIES (Cont.)

PROC.WB	This field contains the associated processor bus address number in the high order digit. 0 means it was invoked at WARMSTART. The number of assigned work blocks in the low order digit uses bit pattern 1 bit per WB, not binary count.
EVENTS	Control information pertaining to the action associated with the status block. This field is used by activity management. Valid when the action is waiting on an event or when the action caused an event. If this field contains @FF@, the event has occured; if this field contains @00@, the event has not yet occured; if this action caused an event, then this field contains the event number.
EVENT TASK	This field contains the task id associated with the event described in the previous field; but it is only used if action enabled event.
ATTACHED OWNER	The owner of the attached data structure. If the attached type is a work block or a data block, this is an action ID. If the attached type is a link block, this is an action list name.
ATTACHED ID	The ID of the attached data structure.
ATTACHED TYPE	The type of the attached data structure. This is either a work block (00), a link block (01), or a data block (05).
ATTACHED BASE	The memory address of the attached data structure.
SB SPO ID	Reserved for future use.
SB SPO TAG	Reserved for future use.
VMREQ	This field contains the last virtual memory request. Only valid when waiting on VM.
PI REQUEST FUNCTION, PI REQUEST PROCESSOR, PI RESPONSE PROCESSOR, PI RESPONSE FUNCTION, PI MIX NUMBER	Original processor request links. More detail will be found in 10, Task Processor.
PI RESULT, PI FLAGS	These are the result areas for interprocessor communications. More information can be found in section 10, Task Processor.

AM.ACTION.LIST ENTRIES (Cont.)

STACK ENTRIES	These fields are duplicated for additional stackframes and if the outer level declares no local variables. A sample status block entry is shown in figure 8-4.
SEGMENT NO	Field containing the segment number within the MCP file. Refer to the Segment Table.
SEGMENT OFFSET	Contains the return address (offset within the segment).
LAST.L	Contains the location of the calling segment number in the stack frame. (Not valid for outer level.)
NUMBER OF PARAMS	Contains the number of parameters in use by the stack frame. (Not valid for outer level.)
ADDR OF PARAMS	Contains the address of the parameters in use.
LOCAL VARIABLES	Local variables used by the action. The number and type are determined by the procedure in use.

AM.QUEUE.HEAD

This structure is an array of queue heads for system resources. The array is decoded by the system dump analyzer by queue name and queue contents. The queue contents are shown as a list of action IDs (index into the AM.ACTION.LIST). An empty queue is shown as having a single entry of @FF@.

AM.LAST.QUEUE	This field gives the memory address of the head of the last queue to be serviced. Currently (3.04) this address is not interpreted.
AM.WAITING.HEAD	This field is a pointer to a list of actions waiting to start. The analyzer lists the action ids of all actions currently in the waiting list. A single entry of @FF@ in the AM.WAITING.HEAD field signifies that the list of waiting actions is empty.
AM.NORMAL.HEAD	This field is a pointer to an action which is the top of a circular list. The analyzer lists the whole circle which is a set of action ids.
AM.LONG.DELAY	This field is a pointer to actions which have been delayed for seconds. The analyzer lists all such actions. A single entry of @FF@ implies an empty list.
AM.SHORT.DELAY	This is a pointer to a list of actions which have been delayed for milliseconds. The analyzer lists the ids of all such actions. A single entry of @FF@ implies an empty list.

```
**************
AM_ACTION_LIST
**************

                ENTRY-NUMBER  00
          STATUS BLOCK ADDRESS  F417
                        COMM  0301 0000 0269 c40u 0000 0000 0000 0902
TASK ID / MIX TABLE INDEX     20
                      PARAMS  0000 00
                        TIME  1411 5000
                        NAME  88                    MN DUMMY
                      CALLER  FF
           SB_STACK_OFFSET    2718
                       LINK1  FF
                       LINK2  FF
                      STATUS  10                    READY
                    PROC_WB   20
                      EVENTS  00
                 EVENT TASK   00
             ATTACHED OWNER   00
                ATTACHED ID   FF
             ATTACHED TYPE    00
             ATTACHED BASE    0000
                SB SPO ID     00
               SB SPO TAG     0000
                      VMREQ   0000
     PI REQUEST FUNCTION      88
   PI REQUEST PROCESSOR       01
  PI RESPONSE PROCESSOR       02
   PI RESPONSE FUNCTION       FF
          PI MIX NUMBER       20
              PI RESULT       00
               PI FLAGS       00
                               STACK ENTRIES
                               ***** *******
                SEGMENT NO     00
            SEGMENT OFFSET     216E
                      LAST_L   0625
          NUMBER OF PARAMS     00
            ADDR OF PARAMS
           LOCAL VARIABLES     0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
                               0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
                               0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
                               0000 0000
```

Figure 8-4. Status Block Entry Example

SEGMENT TABLE

This structure is an array of code and data block descriptors. The analyzer decodes each descriptor of this table for the user. The descriptors indicate the segment's ID, frozen/jelled count, size, disk address, present/access/type, memory address, and page.

HISTORY COMMUNICATE TABLE

This table is a trace of the last ten communicates received by the OS. Each table entry comprises the following fields:

ENTRY	This field contains a number incremented by 1. Its purpose is to show the order in which the communicates were received.
TASK ID	This field is an index into the mix table. Using this information along with the mix table will reveal which job issued the communicate.
BUS ADDR	This field indicates the processor which was running the task that issued the communicate.
VERB	This parameter is held in byte 0 of the CPA and is used to indicate which of the possible functions is required. It may take values in the range 0 - 255. For full details see the CMS MCP MANUAL. The current list of CMS communicates is given below:

01 : OPEN
02 : CLOSE
11 : ZIP
12 : DISPLAY
13 : ZIP & DISPLAY
14 : PAUSE
15 : ZIP & PAUSE
16 : DISPLAY & PAUSE
17 : ZIP, PAUSE & DISPLAY
1A : CONDITIONAL DISPLAY
1B : ZIP & CONDITIONAL DISPLAY
1C : DISPLAY WITHOUT LOGGING
20 : ACCEPT
21 : SUPER.ACCEPT
30 : MCS CONTROL COMMUNICATE
31 : MCS INTERROGATES
32 : MCS REDEFINITION
33 : USER DATA COMM
34 : MCS DCP ORIENTED COMMUNICATE

40 : DATE-TIME
41 : TERMINATE
42 : WAIT
43 : SYSTEM STATUS
50-6F : RESERVED FOR EXPANSION
70-7F : RESERVED FOR SYSTEM UTILITY FUNCTIONS
80 : TEST STATUS
82 : READ (NOT CONSOLE)
84 : WRITE (NOT CONSOLE)
86 : REWRITE
88 : DELETE
8A : STREAM CONTROL
8C : START
8E : OVERWRITE
90 : READ-WRITE
92 : READ (CONSOLE)
94 : WRITE (CONSOLE)
96 : GET
98 : PUT

OBJECT	This parameter is held in byte one of the CPA. Most commonly, it is an index into the task's data segment table giving access to data which characterizes the entity upon which the function is to be performed. For file type I/O communicates, for instance, it is the DST index for the associated file information block.
ADVERB	This parameter, if it exists, consists of a variable number of bytes starting at byte 2, and comprising the remainder of the CPA.

Refer to figure 8-5 for an example of the History Communicate Table.

PI DATA

Refer to figure 8-6 for an example of the PI Data Area.

PI MAILBOX PAGES	This structure is an array of constants which is a list of processor numbers. These are listed to be used as labels for the following arrays, active input mailboxes, active output mailboxes and waiting mailbox ids.
PI ACTIVE INPUT MAILBOXES	This array is used by task processors to indicate that their output mailbox contains a message for the OS. A request is indicated by an @80@ and a response is indicated by a @00@. @FF@ is used to denote an idle mailbox. The array is ordered underneath the mailbox pages field.
PI ACTIVE OUTPUT MAILBOXES	This array indicates the outstanding mailbox ACKs. An acknowledgement is indicated by a 00. The processor related to the acknowledgement can be found in the equivalent position in the active input array above. Again @FF@ is used to denote the idle state.
PI DEAD PROCESSORS	This field indicates which processors have gone not ready. Each bit represents a bus address (1 relative). NOTE This field is byte-reversed.
PI WAITING MAILBOX IDS	This array indicates actions which are waiting for a mailbox acknowledgement. Each byte of this array is an action id (index into the AM_ACTION_LIST). The index into this table (0 relative) is the processor number associated with the action.

```
*************************
HISTORY COMMUNICATE TABLE
*************************
```

ENTRY	TASK ID	BUS ADDR	VERB	OBJECT	ADVERB
*****	****	****	****	******	******
13	0C	04	02 CLOSE	22	1010F003000000000000000000000
14	0C	04	02 CLOSE	21	1010F003000000000000000000000
15	0C	04	41 TERMINATE	80	0000100300000000000000000000
0C	0C	04	84 WRITE	21	1C1090030000000000000000000
0D	0C	04	82 READ	22	00005D730000000000000000000
0E	0C	04	82 READ	22	00109F030000000000000000000
0F	0C	04	84 WRITE	21	10109F030000000000000000000
10	0C	04	82 READ	22	00005E730000000000000000000
11	0C	04	82 READ	22	0010F003000000000000000000
12	0C	04	84 WRITE	21	1010F003000000000000000000

Figure 8-5. Example of the History Communicate Table Portion of a Dump

```
*******
PI DATA
*******
```

```
                PI MAILBOX PAGES  0102 0304 0506 0708 090A 0B0C 0D0E 0FFF
     PI ACTIVE INPUT MAILBOXES    FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
    PI ACTIVE OUTPUT MAILBOXES    FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
           PI DEAD PROCESSORS     0000
       PI WAITING MAILBOX IDS     FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
            PI FREE MAILBOXES     FFFF FFFF FFFF FFFF FFFF FFFF FFFF FF
              PI QUEUE COUNT      03
                  PI INVOKED      00
```

Figure 8-6. PI Data Example

PI DATA (Cont.)

PI FREE MAILBOXES This is an array of flags indicating if the mailbox for a processor is available. (@FF@ indicates available.)

DA_CT (CONFIGURATION TABLE)

This table contains one entry per device. This information is used by data access when performing opens, OLs and other SCL commands. Each entry of the table contains the following fields:

ENTRY/PHT INDEX	This field is an index into the peripheral table.
CMS DEVICE TYPE	This field contains the device type, such as BSM cartridge. The analyzer interprets this field into both SPO mnemonic and English name.
TASK ID	This field indicates the task id (index into the AM.ACTION.LIST). @FF@ in this field means that no action is currently associated with this entry; it is only valid during job load, when it contains task id of job being loaded.
FILE OPEN	This field contains the number of files open on the device.
STATUS	This field indicates the logical status of the device. This is a 2-byte field, the second of which is not used. The following is a decode of this field as shown by the system dump analyzer. MSB 7 - Reserved 6 - Multifile 5 - Purged 4 - Unlabelled 3 - System 2 - SV or PO pending 1 - Assigned LSB 0 - Ready
SPO MNEMONIC	This field contains the associated mnemonic for this device (DMA, DKB, etc.).
FILE ID	For a disk device, this field contains the ⟨pack id⟩. For a line printer, it is ⟨fileid⟩ if a file is currently open on the printer.
SERIAL NUMBER thru LOG UNIT	These fields contain label information found on the device.

Refer to figure 8-7 for an example of a DA_CT entry.

```
**********************************
DEVICE CONFIGURATION TABLE  DA-CT
**********************************

          ENTRY/PHT INDEX  0000
          CMS DEVICE TYPE  CE          DM-3/6 MINI
                  TASK ID  FF
                FILE OPEN  0000
                   STATUS  4000        MULTI-FILE
              SPO MNEMONIC  DMA
                  FILE ID
          ALLOCATION UNIT   FF
        PACK TAG/LOG UNIT   20
            SECTORS/TRACK   3B

          ENTRY/PHT INDEX  0001
          CMS DEVICE TYPE  CE          DM-3/6 MINI
                  TASK ID  FF
                FILE OPEN  0000
                   STATUS  4000        MULTI-FILE
              SPO MNEMONIC  DMB
                  FILE ID
          ALLOCATION UNIT   FF
        PACK TAG/LOG UNIT   20
            SECTORS/TRACK   3B

          ENTRY/PHT INDEX  0002
          CMS DEVICE TYPE  CB          DK-CARTRIDGE
                  TASK ID  FF
                FILE OPEN  0000
                   STATUS  4000        MULTI-FILE
              SPO MNEMONIC  DKC
                  FILE ID
            SERIAL NUMBER   123456
          ALLOCATION UNIT   01
        PACK TAG/LOG UNIT   20
            SECTORS/TRACK   20
```

Figure 8-7. Sample DA_CT Dump File Entry

WB.LB.HEAD

This field is a pointer to a linked list of work and link blocks currently allocated. The analyzer decodes each entry on the list and breaks out the descriptor address and type, and the size, link, type, address and owner of the work or link block.

FIB

OWNER	The owner from the WB.LB.HEAD of DA_TASKnn_LIST using this FIB. DA_TASK_LIST will indicate the user task.
ID	Segment number of the FIB.
FILE STATE	The values in this field reflect the open/close state of the file as follows: @00@ = half closed - new file @01@ = half closed - old file @02@ = open - new file @03@ = open - old file @04@ = being opened @05@ = half closed - printer backup file
COMM IN PROGRESS	A flag indicating if a communicate is currently in progress using this FIB (Flag = 1 indicates in progress)
FILE TAG INFO	MSB 7 : Extended file 6 : Overflow file (exists on more than one pack) 5-0 : Configuration table offset
FILE TAG DIRECTORY	For multiple-pack files, this field contains a pointer to the disk file header (DFH); this DFH will point to the next pack.
FILE TAG PACKTAG	Pseudo pack tag appended to the original file name by pseudo pack implementation.
FILE USAGE	This is a bit map. MSB 7-5 : value 001 = normal/shared 111 = locked 4 : value 001 = shared 3 : value 001 = locked 2-0 : value 001 = normal/shared user 111 = locked user
FT ADDRESS PAGE	Memory page on which the associated file resides.
FT ADDRESS BUS	Memory address of the associated file table.

Other fields are considered self-explanatory. Refer to figure 8-8 for an example of the FIB entry.

```
***
FIB
***

                                    OWNER  05
                                       ID  12
                               FILE STATE  03                  OPEN OLD
                          COMM IN PROGRESS  00
                             FILE TAG INFO  84
                          FILE TAG DIRECTORY  7500
                            FILE TAG PACKTAG  20
                                 FILE USAGE  20
                              ORGANIZATION  02                  SEQUENTIAL
                                    ACCESS  01
                                    MYUSE  01
                               CMS DEVICE TYPE  C3              DK-ANY DISK
                          FPB SEGMENT NUMBER  0B
                                  FPB SIZE  5E00
                                 OTHERUSE  03
                            LAST COMMUNICATE  82
                               RECORDS/BLOCK  0100
                                 RECORD SIZE  8400
                           WORK AREA SEGMENT  00
                            WORK AREA OFFSET  4605
                            WORK AREA LENGTH  8400
                                 MAX WRITTEN  0001 A8
                                MAX POSSIBLE  0001 A8
                              CURRENT RECORD  0000 81
                            CUR REC INVALID  00
                              CURRENT BLOCK  0000 80
                               SECTORS/BLOCK  0100
                                 BLOCK COUNT  0000
                          BACKUP RECORD BUFFER  0000 00
                            MAX NUMBER BUFFERS  01
                        NUMBER BUFFERS ALLOCATED  01
                                 BYTES/BUFFER  8400
                          OFFSET WITHIN RECORD  0000
                           RECORD WITHIN BLOCK  0000
                                DYNAMIC ACCESS  00
                                      EOF FLAG  00
                                .OUTPUT ALLOWED  FF
                              SPARE CHARACTERS  0000
                             TRANSLATION NEEDED  00
                                      MS OPEN  00
                                   ORG DEVICE  39
                                        FLAGS  00
                                 PB FILE NUMBER  39
                              BUFFER AHEAD NUMBER  FFFF FF
                                      FILE ID  5359 534C 414E 4755 4147 4520
                                PB LP DEVICE TYPE  43
                                  FT ADDRESS PAGE  0007
                                   FT ADDRESS BUS  7DA5
                  KEY USAGE  31
                KEY FILE TAG  8390 0021
             KEY FILE BUFFER  3030 3034 3700 01CF 3030 3034 3700 01D0 3030 3034 3700 01D1 3030 3034 3700 01D2
                              3030 3034 3700 01D3 3030 3034 3700 01D4 3030 3034 3700 01D5 3030 3034 3700 01D6
                              3030 3034 3800 01D7 3030 3034 3800 01D8 3030 3034 3800 01D9 3030 3034 3800 01DA
                              3030 3034 3800 01DB 3030 3034 3800 01DC 3030 3034 3800 01DD 3030 3034 3800 01DE
                              3030 3034 3800 01DF 3030 3034 3800 01E0 3030 3034 3900 01E1 3030 3034 3900 01E2
                              3030 3034 3900 01E3 3030 3034 3900 01E4 1307 0057
                     UPDATED  00
       KEY FILE TABLE ADDRESS  0201 F0D3
     KEY FILE TABLE ADDRESS 2  0201 20D3
          DUPLICATES ALLOWED  FF
                 READ STATUS  01
                   KEY OFFSET  0000
              NULL KEY FILE  00
       ROUGH TABLE IN MEMORY  FF
              PRESENT SECTOR  005C 64
                 LAST ACCESS  00
             CURRENT AREA 1  00
           CURRENT SECTOR 1  005C 64
           CURRENT OFFSET 1  A0
              CURRENT RRN 1  0001 E3
             CURRENT AREA 2  00
           CURRENT SECTOR 2  005C 66
           CURRENT OFFSET 2  00
              CURRENT RRN 2  0000 00
                CURRENT KEY  3030 3034 39FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                              FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                  ENTRY SIZE  05
           ROUGH TABLE SIZE  0000 02

                              AREA NUMBER     SECTOR OFFSET  KEY ENTRY
                              **** ******     ****** ******  *** *****
                              00              0300           3030303530
                              00              0000           FFFFFFFFFF
```

Figure 8-8. File Information Block Example

ODT DATA SEGMENT

The fields of this portion of the formatted dump file are explained in the following list. A sample of this section is shown in figure 8-9.

SYSCONFIG ZIP TEXT — Twenty bytes which contain ZIP text taken from the SYSCONFIG file.

REMOTE TASK ID — One byte which contains the task ID of the program logged on as remote SPO.

CONTROLLING SPO ID — Not used.

CONTROLLING SPO TAG — Not used.

SPO ACTIVE ARRAY — This is a 2-byte field; each bit indicates if the associated task is a REMOTE SPO operation.

ODT FLAGS — This is a 1-byte field. The significance of its bits is as follows:

Bit 7 (MSB)	ODT in shutdown mode
Bit 6	Remote SPO is allowed
Bit 5	Processing PO message
Bit 4	OI routine in shutdown
Bits 2 & 3	Not used
Bit 1	A remote SPO exists
Bit 0	Local SPO in use

ACTIVITY FLAGS — This is a one-byte field. The significance of each bit is as follows:

Bit 7 (MSB)	OI START complete
Bit 6	OI SYSMESSAGE down
Bit 5	Not used
Bits 3 & 4	00 - No local SPO exists 01 - Local SPO exists
Bit 2	SYSIO buffer WB exists
Bit 1	Dictionary is open by OI
Bit 0	SYSCONFIG is open by OI

DICTIONARY FIB ID — This is the FIB ID of the dictionary file.

SYSIO WB ID — This is the ID of the work block being used for SYSIO.

ROUTING LIST HEAD — This is the ID of the link block entry at the head of the routing list.

ROUTING LIST TAIL — This is the ID of the link block entry at the tail of the routing list.

SPO QUEUE HEAD — This is the ID of the link block entry at the head of the SPO queue. This is a holding area for messages.

```
****************
ODT DATA SEGMENT
****************

    SYSCONFIG ZIP TEXT  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
        REMOTE TASK ID  FF
   CONTROLLING SPO ID   FF
  CONTROLLING SPO TAG   FF
     SPO ACTIVE ARRAY   0000
            ODT FLAGS   41
                                            REMOTE SPO ALLOWED
                                            LOCAL SPO IN USE
        ACTIVITY FLAGS  8A
                                            OI START COMPLETE
                                            LOCAL SPO EXISTS
                                            DICTIONARY IS OPEN BY OI
    DICTIONARY FIB ID   0200
          SYSIO VB ID   01
    ROUTING LIST HEAD   98
    ROUTING LIST TAIL   98
        ODT LIST HEAD   90
        ODT LIST TAIL   98
       SPO QUEUE HEAD   00
     SCL QUEUES HEADS   0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
       SPO QUEUE TAIL   00
     SCL QUEUES TAILS   0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
      ROUTING POST ID   16
          ODT POST ID   FF
       SPO CHANNEL ID   03
       LONG MSG INDEX   0700
   ADDRESS OF MSG OUT   55C5
         MSG OUT DATA   0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
               MSG IN   4130 0244 5320 3033 2F44 554D 4D59 2E32 3130 3403 FFFF FFFF FFFF FFFF FFFF FFFF
                        FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                        FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                        FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                        FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                        FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                        FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                        FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                        FFFF FF
  CONSOLIDATED STATUS   40
                                            RECEIVE REQUEST

      TRANSMIT STATUS   40
       RECEIVE STATUS   00
            DCI FLAGS   02
                                            TRANSMIT UNDERWAY
           PROC FLAGS   00
         RECEIVE CHAR   15
      DCI TRANSMIT ID   19
      CHARACTER COUNT   00
          TIMER VALUE   0007
         MESSAGE TYPE   00
       MSG OUT LENGTH   1C00
        MSG OUT INDEX   0700
         MSG IN INDEX   0300
        MSG IN LENGTH   1000
         MSG IN WB ID   FF
        SEND PROGRESS   00
    TERMINAL SEQUENCE   1411 0301
      HEADER SEQUENCE   0141 3002
    LINEFEED SEQUENCE   1822 2024 184D 1822 2037 2020 20
       ERROR SEQUENCE   0141 3002 1822 2F23 1848 1822 6823 3C45 5252 3E
       CLEAR & SCROLL   0141 3002 1822 3020 1848 1822 2021 1848 1822 2022 1848 1822 2023 1848 1822 2023
                        1853 1848 1853 1848 1853 1848 1853 1848
```

Figure 8-9. Sample ODT Data Segment

SPO QUEUE TAIL	This is the ID of the link block entry at the tail of the SPO queue.
SCL QUEUES TAILS	Not used.
ROUTING POST ID	This is the ID of the status block that is handling message routing.
ODT POST ID	This is the ID of the status block that is handling ODT output messages. It is used by the routing action.
SPO CHANNEL ID	This is the physical channel for the ODT device.
LONG MSG INDEX	This is the index into output messages that are longer than one screen.
ADDRESS OF MESSAGE OUT	This is a 2-byte address of the message being sent to the ODT.
MSG IN	Input message from the ODT.
CONSOLIDATED STATUS	This is a 1-byte field. The significance of the bits is: Bit 7 (MSB) Transmit request Bit 6 Receive request Bit 5 Timer A request Bit 4 Timer B request Bit 1,2,3 Not used Bit 0 Receive error
TRANSMIT STATUS	ODT DCI status for Transmit state.
RECEIVE STATUS	ODT DCI hard status for Receive state.
DCI FLAGS	This is a 1-byte field. The significance of the bits is: Bit 7 (MSB) DCI is finished Bits 4,5,6 Not used Bit 3 Transmit inside receive Bit 2 Transmit desired Bit 1 Transmit underway Bit 0 Receive underway
PROC FLAGS	This is a 1-byte field. The significance of the bits is: Bit 7 (MSB) An error was logged Bits 5,6 Not used Bit 4 Explicit line feed Bits 1,2,3 Not used Bit 0 Error in message

RECEIVE CHAR	This is the last character received from the ODT.
DCI TRANSMIT ID	This is the ID of the status block that is handling ODT output messges. This information is used by ODT DCI interrupt action.
CHARACTER COUNT	This is the count of the number of characters received from the ODT.
TIMER VALUE	This is a value used to prime a hard timer in the ODT DCI.
MESSAGE TYPE	This is a value used to indicate various internal message types; for example system messages or ODT control messages.
MSG OUT Length	This is the length of the ODT output message in bytes.
MSG OUT INDEX	Index to the current character being output to the ODT.
MSG IN INDEX	Index to the current character being input from the ODT.
MSG IN Length	This is the length of the ODT message in bytes.
MSG IN WB ID	This is the ID of the work block where the incoming ODT message is being placed.
SEND PROGRESS	This is an internal value used to step through the transmission sequence.
TERMINAL SEQUENCE	
HEADER SEQUENCE	
LINEFEED SEQUENCE	Each of these sequences is a series of control characters used by the ODT controller to manipulate the ODT screen under various circumstances.
ERROR SEQUENCE	
CLEAR & SCROLL	

SECTION 9
DATA COMMUNICATIONS

INTRODUCTION

The data communications portion of a system dump is divided into three sections. Each section represents one of the major parts of the B 900/CP 9500 Data Comm (DC) Subsystem. These sections are:

DCA MEMORY (Data Comm Activity Memory)

DCP MEMORY (Data Comm Processor Memory)

BUFFER MEMORY

This section examines these three parts in detail. They are discussed in the order in which they appear in a printed dump file.

DCA MEMORY

DCA memory is part of OS MEMORY and is maintained by the MCP. This area is divided into two subsections. Both subsections reside in OS MEMORY but they are managed differently.

The first and most important area of DCA MEMORY is the resident data segment. This segment of memory forms the nucleus of the DC subsystem. It maintains an address directory for tables located in Buffer Memory (BM) and Virtual Memory (VM), and for various other size values and fields necessary to maintain the DC subsystem.

The second area of DCA MEMORY is part of VM, and contains the tables responsible for driving the DC subsystem. The information in these tables is extracted from the NDLSYS file at the time the Message Control System is loaded.

RESIDENT DCA MEMORY

A list and description of the DCA MEMORY fields resident in OS MEMORY follows. These items are discussed in the order in which they appear in the printed dump. The contents of the 2-byte fields are printed in byte-reversed order.

NOTE

In reading a printed dump, it is very important to understand the limits and basic operation of Virtual Memory. Table 9-1 gives the Data Comm tables which are maintained in VM. At the time a dump is taken, the contents of these tables may be in memory or they may be "swapped out" and located in disk memory (referred to as backing store). Any table or field may yield its space to another user. The VM algorithm determines which tables and fields are swapped out. If a table or field is in backing store when a dump is taken, the dump gives the address in resident memory at which it was last located. It is particularly important to keep this concept in mind if a Hex dump becomes necessary.

Table 9-1. DC Subsystem Table in Virtual Memory

USER JOB table	ID = 1
MCS table	ID = 2
LSNTAB table	ID = 3
LLNTAB table	ID = 4
SUBTAB table	ID = 5
NDL table	ID = 6
DCP table	ID = 7
DCPCON table	ID = 8
LSN INFO table	ID = 9
MCS NAME table	ID = A

RESERVED	This 1-byte field is reserved for special use. At the present time, this field is set to 1 if a DC.RECONFIG has been used since the loading of the DC subsystem.
MCS LOADED	Binary count of the number of Message Control Systems (MCSs) physically loaded.
DCP LIMIT	This 2-byte field contains the highest logical DCP number (defined in NDL).
BUFFER SIZE	This 2-byte field is the size (in bytes) of the Data Comm Buffer as determined in the DCP SECTION of the NDL. It is incremented by 4 when the DATA COMM LOAD ACTION is performed.
STATION COUNT	This 1-byte field contains the total number of stations defined in the NDL. Both real and dummy stations are included in this count.
USER DC LOGS	This is a 4-byte field broken out in the dump listing as two 2-byte fields, USER DC LOG1 and USER DC LOG2. Together these two fields make up a 32-bit field, each bit representing one of the 32 possible (maximum) TASK IDs (user jobs).

NOTE

Only 26 jobs are supported by the 3.04 MCP.

When a user job is executed and goes to BOJ, the appropriate bit is set; the bit is reset when the job ends (DS, DP, or EOJ). The format for these fields is as follows:

USER DC LOG1	Byte 1 (bits 7-0)
	Byte 2 (bits 15-8)
USER DC LOG2	Byte 3 (bits 23-16)
	Byte 4 (bits 31-24)

SUBNET COUNT — This 1-byte field contains the total number of subnet queues which are defined in the FILE SECTION of the NDL.

LINE COUNT — This 1-byte field contains the total number of lines which are defined in the NDL.

DC EOJ ACTION ID — This 1-byte field is initialized to @FF@ when the DC.LOAD.ACTION.ID is performed during the loading of the DC subsystem. This field holds the ACTION.ID if and when the DC.EOJ action is suspended. Suspension of the DC.EOJ action occurs when the last MCS is terminating and any one of the COBOL ACTIVE flags of the User Job Table is set.

LFN BLOCK SIZE — A 2-byte field containing the size of the link blocks used to hold the list of logical file names. The list of logical file names can be found in the Work Block/Link Block list in the Operating System section of the dump.

LSN BLOCK SIZE — A 2-byte field containing the size of the link blocks used to hold the list of Logical Station Names. The list of Logical Station Names can be found in the Work Block/Link Block list in the Operating System section of the dump.

XLSN BLOCK SIZE — A 2-byte field containing the size of the link blocks used to hold the list of Extended Station tables. The Extended Station tables can be found in the Work Block/Link Block list in the Operating System section of the dump. action ID stored in this field is used to reinstate the job.

USER JOB LAST ADDRESS — This 2-byte field is the OS MEMORY ADDRESS at which the User Job Table was last located. The contents of the User Job Table are described later in this section. If it becomes necessary to check the Hex dump, remember that this table may have been swapped out, and this address may be invalid. (LINKED BLOCK ID = 1)

TASK TO MIX TABLE	This is a 32-byte array (one per allowable task number) that can be used to find a job's external mix number when its TASK-ID is known. The job's task number is used as a 0-relative index into the array in order to obtain the mix number. This table is initialized to all 0s.
MCS ID TABLE	This 32-byte array (one byte for each allowable task number) can be used to find a task's relative MCS number when its task number is known. The MCS task number is used as a 0-relative index into the array to obtain the relative MCS number. This table is initialized to all @FF@s.
MCS TABLE LAST ADDRESS	This 2-byte field is the OS MEMORY ADDRESS at which the MCS table was last located. The contents of this table are described later in this section. If it becomes necessary to check the Hex dump, it should be remembered that this table is stored in Virtual Memory and, consequently, the Hex address may not be valid. (LINKED BLOCK ID = 2)
LSN CONVERSION LAST ADDRESS	This 2-byte field is the OS MEMORY ADDRESS at which the LSN conversion table (LSNTAB) was last located. The contents of LSNTAB are described later in this section. (LINKED BLOCK ID = 3)
LSN INFO LAST ADDRESS	This 2-byte field is the OS MEMORY ADDRESS at which the LSN INFO table was last located. The contents of this table are described later in this section. (LINKED BLOCK ID = 9)
LINE CONVERSION LAST ADDRESS	This 2-byte field is the OS MEMORY ADDRESS at which the Logical Line (LLN) Conversion Table was last located. The contents of this table are described later in this section. (LINKED BLOCK ID = 4)
SUBNET INFO LAST ADDRESS	This 2-byte field is the OS MEMORY ADDRESS at which the SUBNET INFO table was last located. The contents of this table are described later in this section. (LINKED BLOCK ID = 5)
DCP CONVERSION LAST ADDRESS	This 2-byte field shows the last address for the DCPCON Table. (LINKED BLOCK ID = 8)
DCP TABLE LAST ADDRESS	This 2-byte field shows the last address for the DCP Table. The contents of the DCP Table are described later in this section. (LINKED BLOCK ID = 7)

NDL DATA LAST ADDRESS

This 2-byte field gives the last OS MEMORY ADDRESS of the NDL Table. The contents of the NDL Table are described later in this section. (LINKED BLOCK ID = 6)

MCS NAME LAST ADDRESS

This 2-byte field gives the last OS MEMORY ADDRESS at which the MCS NAME table was last located. The contents of this table are discussed later in this section. (LINKED BLOCK ID = A)

ABP ADDRESS

This 4-byte field is the offset, page, and bus address of information about the Available Buffer Pool. The first two bytes of this field contain the offset, the third byte is the page number, and the last byte is an address pointing to where buffer pool information is stored. This second location has a 12-byte field divided into five groups:

Read With Lock Word (RLW)	1 byte:	00 = unlocked 01 = locked
Processor ID	1 byte	
Buffer Count	2 bytes	
ABP Head Pointer	4 bytes:	page, bus, address
ABP Tail Pointer	4 bytes:	page, bus, address

The APB TAIL POINTER field is printed by the dump at the beginning of the ABP area in the BUFFER section.

REQUEST Q ADDRESS

This 4-byte field gives three pieces of information. The first two bytes contain the offset, the next byte is the page number, and the last byte gives the beginning address where information describing the first REQUEST Q is stored. The information available is:

Read With Lock Word (RLW)	1 byte:	00 = unlocked FF = locked
Request Q Head Address	4 bytes:	page, bus, address
Request Q Tail Address	4 bytes:	page, bus, address

There can be up to nine request queues (one for each DCP) numbered 0 through 8. Request queues are printed by the dump in the BUFFER section.

SUBNET Q ADDRESS

This 4-byte field is divided into three sections: first two bytes contain the offset, the next byte is the page number, and the last byte is the beginning address at which information describing the first subnet queue is

stored. There is one subnet queue for each file defined in the NDLSYS file. The information is supplied in 10 bytes, divided into four fields:

Queue Limit	1 byte
Queue Count	1 byte
Subnet Q Head Address	4 bytes: page, bus, address
Subnet Q Tail Address	4 bytes: page, bus, address

The subnet queues are printed in the dump and are located in the BUFFER section.

RESULT QUEUE ADDRESS — This 4-byte field contains the offset in the first two bytes, the page number in the next byte, and the last byte contains the bus address where information describing the result queue is stored. There is only one result queue in a data comm subsystem. It is used to return information to the OS processor.

Ten bytes describe the result queue:

Read With Lock word	1 byte: 00 = locked FF = unlocked
Processor ID	1 byte
Result Q Head Address	4 bytes: page, bus, address
Result Q Tail Address	4 bytes: page, bus, address

The result queue is printed by the dump in the BUFFER section.

DC BUF MEM ADDRESS — This 4-byte field is the page, bus, and offset in buffer memory where the data comm buffers begin.

RESERVED — These four bytes are currently unused.

LINE INFO SIZE — This 2-byte field is the size of each line info area. Each line info area immediately precedes its respective line table in DCP memory, and contains the additional line information needed by the DCP. (For a description of line information area fields, see DCP Memory Section.)

PHYSICAL DCP LIMIT — This 2-byte field contains the highest physical DCP number attached to the system.

DC LIST ID — This 1-byte field contains the value used to identify the elements in the data comm linked block list in OS Memory, and is used when any data comm linked blocks are accessed.

VIRTUAL DCA MEMORY

This is the second part of the DCA Memory area. Descriptions of the tables and fields within Virtual Memory follow. The items are given in the order in which they appear in the printed dump.

USER JOB TABLE (LINKED BLOCK ID = 1)

This table has an entry for each job declared in the mix. For each job, the following fields of information are printed in the dump listing:

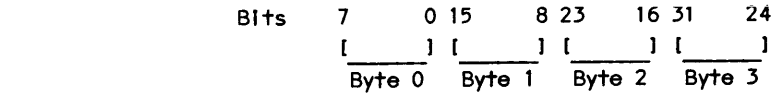
OUTPUT LIMIT/COUNT

This array contains 32 entries, one for each allowable task number. Each entry consists of two bytes: The leftmost byte is the current value of the task's output limit; the rightmost byte is the current value of the task's output count. These values are initialized to @02@ and @00@, respectively, for each user job by the NDL compiler. The user job's task ID is used as an index into this array.

COBOL ACTIVE FLAGS

This 4-byte field consists of 32 bits, one for each allowable task number. Each time a user job is actively executing VERB33, a bit is set for that job's task number. This flag will prevent the last MCS from going to EOJ and terminating the whole DC subsystem while VERB33 is active for a task. DC.EOJ suspends itself if any of these bits are set.

The layout of the 32 bits in memory is as follows:



WAITING QUEUE

This array contains an entry for each allowable task number. Each entry contains the queue reference that the task is waiting on, or @FFFF@ if the task is not waiting on a queue. The job's task number is used to index this array.

LAST COMM

This 32-byte array contains an entry for each allowable task number. Each entry contains the active ADVERB of every VERB33 action being performed. At the end of the VERB33 action, the entry is restored to @FF@. The job's task ID is used to index this array.

IPC ORIGIN STATION

This array contains 32 2-byte entries, one for each allowable task number. This array is used by the IPC communicate to save the LSN of the sending IPC program. The LSN can be cross-referenced by using the LSN INFO Table. The job's task ID is used to index this array.

LAST QUEUE	This 32-byte array contains an entry for each allowable task number, and is used with the Complex Wait function of MPLII and COBOL. The queue identity number for which the job has been suspended is stored in this field. This field works with a round-robin event scanner, looking for all or any one of the subnet queues which were specified in the Complex Wait statement. The job's task ID is used to index this array.

MCS TABLE (LINKED BLOCK ID = 2)

This table contains an entry for each MCS declared in the mix. For each MCS, the following fields of information are printed in the dump listing:

MCS	This is the relative MCS number. This value is not stored in the table, but rather, is used as an index into the table.
MRA	This 1-byte field is the ID of the linked block that contains the MCS message reference area. It can be used as an index into the linked block list to reference the MRA.
QUEUE HEAD	This 4-byte field is the bus, page, and offset address of the head of the MCS queue.
QUEUE TAIL	This 4-byte field is the bus, page, and offset address of the tail of the MCS queue.
QUEUE COUNT	This 2-byte field contains the current number of messages on the MCS queue.
TASK ID	This 1-byte field holds the MCS's internal task number.
MSG REF	This 1-byte field is the size of the MRA link block defined in the MCS source program. The value in this field reflects the size of the Message Reference Area (MRA) as it was defined in the MPLII MCS program. Refer to sections 10 and 12 of the MPLII Reference Manual, form 2007363, for additional information.

LSNTAB (LINKED BLOCK ID = 3)

This is the logical station conversion table. This table contains an entry for each station defined in NDL. For each station the following fields of information are printed in the dump listing:

LSN	The logical station number as defined in NDL. This is used as the index when using the LSNTAB.
PROCESSOR	This 2-byte field holds the number of the

	physical DCP to which this station is attached. If this field contains @FFFE@, the DCP is unloaded due to an error condition in the processor. If this field contains @FFFF@, the station was not declared to be attached to a line in the NDL, or was detached by a REDEFINE.STATION communicate.
STATION TABLE ADDRESS	This 2-byte field contains the address of the Station Table in DCP MEMORY. Refer to the Processor field to determine which DCP Memory to reference in the dump.
DISK TABLE SIZE	This 1-byte field contains the size (in bytes) of the Station Table on disk for this station.
ORIGINAL OWNER	This 1-byte field contains the relative MCS number of the MCS that was assigned to this station in the NDL file. If no MCS was assigned in NDL, this field is initialized to @FF@, and will later be assigned the relative MCS number of the first MCS to access this station. If this is a single-MCS system, then this field is initialized to @00@.
CURRENT OWNER	This 1-byte field contains the relative MCS number of the MCS that currently owns this station. It is initialized to the value of the original owner.
LLN	This 1-byte field contains the logical line number of the line that this station is currently attached to. If the station is not currently attached to a line, this field contains @FF@.

LLNTAB (LINKED BLOCK ID = 4)

This is the logical line conversion table. This table contains an entry for each line defined in NDL. For each line the following fields of information is printed in the dump:

LLN	This is the logical line number as defined in NDL. This value is used as the index into the LLNTAB.
PROCESSOR	This 2-byte field contains the physical DCP number to which this line is attached. If this field contains @FFFE@, the DCP has become unloaded due to an error condition in the processor.
LINE TABLE ADDRESS	This 2-byte field contains the address where the Line Table is found in DCP Memory. Refer to the PROCESSOR field to determine which DCP to reference in the dump.

DISK TABLE SIZE	This 2-byte field contains the size (in bytes) of the Line Table on disk for this line.
RECONFIG PENDING	This 1-byte field is a flag used to prevent two MCSs from simultaneously redefining the same line, or from executing RELOAD while RECONFIGURATION or another RELOAD is executing. This flag is set to @FF@ if a REDEFINE or RELOAD is executing for this line; otherwise, it is set to @00@.

SUBTAB (LINKED BLOCK ID = 5)

This table contains an entry for each subnet declared in NDL. For each subnet, the following fields of information are listed:

SUBNET	The relative subnet number as declared in NDL. This is used as the index into the Subnet Table.
ORIGINAL OWNER	This 1-byte field contains the relative MCS number of the MCS that was assigned to this subnet in the NDL file. If no MCS was assigned in the NDL, this field is initialized to @FF@, and later will be assigned the relative MCS number by the first MCS to access this subnet. If this is a system subnet, this field is initialized to @00@.
CURRENT OWNER	This field contains the relative MCS number of the MCS that currently owns this subnet. It is initialized to this value by the original owner.
ATTACHED MIX TABLE INDEX	This 4-byte field contains 32 bits, one for each allowable task number. A bit is set for each user job that is currently attached to this subnet. Task numbers range from 0 to 31 and are printed here as a set of 32 decimal digits.

NDL (LINKED BLOCK ID = 6)

This table contains information that points to the various structures and data within the NDLSYS file.

MODEM COUNT	This 1-byte field contains the number of modems defined in the NDL.
TERMINAL COUNT	This 1-byte field contains the number of terminals defined in the NDL.
STATION TABLE SIZE	This 1-byte field contains the size of each of the Station Tables in the NDLSYS file.
NDLSYS FIB ID	This 2-byte field contains the ID of the system FIB for the NDLSYS file.

RECON FIB	This 2-byte field contains the ID of the system FIB for the RECON file.
LINE TABLE ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the Line Tables begin. Line Tables are on disk in LLN order. Full details of the format of the NDLSYS file will be found in the Data Comm Subsystem Reference Manual (form 1090909).
LINE DISP ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file of the Line Displacement Table. This table contains a 2-byte pointer to each Line Table. Index into this table is by LLN.
STATION TABLE ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the Station Tables begin. Station Tables are on disk in order of LSN.
STATION DISP ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the Station Displacement list begins. This table contains a 2-byte pointer to each Station Table. Index into this table is by LSN.
MODEM TABLE ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the Modem Tables begin.
MODEM TABLE LENGTH	This 2-byte field contains the length in bytes of the Modem Tables disk area.
TERM TABLE ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the Terminal Tables begin.
TERM TABLE LENGTH	This 2-byte field is the total length (in bytes) of the Terminal Table on disk.
FILE TABLE ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the File (subnet) Tables begin.
EX-STATION TABLE ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the Extended Station tables begin.
EX-STATION TABLE LENGTH	This 2-byte field contains the total length (in bytes) of the Extended Station Tables disk area.
EX-TERM TABLE ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the Extended Terminal Tables begin.

EX-TERM TABLE LENGTH	This 2-byte field contains the total length (in bytes) of the Extended Terminal Tables disk area.
STATION NAME TABLE DISK ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the station name table begins. The names are arranged alphabetically within the table and each entry is 12 bytes long, space filled on the right.
STATION NAME TABLE LENGTH	This 2-byte field contains the total length (in bytes) of the station name tables disk area.
FILE NAME TABLE ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the File Name Table begins. The file (subnet) names are arranged alphabetically within the table, and each entry is 12 bytes long, space filled on the right.
FILE NAME TABLE LENGTH	This 2-byte field is the total length (in bytes) of the File (Subnet) Name Table on disk.
DCP TERMINALS ADDRESS	This 2-byte field contains the disk file record number in the NDLSYS file at which the DCP Terminals (FORMAT B) segment begins. This segment contains information concerning the DCP code files and their associated terminals.
DCP TERMINALS LENGTH	This 2-byte field contains the total length (in bytes) of the DCP Terminals (FORMAT B) disk information.

DCP (LINKED BLOCK ID = 7)

This table contains an entry for each DCP declared in NDL. For each DCP, the following fields of information are displayed:

DCP	The logical DCP number. This is used as the index into the DCP table.
FILENAME	This 2-byte field contains the name of the codefile that is currently loaded into this DCP.
PHYSICAL DCP NUMBER	This 2-byte field contains the physical DCP number associated with this logical DCP. If the DCP was never loaded or is frozen, this field contains @FFFF@.

DCPCON (LINKED BLOCK ID = 8)

The DCP Conversion Table is an 8-byte array (one byte for each allowable DCP) that can be used to find a DCP's logical DCP number when its physical DCP number (bus address) is known.

The physical DCP number is used as an index into the array to obtain the logical DCP number.

LSN INFO (LINKED BLOCK ID = 9)

This table contains additional information for each station declared in the NDL. For each station, the following fields of information are displayed:

LSN	The logical station number as defined in NDL. This is used as an index into the LSN INFO table.
OUTPUT QUEUE	This 1-byte field contains the output routing for this station. A value of @FF@ indicates that output to this station will be routed to the MCS queue (MCS participating). A value of @00@ indicates that output for this station is routed directly to the station queue (MCS non-participating). A value of @FE@ indicates that output to this station will be routed to the IPC subnet queue.
OUTPUT SUB Q	This 1-byte field contains the valid subnet queue number for this dummy station. This field is used to support the IPC interface.
INPUT QUEUE	This 1-byte field contains the subnet number of the subnet queue that this station's input will be routed to. @FF@ indicates that this station's input will be routed to the MCS queue (MCS participating).
ATTACHED MIX TABLE INDEX	This 4-byte field contains 32 bits, one for each allowable task ID. A bit is set for each user job that is currently attached to this station. Task numbers range from 0 to 31, and the field is printed as a string of 32 decimal numbers.

MCSNAME (LINKED BLOCK ID = A)

This table contains the number of MCSs defined in the NDL, and the name of each MCS. This information is displayed as follows:

MCS	This is the relative MCS number. It is used by the system as an index into the MCS Name list.
PACK ID	This 7-byte field contains the Pack ID specified in NDL for this MCS.

FILENAME	This 12-byte field contains the MCS File ID.
MCS-COUNT	This 1-byte field contains the number of MCSs defined in NDL. For single-MCS systems, this field contains @FF@.

DCP MEMORY

The second part of the Data Comm Section is DCP Memory. The DCP memory contains the microcode generated by the NCP900 or NCP900P program, the global data area, and the line- and station-related information. These data areas are divided into two subsections, the first being the Reserved Pointer Area, and the second being the Line Information Area. If the DC subsystem has more than one DCP, the lowest bus address is dumped first, followed by the next higher DCP bus address.

RESERVED POINTER AREA

This area contains all of the address and stack pointers necessary to maintain this DCP. In addition, this area contains the global data area, and holds information pertaining to PI, line and station sizes, processor registers, SAVE state, and processor status information. In short, the majority of key DCP information is contained in this area. This information is displayed as follows:

DCP ADDRESS	This is the bus address of this DCP and is determined by the Analyzer.
ERRORFORCE, INTERRUPTFORCE, CLEARFORCE	These three fields are used by the microcode when forced through low memory by the processor.
OS PROCESSOR ID	This two-byte field is the bus address of the OS Processor. It is not byte-reversed.

INPUT MAILBOX (IMB) IMB REQUEST FUNCTION IMB REQUEST PROCESSOR IMB RESULT PROCESSOR IMB RESULT FUNCTION IMB SYSTEM MIX NUMBER IMB AWAITING MAILBOX IMB FLAGS OUTPUT MAILBOX (OMB) OMB REQUEST FUNCTION OMB REQUEST PROCESSOR OMB RESULT PROCESSOR OMB RESULT FUNCTION OMB SYSTEM MIX NUMBER OMB AWAITING MAILBOX OMB FLAGS MY MAILBOX FLAG NM FLAG	These fields are the standard PI fields. Layout and content of mailboxes is covered in Section 10. The only values that are used by the DCP for request function are @7C@ (dummy poll response) and @08@ (DCP waking up DCA to look at result queue).

DCP ID	This field is initalized by the Data Comm Loader and should be the Processor ID of this DCP.

NDLSYS DATE STAMP, RANDOM NUMBER — These two fields are used to check that the NPC microcode file and the NDLSYS file are a matched pair. During NPC900, the random number is inserted into both NDLSYS and the microcode file(s) that is (are) being generated.

LINE INFO AREA SIZE — This field contains the size in bytes of the LINE INFO area.

PRI LINE INFO AREA SIZE — This 1-byte field is the size of the extra area needed for multi-MCS control information. These fields are present in the primary LINE INFO area only.

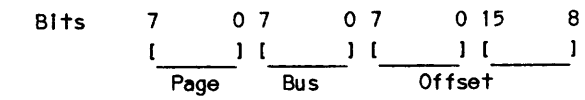
DCP END ADDRESS — This is the address in DCP Memory of the end of the microcode.

DCP TABLES BEGIN ADDRESS — DCP tables are loaded at the top of DCP Memory. This address points to the bottom of the table area. (This should be greater than DCP end address.)

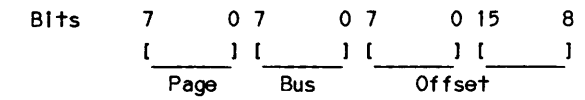
BUFFER MEMORY READ ADDRESS — This is the location of the first page of data comm buffers.

BUFFER MEMORY RWL ADDRESS — This is the location of the first page of data comm buffers with the RWL bit set.

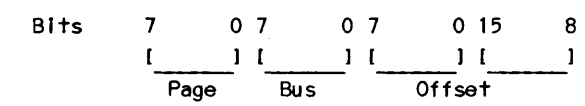
PTR TO REQUEST Q PTR — This is the address of the Request Queue pointer.



PTR TO RESULT Q PTR — This is the location of the Result Queue pointer.



PTR to AVAIL.POOL PTR — This is the location of the data comm Available Pool pointer.



LINE VECTOR TABLE	This is an array of 16 2-byte entries, one per port. Each entry is the address of the related Line Table entry. Unused ports have an address of @FFFF@. Port 0 is used for interprocessor communication (PI). This port should have an entry of @FFFF@.

NOTE

For entries HC LINE NEXT POINTER through HOST LINE PRIORITY, for the purposes of SCHEDULER, host control (HC) is treated as an additional line, and has a pseudo LINE INFO area. This group of fields is that area.

HC LINE NEXT POINTER	This is the forward link to the next line. As HC is always the "last line," this points to the first line in the queue for Line Manager.
HC LOOP FLAG	Unused.
HC LINE FUNCTION	Address of code to be executed by host control when it gains control.
MAX STATION TABLE SIZE	Indicates the size of the station tables used in this DCP (either normal or extended).
HC LINE CHANGE COUNTER	HC only runs every nth cycle (currently every 10th). This counter controls the size of the cycle.
HC LINE BACKWARD POINTER	This is the backward link to the previous HC line.
HOST LINE PRIORITY	This field is always 0.
PI SEND Q HEAD	DCP memory address of the head of the PI queue.
DCP RESULT Q HEAD ADDRESS	Address in buffer memory of the head of the DCP result queue.
DCP RESULT Q TAIL ADDRESS	Address in buffer memory of the tail of DCP result queue.
DUMMY STATION TABLE	Used as a work area during Station Table operations.
LINE INFO BASE ADDR TABLE	This table contains the address of the Line Table entries in LLN order. All unused entries are filled with @FFFF@.

RELOAD IN PROGRESS — This 1-byte field is set to @FF@ by the OS to request that the DCP enter its IDLE state in preparation for reloading. This field is reset (@00@) by the DCP when it is "idle".

ENHANCEMENT BYTES — Reserved.

NPC VERSION — This is the version of the NDL Post Compiler (NPC) used to generate the microcode file currently in the DCP.

STATE SAVE AREA — If the DCP develops a problem or becomes "hung," this area allows the same degree of debugging as any other processor. A detailed description of the State Save Area is contained in the OS Processor section.

ERROR NUMBER — This field in the State Save Area is special to the DCP processor. Error numbers and their meanings are:

@00@ : Hardware error.

@01@ : Interrupt error.
An unexpected interrupt has occurred.

@02@ : NDL error.
The DCP has detected an error in the supplied NDL that makes further execution unsafe.

@03@ : Initialize error.
The DCP branched to address 0, and has gone through initialize code twice.

Errors @02@ and @03@ will probably require TIO or plant resolution.

The values in the State Save Area have no meaning unless the data comm was active at the time of the dump.

LINE INFORMATION AREA

Information regarding the lines under control of this DCP is held in three parts. The first part is the "Line Number" area. This area holds all pointer information work areas, flags, etc. The second part is the standard CMS "Line Table." The third part is the "Station Table" entries for all stations on this line.

LINE NUMBER

NEXT POINTER — This 2-byte field contains the absolute address of the next line info area in the "Round Robin" queue.

PORT NUMBER	This is the physical port number. It is maintained as a bit mask where Port 2 = @04@, port 7 = @80@.
FUNCTION	This is the address of the microcode that handles the anticipated event. If the line has never been made ready, this field will be 0.
RETURN POINTER	The code address to return to when the active function completes.
BACKWARD POINTER	Points to the previous entry in the "Round Robin" list. Only used in queue delinking.
PRIORITY CODE	The line priority to be used when placing this line into the top-down Manager queue.
PRIORITY PTR	Points to the next line at the highest priority. If this is the only one, then this field will point to itself. This field is used in the Line Manager for top-down scheduling.
ACTIVE STATION PTR	Contains the DCP memory address of the Station Table entry of the currenly active station (if any).
ACTIVE STATION	Contains the RSN of the currently active station (the NDL variable "STATION").
ACTIVE STATION VECTOR PTR	Contains the DCP memory address of the current station's Station Vector in the Line Table. The Station Vectors are contained in the CMS table station descriptors. This field is redundant for the dump reader, as the appropriate entry can be readily found by use of the RSN field.
WORK1 thru WORK3	Temporary storage areas.
CURRENT BUFFER	Absolute address of the base of the current buffer.
THIS BUFFER SIZE	Contains the remaining buffer size. This field is updated at buffer crossover.
THIS BUFFER SIZE SAVE	This is used to store "THIS BUFFER SIZE" at buffer crossover time. This is necessary to enable the system to correctly handle a BACKSPACE after the last character has been stored.
SAVE POINTER	Used by subroutines that enter the Manager. This is not the normal mode of entry. It is used when a needed return address sits on the top of the NBDS stack and the future integrity of the stack is in danger due to an impending yield to another line.

BUFFER SIZE	Contains the remaining buffer space. This is maintained in twos complement and incremented with each character stored. A value of @FFFF@ indicates overflow.
BUFFER COUNT	The absolute address of the next character position to be read from or written to.
TERMINAL NUMBER	Contains the logical terminal number associated with the currently active station.
INPUT CHARACTER	Contains the translated and checked image of the last character on the line. (NDL CHAR)
CHARACTER	Contains the last character as it appeared on the line. (NDL LCHAR)
TIMEOUT	Used to store S-OP-defined times.
TEXT SIZE	Contains the current amount of text accumulated or transmitted so far. This is maintained in ones complement form. It is updated only on buffer crossover by the amount in THIS.BUFFER.SIZE.
RETURN TO POINTER	Contains an S-OP address where control will be returned when current function is completed.
CRCL	Holds lower byte during CRC calculations.
BCC CRCM	Holds BCC or upper CRC byte during BCC/CRC generation.
ADAPTOR DESCRIPTOR	A copy of the DCI hardware register.
DATA SET DESCRIPTOR	A copy of the DCI hardware register.
MESSAGE HEADER/LENGTH	Contains MAXINPUT for the current terminal.
TERMINAL SAVE Q LIMIT	Contains the limit for the Output Save Queue of a given station. The limit is specified in the station's Terminal Table. Because the terminal tables are not stored in DCP memory, the TERMINAL SAVE Q LIMIT is stored in the Line Configuration area and updated at station change time.
YIELD COUNTER	Used to ensure that a high speed line does not monopolize a data comm processor's time.
AT LEAST TWO CHARACTERS	Used to determine if certain BDLC receive S-OPs have received at least two characters.

STREAM A — Used to buffer TRANSMIT TEXT data from data comm space to the adaptor.

STREAM B — Same as Stream A.

ADAPTOR ABORT INFO — This is two 1-byte fields which contain information to be copied to the RESERVED and SUBQ fields of a Line Not Ready result header (with adaptor fault event set). These fields are also valid in the case where a function is returned with the result UNABLE TO INITIATE caused by an adaptor mismatch. The following information is helpful:

Subnet Queue	Description	Corresp. Reserved Field	Description
0	No CTs after 8 seconds	n/a	
1	No adaptor present	0	Delay operation
		1	Transmit
		2	Receive
		3	Switched line establish dialout
		4	Dialout
		5	Adaptor lost
2	Adaptor not dialout capable	n	Where n = ID of adaptor
3	Line address requires dual ENDC	n	Where n = ID of adaptor
4	LEM link parity	0	Read
		1	Write

MESSAGE HEADER POINTER — Contains the absolute address of the message area currently in use.

HOST WORK1 thru 4 — Used as host control work areas.

DUMMY TALLY, DUMMY DESCRIPTOR, DUMMY STATION POINTER — Used in place of STATION TALLY, STATION DESCRIPTOR, and STATION POINTER when an invalid station has been chosen.

FLAGS — These fields reflect the state of the line. Individual bytes contain the following flags.

	Bits	Message	Meaning
FLAG 1	7	NOT USED	
	6	CRC	The CRC toggle in NDL. If TRUE, horizontal parity is generated. If FALSE, horizontal parity is not generated.

	Bits	Message	Meaning
	5	NO.TRANSLATE	No translation
	4	SYNCS	Store received syncs
	3	FULL DUPLEX	Set for full duplex
	2	TRANSPARENT	Line is in transparent mode Set/Reset by binary = TRUE/FALSE
	1	MIDDLE SHIFT MODE	When using caseshift translation,
	0	UPPER SHIFT MODE	these flags define the shift mode in effect.
FLAG 2	7	RCV TEXT	Set while receive text is in progress so that the common receive manager can distinguish a RCV text from a RCV character.
	6	ZERO TIMEOUT	The last receive instruction had 0 as its time value. (0 requires special handling in the manager.)
	5	INPUT FLAG	If set line had no output space for half-duplex, this implies that line is in line control or input request mode.
	4	LINE CONTROL	Set when NDL is running the line control or auxiliary line control of the program for this line.
	3	SPACE AVAIL	Message space is available. Set after successful getspace, or on entering an output request
	2	HORIZONTAL.ODD	Set if using odd horizontal. parity.
	1	CRC.1	Indicates CRC polynomial in use: 0 : x**16+x**15+x**2+1 1 : x**16+x**12+x**5+1
	0	SYNC/ASYNC	TRUE implies ASYNC.
FLAG 3	7	FUNCTION IN PROGRESS	If set, HC does not get a message off the request queue for this line. Also, inhibits S-OPs from interrupting the line, irrespective of the state of the line (busy).
	6	VERTICAL EVEN	Set if even vertical parity is present.

	Bits	Message	Meaning
	5	NEW CHAR PRSNT	Set if receive manager received new character from line.
	4	STA.NRY.PENDING	A Make-Station-Not-Ready request has been made for this station. It will be done after THIS.FUNCTION. implies that line (busy) is TRUE.
	3	LN.NRY.PENDING	As above, but for line.
	2	XMT/RCV LAST	Indicates whether the most recent function was a transmit or receive. Used in byte variable and toggle accessing for values that are maintained on the adaptor (1 for each side XMIT and RCV). True = RCV.
	1	ABORT IN PROGRESS	In a multi-MCS environment, the DCP is in the process of informing all MCPs that a line abort has occurred.
	0	FLAG.FILLING	The BDLC line is currently flag filled.
FLAG 4	7	WAIT FLAG	Set if line is in Wait.
	6	AUX ACTIVE	Set if auxiliary line is in line queue.
	5	AUX	Indicates that this Line Information area is an auxiliary of a full-duplex pair.
	4	IGNORE BREAK	Indicates to the transmit messenger that the S-OP wishes to ignore a break condition on an ASYNC line and underflows on a SYNC line.
	3	FLAG EXPECTED	The receiver expects a BDLC flag.
	2	OLD TRANSPARENT	Reflects the state of transparent for the most recent XMIT/RCV. Aids in the retrieval of the character, since transparent could have changed state from the time the character was received and its access.
	1	ADAPTOR IN USE	An instruction requiring the presence of a line adaptor has been executed.

	Bits	Message	Meaning
	0	NOT USED	
FLAG 5	7	TERMINATING BLOCK	Set if terminating block.
	6	NO LINE CTRL	Set if shouldn't re-write line control after pending resolution.
	5	DCIB TRUE	Set if the line adaptor is DCIB.
	4	HOST MCS FLAG	Indicates logical state of the associated MCS.
	3	NOT USED	
	2	INV FRAME	Set if the BDLC line has received an invalid frame. Its use is internal to the routine that processes receive errors. Should be 0 outside that routine.
	1	ABORT LINE	Set if LEM line error occurred during Make-Line-Ready process before Line Linked is set.
	0	LINE LINKED	Set if Line Ready process has linked line into line list. Line can be made Not Ready at this point.
FLAG 6	7	LINE LOAD	Indicates which stream field must next be loaded with transmit data: 0 = String A 1 = string B
	6	LINE STREAM	Indicates which stream field is current: 0 = string A 1 = string B
	5	LINE BITS	Set if line is BDLC.
	4	TERMINATING DISABLE OUTPUT	Set if Terminate Disable input is being processed to distinguish it from a Terminate. Error is common code.
	3	NOT USED	
	2	NOT USED	
	1	NOT USED	
	0	INVALID QSR	Set when the DCP firmware notices a peculiar condition on the line adaptor.

COUNT A	The number of characters in stream A.
COUNT B	The number of characters in stream B.
PRIMARY POINTER	Address of the LINE INFO AREA of the primary line of a full-duplex pair. For half-duplex pair, for a half-duplex line, or for the primary, this will point to itself.
CO LINE POINTER	Address of the LINE INFO AREA of the aux line of a full-duplex pair. For a half-duplex line, this field is @FFFF@. For the aux line of a full-duplex pair, this field points to its primary.
LNE PENDING MCS ID	This 1-byte field is used if an MCS is making a line Ready/Not Ready, and holds the MCS relative routing number of the requesting MCS. This field is used later (when the action is finished) to route the result of the line condition back to the parent MCS.
LNE LOG LN RDY COUNT	This 1-byte field is used to hold the number of lines the MCS has ready.
LNE HOST MCS ID	This 1-byte field is used as a Save Area for the MCS ID, and is used in relationship with any pending operation (such as making a line Ready). When the operation (function) is completed, this field is copied into the result descriptor header.
LNE MCS LOG STATE	This array is 16 bytes long, and is used to hold information on the state of the lines (one byte for each line). @00@ indicates the line is Not Ready, @FF@ indicates that it is Ready.
LNE SNR MCS DATA	This 2-byte field is used to hold the MCS data field of a MAKE.STATION.NOT.READY message header if that header encounters a Pending condition. This field will be copied to a new header and returned to the MCS when the Pending condition is resolved.
LNE LNR MCS DATA	Same as above, for MAKE.LINE.NOT.READY.
LINE TABLE	The meaning of the contents of this table are identical to that described in the NDL tables in this section of the manual. For information on the usage of the Line Table, refer to the Data Communications Subsystem Reference Manual, form 1090909.
STATION TABLE	This table is identical to the area described in the NDL tables found in this section. For

additional information on the Station Table, refer to the Data Communications Subsystem Reference Manual.

BUFFER MEMORY

The third and last area of the data comm dump section is the set of data comm buffers. Buffers are listed in relation to the structure to which they belong. All buffers are laid out in the following manner:

Byte	Description
0 - 3	Link to next buffer in this message with @FFFF@ the last buffer.
4 - 7	If this is the first buffer of a message, it points to the next message. If it is not the first buffer, the field is @FFFF@.
8 - n	Message text.

NOTE
In the first buffer of a message, the first 36 bytes contain the MCS header.

AVAILABLE BUFFER POOL (ABP)

The Available Buffer Pool is a set of buffers. The size was allocated by the DC BUFFER field of SYS.CONFIG and by the NDLSYS file using the DCP section buffer fields. All buffers are dumped to allow a trace of a recent message. By using the two link fields, buffers can be linked forward or backward. Information on the linking fields follows.

ABP READ WITH LOCK	A 1-byte field used by the Read-With-Lock hardware (@00@ = unlock, @FF@ = locked).
ABP PROCESSOR ID	This contains the ID of the processor using the ABP.
ABP COUNT	This field contains the number of buffers in the ABP.
ABP HEAD	This field points to the head of the ABP.
ABP TAIL	This field points to the tail of the ABP.

The ABP buffers are located here in the printed dump.

RESULT QUEUE

The result queue area is laid out similarly to the ABP. The fields are identical in function to those of the ABP.

REQUEST QUEUE

The request queue for each DCP number is given here. The fields are identical in function to those of the ABP.

SUBNET QUEUE

The contents of each subnet (NDL file) is given here. The fields have the following significance:

QUEUE NUMBER	Analyzer developed reference (logical Q no.).
QUEUE LIMIT	The current queue limit for this subnet. Default value is 2.
QUEUE COUNT	The number of messages in this subnet queue.
QUEUE HEAD/TAIL	The addresses of the head and tail of the subnet queue.

SECTION 10
TASK PROCESSOR

INTRODUCTION

This section of the manual provides a field-by-field description of the Task Processor (TP) portion of the dump. Data contained in the TP section includes:

Input and output mailboxes used to communicate with the OS processor
Pointers used to locate system and task structures in the task processor memory
State Save Area, used to store information when a nonrecoverable error is detected
Analysis of memory utilization
Breakdown of interpreter and task run structures
Status information for each task

A section titled "Task Processor on BUS nn," is printed for each TP on the system. The first indication of a task processor failure is usually the message, "Task Processor <BUS ADDRESS> LOGICALLY NOT READY," displayed on the SPO. In this instance, the procedure discussed in the following paragraphs should be performed.

INITIAL DUMP ANALYSIS

There can be one of two reasons for a task processor to be in a LOGICALLY NOT READY state. The first reason is that the task processor detected an internal failure and stopped communication with the OS processor.

The second possible reason is that the OS processor failed to receive a response from the task processor which the operating system had polled.

In both of these cases, the OS freezes the task processor and attempts to create a system dump. However, it should be noted that, in the first case, a processor maintenance entry will be logged. This entry contains the same information as the State Save area.

The initial step in dump analysis is to identify the bus address of the failing task processor. This can be achieved by looking at the system log or by examining one of the following fields in the system dump:

Frozen Processor field found in the Dump File Parameters section (page 2 of the formatted dump file)

Dump File Map (third section in the dump file)

PI Dead Processor found in the PI Data Area of the dump's OS section (which follows the History Communicate Table).

The Save State Address is always primed. If the OS detected the error, the Save State information will be invalid and will be initialized to 3 bytes of @FF@, 49 bytes of @00@, and 10 bytes of @FF@.

If the task processor detected the error, the Save State information will be valid.

If the error number is @47@, the error was hardware-detected. Interrogation of the IC Status fields found in the State Save Area will indicate the type of hardware failure. More detail on State Save Area analysis is contained in section 8 of this manual.

If the error number did not indicate a hardware failure, the fields IMB FUNCTION and IMB FLAGS should be examined. These fields indicate the last function that was performed and the current status of the input mailbox. Possible entries and their messages are discussed later in this section.

Repeat this step for the OMB REQUEST FUNCTION and the OMB FLAGS. If the reason for the error cannot be determined, further analysis will be required. The rest of the task processor fields are detailed in the following subparagraphs in the order in which they appeared in the dump.

TASK PPROCESSOR ON BUS nn

ACCOUNT	This is the address resolution count. The information shown here is used in the detection of thrashing.
STATE SAVE AREA ADDRESS	This field will always be primed to the address of the State Save Area.

MAILBOX DESCRIPTIONS

The Input Mailbox is used for messages from the OS to the TP. The Output Mailbox is used for messages from the TP to the OS. For a full explanation of mailboxes and their use in interprocessor communication see section 4.

INPUT MAILBOX (IMB) TEXT	This area is used to store the message passed. The exact usage is defined by the request function field. The text area contains the Fetch Communicate Message (FCM) if communicate information is indicated in the request function field.
IMB REQUEST FUNCTION	This field is used to indicate the type of information in the text area. When the request bit (MSB) is set in the IMB flags, the OS is requesting the task processor to perform a function. These functions are as follows: @07@ Command handler. @08@ Interpreter load. @09@ Monitor action. @FF@ Data move function. Parameters to these functions are in the IMB text area, as shown in figures 10-1 and 10-2; responses are shown in figure 10-3. If the TP had previously requested, by means of its output mailbox, action in the OS, there may be a result to be passed back to the TP. In this case, the request function will be the

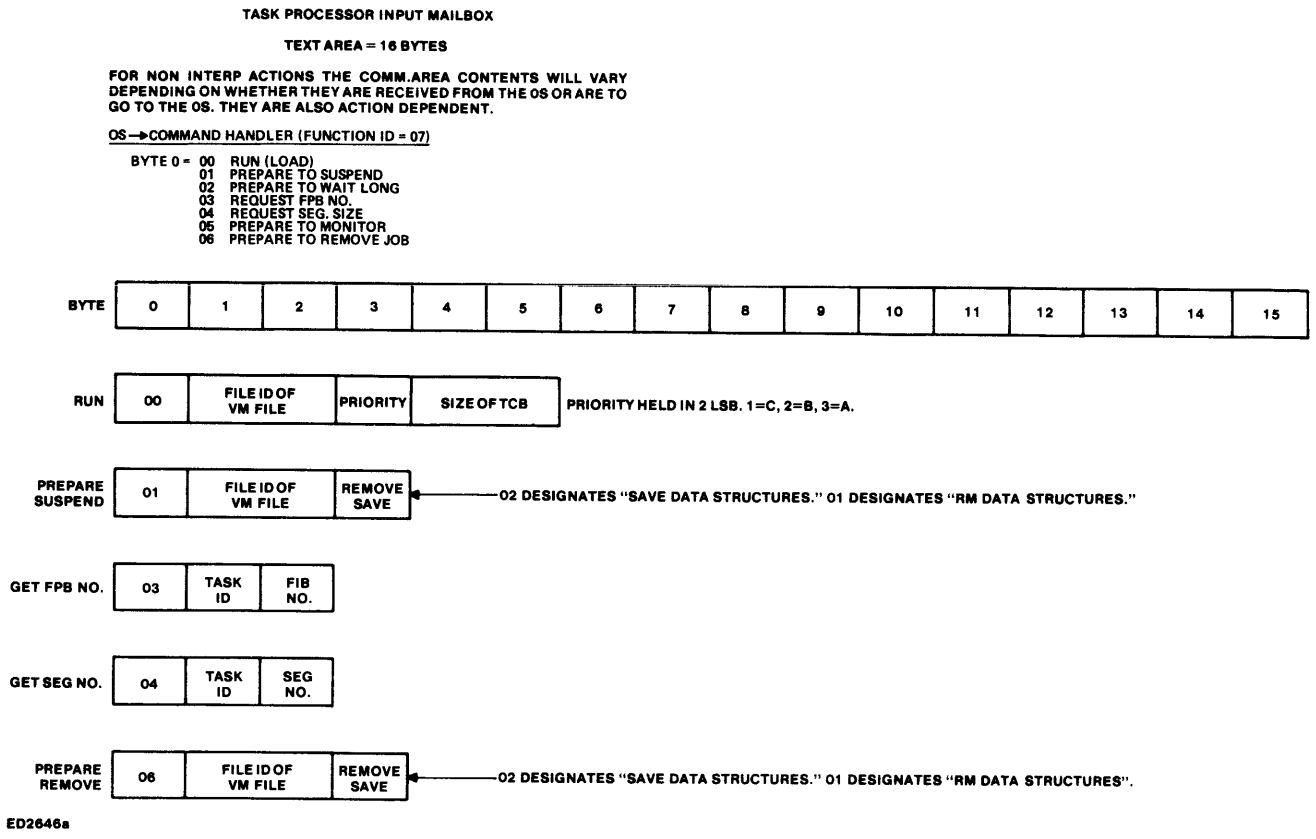


Figure 10-1. TP Input Mailbox Format for Function 07

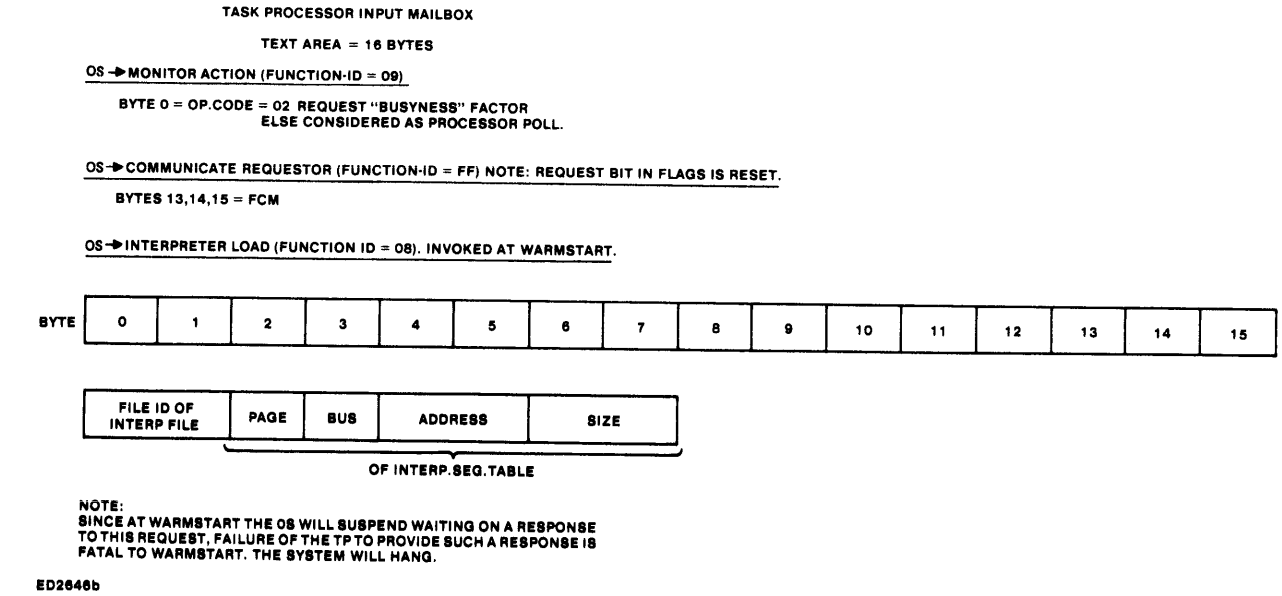


Figure 10-2. TP Input Mailbox Format for Functions 08, 09, FF

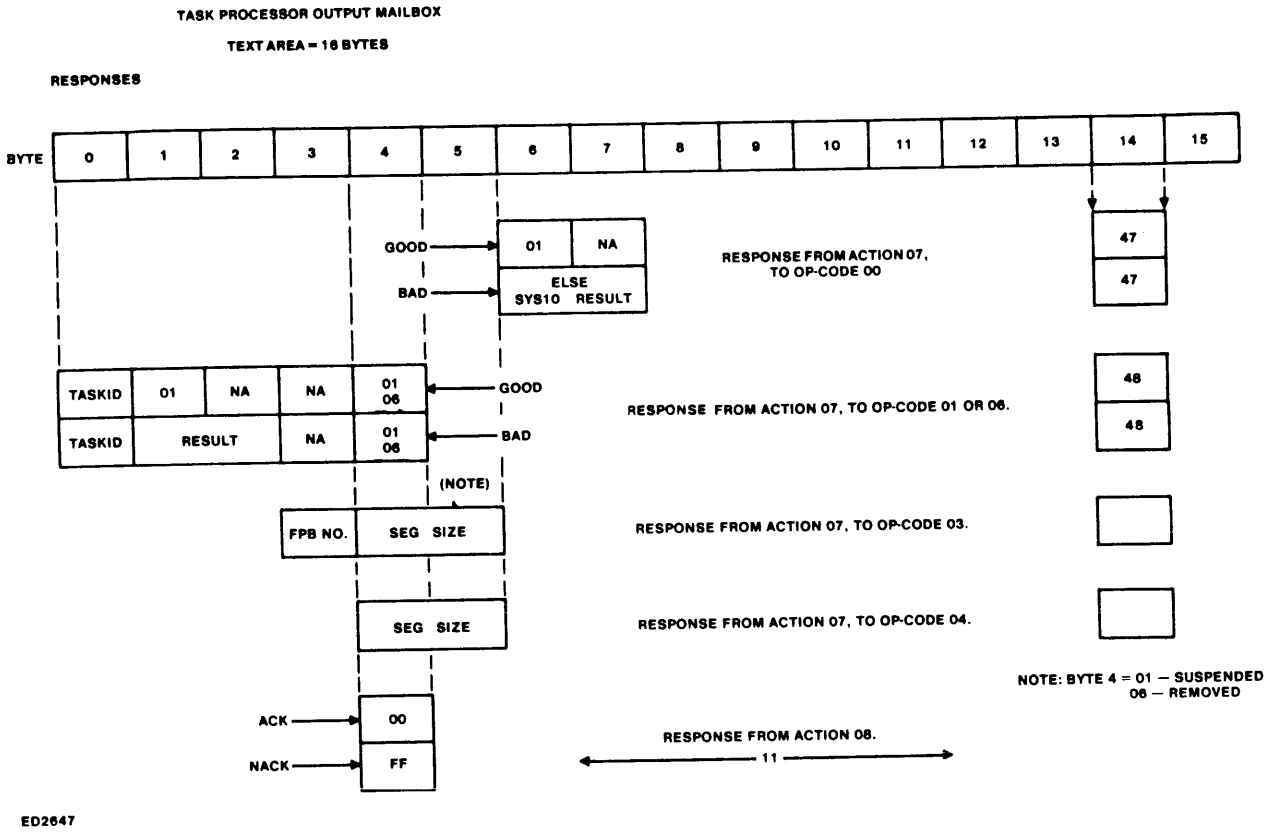


Figure 10-3. TP Output Mailbox Format

action ID of the requesting action, the request bit will not be set, and the IMB text area will contain the response.

IMB REQUEST PROCESSOR — This field contains the bus address of the processor originating the request. (Not used by task processor.)

IMB RESULT PROCESSOR — This field contains the bus address of the processor this message went to. (Not used by task processor.)

IMB RESULT FUNCTION — This field contains the ID of the action that is to receive the result. When the function field points to a TP action, the ID is an index into the status block address array. Refer to the "SB address array" description in a later subsection.

IMB SYSTEM MIX NUMBER — This field contains the mix number of the number of the job identified in the OS processor Mix Table. Information passed is associated with this job.

IMB AWAITING MAILBOX Not used.

IMB FLAGS The bit assignments of the flags are as follows:

```
MSB  7  - Request
     6  - Unused DCP
     5  - TP error
     4  - Rejected by TP
     3  - Accepted by OS
     2  - Given to OS
     1  - Accepted by TP
LSB  0  - Given to TP
```

The following flag values indicate the most common transaction sequence and may help to identify the status of the text area:

@08@ Mailbox owned by OS
@81@ Mailbox contains data for TP
@84@ TP has accepted the input message

@06@ may also be seen in the flags field. This indicates a successful data move by the TP.

The meanings of the flags without the "8" bit are also as above except that the text contains a response from the OS rather than a request.

OUTPUT MAILBOX (OMB) TEXT
Text same as for Input Mailbox.

OMB REQUEST FUNCTION This field is used to indicate the type of information contained in the mailbox text area. Values here represent action-IDs in the OS.

When the flags request bit is set, the following flag values are used:

@15@	REQ-SYSMESSAGE	TP request to display information on the ODT/Remote SPO.
@30@	REQ.IO	TP request to bring a structure into its memory.
@47@	REQ.RUNNING	TP informing the OS that a job has been loaded or restarted, and should now be marked as executing.
@48@	REQ.REMOVE	TP informing OS that a task has been suspended or terminated as requested.
@4D@	REQ.THRASHING	TP informing OS that a thrashing condition exists.

@52@	REQ.JOB.ERROR	TP informing OS that an error has been detected. Probably DS-DP condition.
@8B@	REQ.MONITOR	Invokes DUMMY action in OS to confirm DP still alive.
@FF@	REQ.COMMUNICATE	First byte of text = communicate.

OMB request processor, OMB result processor, OMB result function, OMB awaiting mailbox, system mix number are the same as IMB.

OMB FLAGS

The bit assignments of the flags are as follows:

```
MSB  7 : Request
     6 : Unused DCP
     5 : TP error
     4 : Rejected by OS
     3 : Accepted by OS
     2 : Given to OS
     1 : Accepted by TP
LSB  0 : OS acknowledgment
```

The following flag values indicate the most common transaction sequence, and may help to identify the status of the next area.

@02@	Mailbox owned by TP
@84@	Mailbox contains data for OS
@81@	OS has accepted the mailbox request
@02@	Mailbox owned by TP

The meanings of the flags without the "8" bit are the same as above except that the text contains a response from the TP rather than a request.

SIGNAL OUT LOCATION

This is the location where the task processor sets a flag to alert the OS to check the mailbox.

ERROR OFFSET

This field points to the location in the task processor code where the error was detected.

INTERPRETER SAVE AREA

```
INTERP KDUMP,
INTERP SINTFLAG,
INTERP SICOUNT,
INTERP LDUMP,
INTERP TCBPTR,
INTERP TCBPAGE,
INTERP RTRNDUMP,
INTERP JDUMP,
INTERP COMMPTR
```

These are fields used by the interpreter to save off information during interrupt handling, etc., to enable control to be reinitiated at the correct place.

POINTER FIELDS

OS BUS	This field contains the bus address of the operating system interrupt handling to enable control to be reinitiated at the correct place.
Current Task	This field contains the ID of the task which has control of the task processor.
OS Request Task	This field points to the task requested by the OS.
Terminate Tasks Task	This field points to the task which is terminating.
Current Message	This field points to the mailbox containing a message for the current task.
Terminate Queue	This field points to the tasks waiting to be terminated.
Task References	This is a list of user jobs or TP-created tasks.
First Job	This field points to the first user task.
Ready Queue	This field points to the next job which will obtain control of the task processor.
Please Yield	The OS is requesting the task processor to give up control of the current task. Possible values are: @FF@ = OS requesting processor to yield control @00@ = No request from OS
Allocates Called Allocates Idled Allocates Measure	These fields are used to determine processor activity.
Page Reference	This field points to the pages of the task processor.
Last Page	This field indicates the number of pages on the task processor.
Blocks Lock	@00@ = Unlocked; @FF@ = Either a TCB, PCB, or ICB is in the process of being created/deleted. While in a locked state no one may make or delete from this block.
ICB First Page	This one byte field points to the page of the first ICB.
ICB First Location	This field points to the location on the page of the first ICB.

ICB Last Page	This field points to the last ICB on the list on a particular page.
ICB Last Location	This field points to the location on the page of the last ICB.
	The PCB and TCB pointer fields, which are the next eight entries, are the same as the ICB pointer fields above.
Mailout Lock	A value of @FF@ indicates a locked condition. This lock is used to control access to the mailbox which can be used by only one task at a time.
OS Processor	This field contains the bus address of the OS processor.
Task Processor	This field contains the bus address of the task processor.
H Bugspace Location Bugspace Location Num Debug Text	These fields will always have a value of zero. They are regarded as reserved fields.
Task Run Space	This field contains the address of the next task to be loaded.
No Task Space	A value of @FF@ means there is no space to load a task on this processor.
Segments Task Kind	These are compiler-related fields.

SPACE ANALYSIS

This area reflects the state of each page of memory in the task processor. Since hardware memory addressing is not continuous across pages, memory management of each page is effected independently. For each page of memory present, the following fields are listed:

Page Number	This field indicates the memory page in the task processor.
First Space, Last Space	These fields contain the addresses of the lowest and highest spaces in the page. The spaces of a page are linked forward and backward in memory address order. Therefore, the spaces of a page can be searched in either direction.
Lock	A value of @FF@ means locked. Since each page has its own lock, memory management in several pages can proceed concurrently. This lock is

	used to control memory allocation on each page of memory in the task processor. This lock can be used by only one task at a time.
Task Waiting	This field contains the task record addresses of tasks waiting on a lock.
First Available	This field contains the address of the lowest addressed available space. All the available spaces in a page are linked together in memory address order.
GS Count	This is the Get Segment Count that is used in detection of thrashing.
Blocks	The number of PCBs, TCBs, or ICBs contained on that page.

Each page of memory in the Space Analysis section contains a table of the space allocation of the memory on that page. It contains the column headings NEXT, LAST, SIZE, STATE, and USED. The information under these headings is explained as follows:

NEXT	The NEXT field is the forward link to adjacent spaces. LAST (preceding entry) + SIZE = NEXT (unless NEXT = NULL).
LAST	The LAST field is the backward link to adjacent spaces. An example of how to read the NEXT and LAST fields is given in figure 10-4.
SIZE	This field is the size of the space in bytes, inclusive of the space record itself.
STATE	The state of the space can be either available, overlayable, or non-overlayable.
USED	This field is a usage count for S-SEGMENTS. @07@ = used, @00@ = available.

PAGE NUMBER	02
FIRST SPACE	0000
LAST SPACE	FFE8
LOCK	00
TASK WAITING	0000
FIRST AVAILABLE	1476
GS COUNT	08
BLOCKS	0001
ARCOUNT	A4FF

NEXT	LAST	SIZE	STATE		USED
****	****	****	*****		****
000B	0000	0C0B	FF	NON OVERLAYABLE	07
0167	0000	015C	80	OVERLAYABLE	07
02A9	000B	0142	80	OVERLAYABLE	07
0497	0167	01EE	80	OVERLAYABLE	07
0776	02A9	02DF	80	OVERLAYABLE	07
09A0	0497	022A	80	OVERLAYABLE	07
0FB9	0776	0619	80	OVERLAYABLE	07
113A	09A0	0181	80	OVERLAYABLE	07
146E[1]	0FB9	0334	80	OVERLAYABLE	07
FF3A	113A	EACC	00	AVAILABLE	00
FFE8	146E[2]	00AE	FF	NON OVERLAYABLE	07
0000	FF3A	0008	FF	NON OVERLAYABLE	07

A sample space allocation table is shown above. Assume that you wish to locate the last area of AVAILABLE space in the Hex dump. The line of information relating to this area has been outlined so that it will stand out. This piece of memory, of size EACC (in Hex), is located between two addresses of 146E. These addresses are the NEXT of the preceding line(1) and the LAST of the line which follows(2). This indicates that the available entry is located at 146E in this page.

Figure 10-4. Example of How to Read NEXT and LAST Fields

BLOCKS

There are three kinds of blocks: Interpreter Control Blocks (ICBs), Program Control Blocks (PCBs), and Task Control Blocks (TCBs). There is an analysis section for each kind of block in the dump.

ICB ANALYSIS

This area contains interpreter information. There is an ICB ANALYSIS section for each interpreter in a task processor. A description of the fields as they appear in this section is as follows:

ICB Page	This field indicates the page where the ICB is located.
ICB Location	This field indicates the location of the ICB on the page.
Space Next, Space Previous	These are forward and backward memory links in the space record.
Space Size	This field indicates the size of the ICB in bytes.
Space State	The space state can be either available, overlayable, or non-overlayable.
Space Used	@07@ indicates recently "used", and @00@ indicates "available."

The BLOCK HEADER is comprised of the next six fields appearing in the dump. This area precedes all ICB, PCB, and TCB records. These fields provide a description of the block, and are linked to previous blocks of similar type on a task processor page. The fields are named Next Page, Next Loc, Previous Page, Previous Loc, Kind, and ID.

The following fields appear in the dump after the Block Header:

ICBS Users	This field contains the number of users for a particular ICB in a task processor.
ICBS Flags	Not used.
ICBS CST BASE	Contains the first byte of the ICB file descriptor. This is the starting location of the ICB file.
ICBS CST Limit	Contains the last byte of the ICB file descriptor. This is the ending location of the ICB file.

PCB ANALYSIS

This area contains information regarding the programs assigned to a specific task processor. The fields in this section are the same as those of the ICB Analysis area.

TCB ANALYSIS

The standard Block Header fields are followed by a TCB descriptor and a segment table.

TCB Descriptor

Priority	This field indicates the external priority of the user task, where @01@ = C @02@ = B @03@ = A
I Seg	Interpreter segment number.
I Off	Offset into the interpreter segment.
CST Size	Size of the code segment table.
CST Sector	Offset of the code segment table.
IST Size	Size of the interpreter segment table.
IST Sector	Offset of the interpreter segment table.
DST Offset	Offset into the data segment.
PPA Size, PPA Sector	Size and offset of program parameter area. For COBOL, this area is the COP table. Not used by MPL.
CH Return	Not used.
Data Stack Pointer thru Control Stack Limit	Job-related parameters.
Flags	The flag bits are: MSB 7 & 6 Used for thrashing detection 5 Not maintained 4 DS or DP condition 3 Suspend 2 Save data structures 1 Remove LSB 0 Long wait
Sintmask	Not maintained.
FCM	
SPSA Seg and SPSA Off	S-program start address segment number and displacement.

TCB SEGMENT TABLE

The layout is as in a PCB. The disk address field points to the offset within the appropriate disk file from where the data segment is to be loaded. This could be the VM file, the code file, or a "zero fill," depending on the current state of the program. FIB segments are replaced by FPB segments. The FPB is CMS standard. These are listed after the TCB segment table. The STE number is an Analyzer-computed field to cross reference the FPB to the segment table.

JOB QUEUE

Contains status information for each task.

OS REQUEST TASK

Contains information similar to an output mailbox that has not been serviced yet.

TERM TASKS TASK

Contains information regarding tasks waiting to be terminated by job management.

READY QUEUE, TERMINATING QUEUE, LOCKED QUEUE

The task processor has three queues: Ready Queue, Blocks Lock (waiting), and Terminate Queue. A pointer to the head of these lists is in the pointer fields. Each queue is in prioirity order, highest to lowest.

COMPUTING AMOUNT OF SPACE OWNED BY A TASK

A task uses memory for a task record, PCB, TCB, and present segments. The size of these areas can be found as follows:

Task record size	= size of field of preceding "space" record.
PCB size	= size of PCB record plus present segments. Use the "PCB_ID" field in task to find PCB record. Size field of preceding "space" record contains the size of the PCB (including block header, space record, and segment descriptors).
Segment size	= size specified in the segment descriptor if the absent bit (bit 6 in flag) is not set.

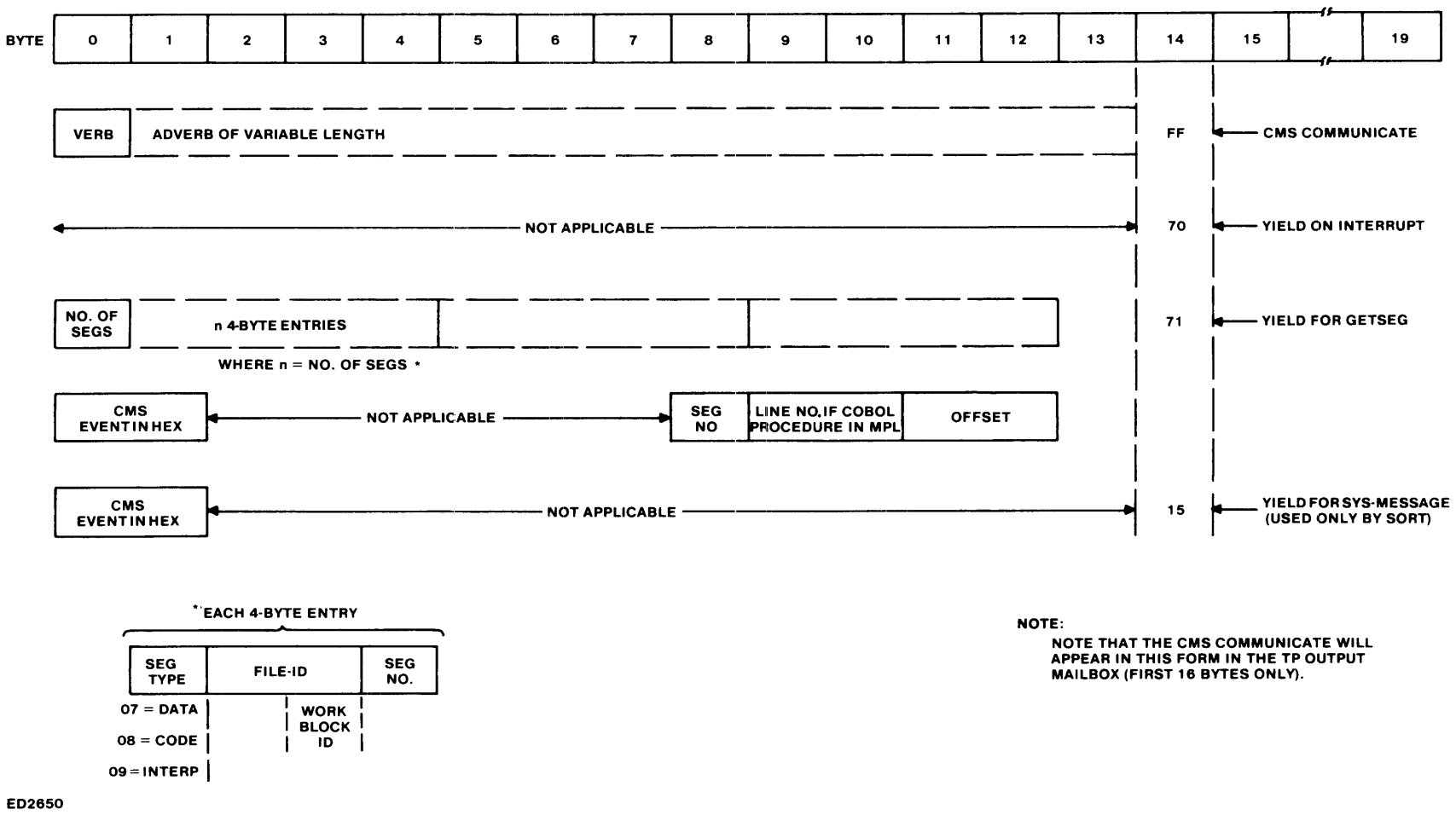


Figure 10-5. Task Record Text

SECTION 11
DISK MANAGEMENT

INTRODUCTION

Details concerning the B 900/CP 9500 disk subsystem are provided in this section of the dump. The configuration and physical status of all disk drives attached to the system can be ascertained, as well as commands handled by the disk drives. The following structures are contained in this section:

TASK STACK - information relating to internal disk processor's operation

State Save Area

Pointers to structures within the disk processor's memory

PORT TABLE - configuration of I/O channels

COMMAND TABLE- I/O command data

DRIVE TABLE - containing actual drive status

INITIAL DUMP ANALYSIS

If the disk processor encounters a hardware error causing the processor to freeze, the Hex display indicates "F00000" on the lower banks of lights. Two common causes of a frozen processor are the system disk going "NOT READY," and the disk processor detecting a memory parity error. In these situations, a ROM/PROM dump needs to be taken, followed by an examination of the Disk Management section of the dump.

The important areas to interrogate in this section of the dump are the DRIVE TABLE, the COMMAND TABLE, and the PORT TABLE. The DRIVE TABLE entries provide the result status of the last I/O command issued on all drives attached to the system. Any I/O errors are indicated in these entries. The last I/O performed on each disk drive is identified in the COMMAND TABLE. Lastly, the PORT TABLE defines the existing disk drive configurations.

If a failure has been detected within the disk processor, then the State Save Area needs to be studied. The error number identifies the cause of the failure. If this field has a value of @00@, then a hardware error has occurred. Further analysis of the State Save Area as outlined in section 8 of this manual will assist in identifying the type of failure detected.

Following is a detailed description of the tables found in the Disk Management Section.

FIRST FIELDS

The first items encountered in the Disk Management section of a dump are explained as follows and shown in figure 11-1.

OS PAGE	This is a 9-byte field, of which the first two bytes (reversed) are the bus and page of the OS Processor, with the IC enable bit set.
OS INTERFACE AREA	This is the base address of the disk/OS processor interface area within the OS processor page 0.

```
********************************
****** DISK MANAGEMENT *******
********************************

                    OS PAGE  0021 3A00 0000 0000 00
          OS INTERFACE AREA  E366

            STATE SAVE AREA
  STATE SAVE AREA ADDRESS  3A00
              ERROR NUMBER  FF
                 ACTION ID  FF
               ACTION NAME  FF                    END OF LIST EXCEEDED
                        M1  0000
                        WR  0000
                        M2  0000
                       B32  0000
                       MAX  0000
                       MXA  0000
                       MXB  0000
                      B1FL  0000
                        B0  00
                       REQ  00
                        XY  0000 0000 0000 0000
                         J  0000
                         K  0000
                         L  0000
                        AD  00
             TOP OF STACK  0000
            TOP OF XSTACK  0000
             2ND ON STACK  0000
            2ND ON XSTACK  0000
             3RD ON STACK  0000
            3RD ON XSTACK  0000
          BOTTOM OF STACK  0000
         BOTTOM OF XSTACK  0000
       IC - ERROR STATUS   00
     IC - STATUS WORD 1    00
     IC - STATUS WORD 2    00
     IC - STATUS WORD 3    00
     IC - STATUS WORD 4    FF
     IC - STATUS WORD 5    FF
  IC - FIRST ERROR REG     FFFF FFFF

                  DP SIZE  6D9A
              CURR TASKS   04
              MAX DRIVES   10
            TASK INFO PTR  ED56
           TASK STACK PTR  3D57
            MGR STACK PTR  B502
             PORT TBL PTR  FD69
              CMD TBL PTR  AD6C
            DRIVE TBL PTR  CD6D
             RD AVAIL PTR  CC95
             RD INUSE PTR  FFFF
```

Figure 11-1. Sample Disk Management Section Initial Entries

STATE SAVE AREA	Refer to the State Save Area in the OS Processor section, section 8.
DP SIZE	This field is a 2-byte entry, reversed, that contains the number of bytes of code and data in use by the DSCP. Additionally, if the DFCM is connected to the system, this field is the base of a 5,760-byte buffer used to transfer data to/from the DFCM.
CURR TASKS	This field is a 1-byte entry that represents the drive that was told to execute a command on the last pass through the drive scheduler. It does not represent the drive that might be currently executing a command.
MAX DRIVES	This field is a 1-byte entry that represents the maximum number of drives that the DSCP will allow for the amount of memory present. In 32K, @08@ drives are allowed; in 64K, @10@ drives are allowed.
TASK INFO PTR thru DRIVE TBL PTR	These fields are 2-byte reversed entries pointing to the named structure.
RD AVAIL PTR	This field is a 2-byte reversed entry pointing to the start of the available result descriptor space. @FFFF@ means that no space is available.
RD INUSE PTR	This field is a 2-byte reversed entry pointing to the start of the result descriptor space that currently contains valid result data. @FFFF@ means that no space is in use.

TASK INFORMATION

The Task Information Table consists of up to 16 5-byte entries. If the DSCP memory contains 32KB on page 0, there will be eight entries; otherwise, the Task Information Table will contain 16 entries. Each entry of this table corresponds to a drive: entry 0, drive 0; entry 1, drive 1; etc.

Each entry of the table has the format:

Byte	Length	Meaning/Contents
1	1	Status: @00@ = Runable @01@ = Waiting I/O @FF@ = Idle
2	2	Task stack base address, two bytes, reversed
4	2	"L" register value used as a pointer into the task stack

STATUS NOTES:

Runable implies that the command will be given control on some future pass through the task scheduler. At this point, the command may be complete, and the OS still needs to be given the result, or the command may not even be started. For a CMS type command (see Commands), it may mean that an intermediate I/O operation is complete.

Waiting I/O implies that the task is executing a command on the drive, and that the command is currently in progress.

Idle is self-explanatory.

Refer to figure 11-2 for a sample Task Information Entry and an explanation of the decoding.

```
TASK INFORMATION        56C3    FFEB 5600 00
                        56C8    FF17 581B 58
                        56CD    FF43 5900 D0
                        56D2    FF6F 5A79 5A
                        56D7    019B 5B9F 5B
                        56DC    FFC7 5C00 00
                        56E1    FFF3 5D00 00
                        56E6    FF1F 5F00 00
```

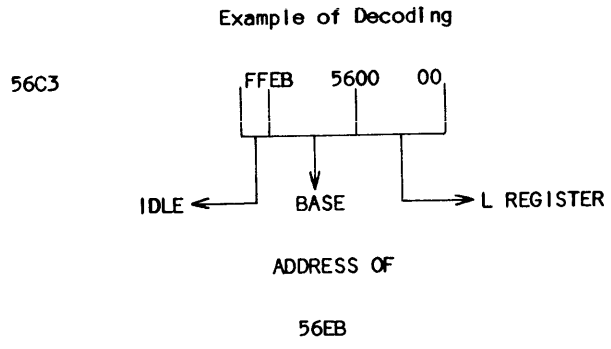


Figure 11-2. Sample Task Information Entry

TASK STACKS

The task stacks are 300-byte entries, one per drive, and represent the call and return stack for the task. Each entry on the stack is a 2-byte reversed address which is used to transfer control to the routine pointed to by the address. Additionally, temporary data is put onto the stack for convenience purposes. The task information "L" register value points to the next available 2-byte task stack entry. All task stack entries are "POP"ed and "PUSH"ed in a first-in-last-out (FILO) manner.

TASK STACK DATA FOR DISK PACK DEVICES

The disk pack devices use the task stack for additional command parameters and data. The portion of a task stack used by a disk pack device is pointed to by the Drive Table for the disk pack device (see Drive Table, Disk Pack Devices, MY.L Field). The format of the Task Stack Pack Area is shown below. (Note: add the value of the MY.L field from the Drive Table to the "byte" offsets below to arrive at the memory address of the data.)

Byte	Length	Meaning/Contents
0	2	Buffer fragment length
2	2	Buffer fragment page
4	2	Buffer fragment base
6	2	Total transfer size. Note: Drive Table buffer size = number of bytes transferred so far
8	1	Block level transfer - HSTLTA block types currently expected. @80@ = NSBR block expected @40@ = NSBW block expected @20@ = TD block expected, but not required
9	1	Block level transfer - HSTLTA block types allowed after a transfer delay occurs @80@ = NSBR block allowed @40@ = NSBW block allowed
10	2	Address of Read Transfer routine for this command
12	2	Address of Write Transfer routine for this command
14	4	Search Command Hit Sector address
18	2	Search Command Hit Sector offset
20	14	A copy of the original values of bytes 0 through 13 to be restored to bytes 0 through 13 if a retry is required

MANAGER STACK

This is a 50-byte entry that contains run-time pointers and data for the Scheduler and Task Manager routines. The format of this structure is the same as for the task stacks. However, only the manager routines use it.

PORT TABLE

The port table entries each describe a port on the DSM processor. The entries consist of 43-byte elements, with the following format:

Byte	Length	Meaning/Contents
0	1	Controller ID
1	1	The 0-relative drive number of the first drive on the port
2	1	The number of drives on the port. Note the following relationship: first drive plus number of drives = the first drive on the next port
3	40	Controller-dependent parameters

CONTROLLER IDENTIFIERS

The following values may be found in the Controller ID field of the Port Table:

ID	CONTROLLER TYPE	DRIVE TYPE	PORT ASSIGNMENT
@52@	Standard disk Host Controller	In drive Table	Any port 3 thru 7
@60@	1 drive 1MB BSMD	B9489-1/11	Any port 3 thru 7
@62@	1 drive cartridge disk 100 TPI	B9480-11	Any port 3 thru 7
@64@	1 drive cartridge disk 200 TPI	B9481-11	Any port 3 thru 7
@66@	1 drive cartridge disk 100 TPI	B9480-21	Any port 3 thru 7
@68@	2 drive 1MB BSMD	B9489-12	Any port 3 thru 7
@6A@	2 drive cartridge disk	B9480-12	Any port 3 thru 7
@6C@	2 drive cartridge disk	B9481-12	Any port 3 thru 7
@6E@	2 drive cartridge disk	B9480-12	Any port 3 thru 7
@6F@	2011 fixed	B9493-18/37	Any port 3 thru 7
@81@	DMTR	N/A	Port 2
@88@	DFCM	N/A	Port 7
@8A@	DISK PACK DDP	IN DRIVE TABLE	Any port 3 thru 7
@FE@	OS PORT	N/A	Port 0
@FF@	N/A	N/A	Port 1, Not used
@FF@	N/A	N/A	Any port that does not contain a device

CONTROLLER DEPENDENT PARAMETERS

Some controller firmware routines keep port-dependent information in the Port Table. The firmware for host and disk pack devices use this area of the Port Table. The formats are as follows:

Host Device Port Table

If the Port Table ID is @52@, the following information applies:

Byte	Length	Meaning/Contents
4	1	Current hardware command string length
5	1	Current hardware command
6	1	Current port relative drive
7	4	Current sector address
11	2	Current I/O length

If byte 5 is Search (@A8@) then:

7	1	Search NULL character
8	1	Search type (greater than, less than, etc.)
9	4	Search starting sector address
13	1	Tag offset within first sector
14	1	Search tag length
15	2	Search I/O length

For all commands:

17	1	Current port relative drive
18	1	Port busy flag (FF = true, 00 = false)
19	1	Prepare-To-Load (PTL) flag
20	1	Power-On-Reset (POR) in progress flag
21	2	Base address of the first drive table for the first drive on this port

Disk Pack Port Table

If the first byte of the port table entry is @8A@, the following information applies:

Byte	Length	Meaning/Contents
4	1	"AD" register setting for this port
5	1	Complement of byte 4
6	1	Current active port relative drive
7	2	Address of status byte for this port (B9387 status is reflected as "PMA")
9	1	Subsystem Poll In Progress flag
10	8	Drive states (drive 0 to 7), 1 byte each @00@ = Idle @01@ = Queued, command ready to send @FF@ = In progress, B9387 has the command

COMMAND TABLE

This area is used to store the command information fetched from the OS processor. Each entry of the table is 18 bytes, and is indexed by drive number. All disk processor commands and layout are described in the following paragraphs. A sample command table section is shown in figure 11-3.

```
                                                Opcode
COMMAND TABLE ENTRY  0        61A3    0000 0000 0000 0000 0000 0000 0000 0000 0000
COMMAND TABLE ENTRY  1        61B5    0000 0000 0000 0000 0000 0000 0000 0000 0000
COMMAND TABLE ENTRY  2        61C7    0000 0000 0000 0000 0000 0000 0000 0000 0000
COMMAND TABLE ENTRY  3        61D9    0000 0000 0000 0000 0000 0000 0000 0000 0000
COMMAND TABLE ENTRY  4        61EB    0100 8000 0300 0000 8000 8AC1 0200 04FF FFFF
COMMAND TABLE ENTRY  5        61FD    0000 0000 0000 0000 0000 0000 0000 0000 0000
COMMAND TABLE ENTRY  6        620F    0000 0000 0000 0000 0000 0000 0000 0000 0000
COMMAND TABLE ENTRY  7        6221    0000 0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 11-3. Sample Command Table Section

COMMAND RESULTS

The results of a command are relayed to the OS processor in three ways: 1) through Command Status (see Drive Table), 2) through result descriptors (see Drive Table), and 3) through result parameters.

The semantics of Command Status are explained in the Drive Table description.

If a disk device error occurs, the error information is maintained in a result descriptor structure, and the unique result descriptor identifier is returned in the last three command parameters, bytes 16 through 18.

The result parameters are command dependent, and are returned to the OS processor in bytes 10 through 15 of the command parameters.

GENERAL COMMAND LAYOUT

There are five types of system commands, as follows:

1. Read.
2. Write.
3. Search.
4. Initialize.
5. Lock/unlock door.

Some commands have a specific format, but in general, they have the following format. Any deviations from the general format are identified in the individual command descriptions.

Byte	Length	Meaning/Contents
1	1	Opcode
2	2	Buffer fragment size
4	2	Buffer fragment page
6	2	Buffer fragment base
8	2	Total buffer size: note the following relation: If buffer fragment size equals total buffer size, then the transfer is not to/from fragmented buffers, but contiguous memory.
10	3	Starting sector address.

COMMAND TYPES

There are two types of commands: System and CMS. System commands require no preliminary processing before being executed by disk drive controller firmware. CMS commands are commands that manipulate disk directory structures, and may be made up of many system commands (READ, WRITE, and SEARCH). The commands are as follows:

Opcode	Type	Description
@00@	SYS	Read Data
@01@	SYS	Write Data
@02@	SYS	Write Data and Check data or data parity
@03@	SYS	Search less than tag argument
@04@	SYS	Search less than tag argument and read hit sector
@05@	SYS	Search Greater than tag argument
@06@	SYS	Search greater than tag argument and read hit sector
@07@	SYS	Search equal to tag argument
@08@	SYS	Search equal to tag argument and read hit sector
@09@	SYS	Search less than or equal to tag argument
@0A@	SYS	Search less than or equal to tag argument and read hit sector
@0B@	SYS	Search greater than or equal to tag argument
@0C@	SYS	Search greater than or equal to tag argument and read hit sector
@0D@	SYS	Initialize
@0E@	SYS	Lock Door
@0F@	SYS	Unlock Door
@10@	SYS	Read Sector Relocation Map
@11@	SYS	Read Device Statistics
@12@	SYS	Clear Device Statistics
@13@	SYS	Read Result Descriptor
@14@	CMS	Read Disk File Header (DFH)
@15@	CMS	Write DFH
@16@	CMS	Locate pseudo pack
@17@	CMS	Locate file
@18@	CMS	Locate file and read DFH
@19@	CMS	Create temporary entry
@1A@	CMS	Allocate file area
@1B@	CMS	Crunch file
@1C@	CMS	Purge file
@26@	CMS	Power off disk
@27@	CMS	AVR disk with label read
@28@	SYS	Initialize disk pack cylinder data
@29@	SYS	Verify disk pack cylinder
@2A@	SYS	Relocate disk pack bad sector

READ

READ transfers data from a disk device to memory. READ uses the standard command format.

WRITE

WRITE transfers data from memory to a disk device. WRITE uses the standard command format.

WRITE WITH DATA OR DATA PARITY CHECK

This command performs a WRITE operation, and then reads the data back from disk to check it. If the device is a 3/6 BSMD, then the data parity is checked. If the device is a cartridge disk device, then the data read back is compared to the data written. If the device is not one of these types, the command defaults to a normal write. This command uses the standard format.

SEARCH

The SEARCH commands search the disk for the occurrence of a tag with a given relationship to a tag argument. Search uses the standard command format with the following changes:

Byte	Length	Meaning/Contents
8	2	Number of sectors to search; two bytes, reversed
10	3	Starting sector address
13	1	Key length
14	1	Key offset within tag argument
15	1	Tag argument length

Search returns the following result parameters:

Byte	Length	Meaning/Contents
10	3	Hit Sector address
13	1	Offset of Hit Tag within the Hit Sector

INITIALIZE

INITIALIZE is valid for cartridge type devices and disk pack devices.

DISK CARTRIDGE DEVICES Initialize for disk pack devices causes sector addresses to be written to one cylinder of the device, and the sector data fields written with a data pattern. The command parameters are as follows:

Byte	Length	Meaning/Contents
8	2	Number of tracks to initialize
10	3	Starting sector address of starting track

DISK PACK DEVICES Initialize for disk pack devices causes sector addresses to be written to one cylinder of the device, and the sector data fields written with a data pattern. The command parameters are as follows:

Byte	Length	Meaning/Contents
10	3	Starting sector address of cylinder
13	2	Data pattern, 2 bytes reversed

LOCK DOOR

This command causes the door of a 3/6 BSMD device to be locked. All command parameters are ignored except the OPCODE.

UNLOCK DOOR

This command causes the door of a 3/6 BSMD device to be unlocked. All command parameters are ignored except the OPCODE.

READ RELOCATION MAP

This command transfers the relocation map of host (SDHC) type devices to the buffer. This command uses the standard format.

READ STATISTICS

This command transfers the statistics of host (SDHC) type devices to the buffer. This command uses the standard format.

CLEAR STATISTICS

This command clears the statistics of host (SDHC) type devices. All command parameters are ignored except for the OPCODE.

READ RESULT DESCRIPTOR

This command finds the correct result descriptor and transfers it to the buffer. The command parameter format is as follows:

Byte	Length	Meaning/Contents
10	1	Result descriptor identifier

READ DFH

This command returns an adjusted DFH to the OS processor buffer. The The command parameter format is as follows:

Byte	Length	Meaning/Contents
10	2	DFH directory index

WRITE DFH

WRITE DFH writes the actual header to disk when given a directory index and an adjusted DFH in the buffer. The command parameters are the same as for READ DFH.

LOCATE PSEUDO PACK

LOCATE PSEUDO PACK searches the PPIT on a given unit for a 7-byte pack ID and returns the pack tag and logical unit number. The command parameters are the standard format. The result parameters are as follows:

Byte	Length	Meaning/Contents
10	1	Pack tag
11	1	Logical unit number

LOCATE FILE

LOCATE FILE searches a Directory Name list for the 13-byte File ID in the buffer and returns the directory index of the DFH if the ID is found. The command parameters are the standard format. The result parameters are as follows:

Byte	Length	Meaning/Contents
10	2	DFH directory index

LOCATE HEADER

LOCATE HEADER is identical to LOCATE FILE, except that a successful LOCATE will return an adjusted DFH. The command parameters are the standard format. The result parameters are the same as LOCATE FILE.

CREATE TEMPORARY ENTRY

CREATE TEMPORARY ENTRY searches the Name List for an available entry, updates the corresponding header entry, and returns the header and directory index if the disk had an available entry to use. The command parameters are the standard format. The result parameters are the same as LOCATE FILE.

ALLOCATE AREA

ALLOCATE AREA allocates an area for the specified DFH. The command parameters are as follows:

Byte	Length	Meaning/Contents
10	2	Directory index
12	2	Area size in sectors
14	1	Area number

The result parameters are as follows:

Byte	Length	Meaning/Contents
10	3	Starting sector address of area just allocated

CRUNCH FILE

CRUNCH FILE alters the specified DFH to return excess space to the non-file directory. If the last area is not on the specified unit, only the file pointers are updated. The command parameters are the same as for READ DFH.

PURGE ENTRY

PURGE ENTRY returns the specified directory entry to the available list, and returns all disk space allocated to the file on this disk to the non-file directory. The command parameters are the same as for READ DFH.

POWER OFF DISK

This command causes the disk to be logically powered off. The following steps are taken:

1. The door is unlocked (if applicable).
2. The Label Integrity flag is reset.
3. The tables for the drive are initialized.

All command parameters are ignored except the OPCODE.

AVR DISK

This command performs the automatic volume recognition functions, such as label verification, setting the integrity flag, and reconstruction. The command parameters are the standard format.

INITIALIZE CYLINDER DATA FIELDS

This command is valid for disk pack devices only. This command writes the sector data fields of the specified cylinder with the specified data pattern. The command parameters are the same as for the INITIALIZE command for disk pack devices. The result parameters are as follows:

Byte	Length	Meaning/Contents
13	3	Sector address where the first/worst error occurred (if an error occurred)

VERIFY CYLINDER

This command is valid for disk pack devices only. This command verifies the cylinder data fields and error protection codes of the specified cylinder. The command and result parameters are the same as for INITIALIZE CYLINDER DATA FIELDS.

RELOCATE BAD SECTOR

This command is valid for disk pack devices only. This command causes the specified sector to be relocated to a spare on the cylinder. The command parameters are as follows:

Byte	Length	Meaning/Contents
10	3	Sector to be relocated
13	2	Data pattern

DRIVE TABLE

The drive table holds all data relative to its corresponding drive. The drive table is divided into two parts: the general data area and the device-dependent data area. The general data area is standard among all drives. The device-dependent area is different for each of the three types of drives: disk cartridge, host, and pack. Each of the areas is described as follows.

DRIVE TABLE LAYOUT

Byte	Length	Meaning/Contents
1	1	Drive Status (Refer to Table 11-1)
2	1	Drive Identifier (Refer to Table 11-2)
3	1	The Port for this Drive
4	1	DFMC Participating Flag
5	2	Command Entrance Routine Address
7	1	Command Status (Refer to Table 11-3)
8	1	Monitor Status (Refer to Table 11-3)
9	1	Command Retry Count
10	1	Result Descriptor Index
11	2	Result Descriptor Size
13	15	Command Parameters (Refer to Command Table)
28	210	Device-Dependent Data Area

Table 11-1. Drive Status

Bit	Set Meaning	Reset Meaning	Notes
0	Ready	Not Ready	
1	Write Inhibit	Not Write Inhibit	
2	Temporarily Available (TNA)	Not TNA	*
3	Disk In Danger	Disk Not In Danger	
4	Disk Expiring	Disk Not Expiring	
5	Mandatory Interrupt	Not Mandatory Interrupt	
6	Spare		
7	Spare		

*TNA applies to 3/6 devices only, and implies that the door of the other drive is open.

Bits 2 through 5 are for SDHC devices only.

Table 11-2. Drive Identifiers

Identifier	Drive Type	Drive Name	Controller Type
@53@	B9493-20	20 MB 211 FD	SDHC, @52@
@54@	B9493-80	80 MB 211 FD	SDHC, @52@
@56@	B9489-21/23	3/6 BSMD 2DR	SDHC, @52@
@57@	B9493-40	40 MB 211 FD	SDHC, @52@
@60@	B9489-1/11	1MB BSMD 1DR	SAME
@62@	B9480-11	100TPI CART 1 DR	SAME
@64@	B9481-11	200TPI CART 1 DR	SAME
@66@	B9481-21	100TPI CART 1 DR	SAME
@68@	B9489-12	1MB BSMD 2 DR	SAME
@6A@	B9480-12	CART 2 DRIVES	SAME
@6C@	B9489-12	CART 2 DRIVES	SAME
@6E@	B9480-22	CART 2 DRIVES	SAME
@6F@	B9483-18/37	2011 FD	SAME
@F8@		205 DPD	DPIOC, @8A@
@78@		206 DPD	DPIOC, @8A@
@B8@		207 FD	DPIOC, @8A@

The drive IDs declare the type of drive present, and sometimes match the port controller ID (See Port Table).

Table 11-3. Command Status Semantics

COMMAND STATUS

Value	Meaning
00	Controller error - internal to DP
01	Command successful
02	Command incomplete, legal unsuccessful result (i.e., SEARCH tag not found)
03	Device error - command aborted
04	Spare
05	I/O descriptor error

MONITOR STATUS

Value	Meaning
00	No error
01	Seek timeout
02	Head off cylinder (Seek error)
03	Sequence error - internal to DP
04	Disk data/address parity error
05	Disk address not found
06	Illegal address in I/O descriptor
07	Status word error - internal to DP
08	Data error in COMPARE
09	Attempt to write to write-inhibited disk

DEVICE DEPENDENT DATA AREA

The byte offsets given here are a continuation of the offsets found in the general data area description. The values of some of the fields in the Drive Tables can be found in the discussion of the specific device type.

Disk Cartridge

The following table defines the remaps of the device-dependent data area for disk cartridge devices.

Byte	Length	Meaning/Contents
28	2	Pointer to the Next Phase routine
30	2	Total buffer size
32	2	Fragment size
34	2	Fragment page
36	2	Fragment base
38	2	Least significant two bytes of the current sector address
40	1	One Buffer flag
41	1	Most significant byte of the current sector address
42	1	Last Transfer flag
43	1	Current retry count
44	1	Dual 2011 flag
45	1	Sectors per track
46	1	Compliment of the "AD" register setting
47	1	Current controller status
48	1	Opcode
49	1	Drive address on the port
50	1	Current Search size
51	1	Current Search flags
52	1	Current Search offset
53	1	Current Search Hit Sector address
54	2	Current cylinder number

HOST DEVICES

Byte	Length	Meaning/Contents
28	180	Search Hit Sector buffer
208	4	Search Hit address
212	1	Search Hit offset
213	1	Hardware opcode
214	2	Fragment size
216	2	Fragment page
218	2	Fragment base
220	2	Address of Command Complete routine for this command
222	2	Address of Transfer routine for this command
224	2	Transfer length
226	1	211 type device flag
227	1	One buffer flag
228	1	Address of drive on port

Byte	Length	Meaning/Contents
229	1	SDHC last status
230	1	Last Class 2 Status Interrupt
231	1	Error mask
232	4	Hardware status, DC status bytes 1, 2 and 3
236	1	Port Polling flag
237	1	Drive status for Polling process

If the Search Hit Sector buffer contains the device attribute record (DAR), then the following fields can be found in the DAR:

Byte	Length	Meaning/Contents
28	1	DAR Check Byte 1
29	1	DAR Check Byte 2
30	1	DAR Confidence Test results
31	7	Drive ID bytes
38	12	DAR for Drive 0
50	12	DAR for Drive 1

DISK PACK DATA AREA

Byte	Length	Meaning/Contents
28	1	@80@, Command Block Type byte
29	1	@0C@, Command Block Count byte
30	1	Current B9387 Opcode
31	1	Current B9387-Relative Unit
32	2	Current command variants
34	4	Current I/O length
38	4	Current sector address
42	1	@81@, Write Block Type Byte
43	1	Write Block count byte
44	1	Search Null Character
45	1	Search Relationship
46	1	Search Tag Offset within sector
47	1	@00@, not used
48	2	Reason for last transfer delay
50	1	@00@, not used
51	1	Last short result descriptor info byte
52	2	Value of "L" register at command start
54	2	MY.L, the current value of the "L" register
56	2	Pointer to the Port Table of the port to which this drive is connected
58	2	Address of the current result descriptor for this drive
60	1	Result descriptor identifier

RESULT DESCRIPTORS

There can be two result descriptor lists: AVAILABLE and INUSE. The AVAILABLE list contains all result descriptors not containing data, and the INUSE list has all the result descriptors containing data when the dump was taken.

The INUSE list can be expected to have all result descriptors that are currently being used to store error information. Typically, this list is empty, as data from INUSE result descriptors would have been fetched by the OS processor, and the descriptors would have been returned to the AVAILABLE list.

A result descriptor is divided into two parts: control information and data. The control information is used to maintain the result descriptor lists, and the data portion contains the drive-related failure data.

CONTROL INFORMATION

The format of the CONTROL INFORMATION is as follows:

Name	Byte	Length	Meaning/Contents
LINK	1	2	Link to the next RD on this list; @FFFF@ implies that the end of the list has been reached.
INDEX	3	1	Index. This is the unique RD identifier.
DRIVE	4	1	Drive. If the RD is available, this field is @FF@; otherwise, it is the drive number of the drive whose data is in the RD.

RD DATA

The RD Data portion of a result descriptor contains the information describing the error, and the format of the data depends upon the type of device that owns the RD.

Disk Pack RD

Byte	Length	Meaning/Contents
1	1	B9387 Opcode
2	1	B9387-relative unit number, 0-7
3	2	B9387 command descriptor variants
5	4	Starting sector address
9	4	I/O operation length
13	1	Last byte read from the B900 disk pack DDP
14	1	Last byte expected to have been read from the DDP - see note 4
15	1	DDP status after last byte read from DDP
16	1	B900 error number - see note 1
17	1	B9387 longitudinal parity byte
18	1	Host-computed longitudinal parity byte
19	1	Last CIO command sent - actual value
20	1	Status response to last CIO command

21	1	Direction of transfer - see note 2
22	1	B9387 long result descriptor byte 0
23	1	B9387 long result descriptor byte 1
24	44	B9387 long result descriptor bytes 3 - 45

Notes:

1. 01 = Bad vertical parity error on B900 input transfer
   02 = Bad longitudinal parity error on B900 input transfer
   03 = DDP 2.0-second timer expired before B9387 response
   04 = Host block count different from B9387 block count on input or output transfer
   05 = Software 2.3-second counter expired while waiting for B9387 response; DDP timer failed
   06 = Incorrect Block Type on B900 input transfer
   07 = Count byte did not correspond to type byte on B900 input transfer
2. 00 = From B900 DDP to B900 DSM processor
   01 = From B900 DSM processor to B900 DDP to DSM processor
3. This is the actual Hex value sent to the DDP
4. If the error number is 06 (Bad Type Byte), then this field has the following meanings:

   01 = Expected TD, long RD, short RD, READ OK, READ retry, READ correction, or READ uncorrectable
   02 = Expected TD, long RD or short RD
   03 = Expected long RD or short RD
   04 = Expected TD during SEARCH

NULL RESULT DESCRIPTORS

A result descriptor is termed a "Null RD" when it contains error data, but the error occurred before a command was sent to the drive. An example of this is a mandatory interrupt on a 3/6 device, which occurs when the BSMD is placed in the drive and the door shut. In these cases, the status of the device changes from Not Ready to Not Ready-Mandatory Interrupt. In order for the OS to display the mandatory interrupt message with device status, the DSCP places the status in a NULL RD, and changes the device status in the OS. The OS sees the status change, and issues the READ RESULT command to the DSCP, with the unique result descriptor identifier parameter as @FF@ (hence, NULL). This causes the DSCP to return the correct RD.

A NULL RD in the INUSE List is indicated by an index value greater than @E0@.

DISK CARTRIDGE DRIVES DATA CONSTANTS

Disk cartridge constants are:

Field Name	Values	Meanings
Search Flags	01	Dual search required
	02	Pattern search
Controller Status	00	Seek mode
	01	Command mode
	02	Idle

HOST DATA CONSTANTS

Host data constants are:

Field Name	Values	Meanings
Current Hardware CMD	@A1@	Read
	@CE@	Write
	@A8@	Search
	@A2@	Read search result
	@AD@	Read relocation map
	@AE@	Read statistics
	@A4@	Read DAR
	@A7@	Read device status
	@FD@	Abort command
	@FB@	Clear Statistics
	@F2@	Set Write Protect
	@F4@	Reset Write Protect
	@C1@	Unlock door
	@C2@	Lock door
	@C4@	Reset MTR
Controller Status	@00@	Idle
	@01@	Pending
	@02@	Busy
Error Mask	@85@	Read After Write Check in progress, drive Write inhibited
	@81@	Read After Write Check failed, Drive Write enabled
B9387 Opcode	@01@	Read
	@02@	Write
	@03@	Search
	@10@	Initialize
	@11@	Verify
	@12@	Initialize Data Field
	@14@	Relocate Sector
	@20@	Test operation (soft poll)
	@80@	Initialize C/W parameters
Command Variants	@8000@	Wait for Seek to complete
	@0000@	Do not wait for seek to complete
Search Relations	@01@	Less than
	@02@	Equal to
	@03@	Less than or equal to
	@04@	Greater than
	@05@	Not equal to
	@06@	Greater than or equal to
Task States	@00@	Idle
	@FF@	In process
	@01@	Queued
Blocks Expected	@80@	NSBR expected
	@40@	NSBW expected
	@20@	TD expected

RECOMMENDED PROBLEM SOLVING PROCEDURES

The Disk Processor can be involved in two types of system problems. These are a system hang and a @F00000@ clearstart. What follows are some recommended procedures for determining if the problem is due to a hardware failure or a software failure.

System Hang

If the system hangs such that the time-of-day is still displayed, and if the processor displays continue to cycle, and the SPO transmits but does not respond, then perform the following steps:

1. Take a ROM dump and run ROMCONVERT and the SYSANALYZER on the converted dump.
2. Determine which disk is the system disk from the Configuration Table portion of the OS dump.
3. Find the Drive Status field in the Drive Table for the system disk.
4. If bit 1 of the drive status is set, meaning "Ready," then the hang is due to some other factor.
5. If bit 1 of the drive status is reset, meaning "Not Ready," then the hang might be due to a not-ready system disk.
6. If the system disk is a SDHC 211 type device, then the Last Status field of the Drive Table contains the status that caused the disk to go not ready.
7. Additionally, check the Result Descriptor Inuse list to see if an RD belongs to the system disk. If so, then this RD might contain more information about what caused the system disk to go not ready.
8. If the system disk is a B9387 DPDC disk pack device, then find the Port Table entry for the DPIOC.
9. Locate the address of the Status Byte field in the Port Table.
10. Find the byte having this address in the disk processor Hex dump.
11. If this byte is @00@, the system disk was made Not Ready because the B9387 went off line for one of three reasons:
    1) Someone placed the B9387 off line at the B9387.
    2) A cabling problem exists that caused the off-line signal to change states (noise, cable fell off).
    3) The B9387 controlware detected a temperature warning condition and will power, or has powered, itself off.

F00000 Clearstart

A @F00000@ clearstart may be caused by three things: 1) a real hardware failure; 2) corrupted buffer memory buffer addresses; 3) corrupted I/O descriptor buffer address. To determine which of these things occurred, perform the following steps:

1. Take a ROM dump, run ROMCONVERT and SYSANALYZER on the dump.
2. Locate the Disk Processor Save State Area.

3. If the IC error and status information indicate that a real error occurred, and MAX, MXA and MXB contain valid bus/page data, then a real error did occur at the IC First Error Reg address.

4. Otherwise, if MAX, MXA, MXB do not contain valid bus/page data, then the problem is a software error.

SECTION 12
BUFFER MEMORY

INTRODUCTION

Buffer memory is a global system resource used for file buffers, system I/O descriptors, data comm buffers, and queues. The amount of buffer memory is specified in the dependent portion of the SYSCONFIG file.

This section first discusses the technique for determining the amount of buffer space being allocated. This is followed by a detailed examination of the Buffer Memory section of a memory dump.

The system uses the top page (or more) of OS memory as buffer memory. The system reserves buffer memory for the data comm buffers at WARMSTART. The amount reserved is that specified in SYSCONFIG. The data comm buffers are analyzed and listed as part of the data comm section. No reference is made to them here.

The remainder is used as OS buffer memory. This must be at least 2048 (2K) bytes. Buffer memory can be a maximum of 1024 KB.

Great care must be taken when assigning buffer memory. The performance of the B 900/CP 9500 can be seriously degraded if inappropriate values are specified in the buffer memory portion of SYSCONFIG. If too much buffer memory is allocated for user programs, the MCP is forced to do more segment overlaying. This is a result of not enough available MCP memory. If an insufficient amount of buffer memory is allotted for user programs, the programs will "wait on buffer space."

I/O BUFFER SPACE CALCULATION

There are three ways to determine the amount of required buffer memory. The first method is to experiment with different buffer allocations until an optimum system performance is attained. This is an inconvenient procedure and possibly inaccurate.

The second method is to load the system with a mix of jobs that will reflect a maximum throughput condition required by the user. At this point, a memory dump should be taken using the "GT MD" intrinsic. Examine the "AVAILABLE COUNT" field found at the beginning of the buffer memory section in the dump. Multiply this value by 195 (the size of each buffer plus descriptors). Remember to byte-reverse and convert to decimal the value in the field before multiplying. This result will indicate the amount of available memory. Subtract the amount of available memory from the amount specified in SYSCONFIG. This is the amount of required buffer memory. It is advisable to add a safety factor of 2K to this amount.

The third and most accurate method requires knowing the block sizes and the number of buffers needed for each opened file on the system. With this information, use the following procedure:

I/O BUFFER SPACE CALCULATION

250 plus the following equation for Opened Files

1. Number of buffers divided by 4 rounded up, plus

2. Block size divided by 180 rounded up, times the number of buffers, plus

3. 1 if not indexed, or
   3 if indexed file, times

4. 195

Example #1 - Indexed file, 6 buffers, block size = 270

1. 6/4 = 1.5, r = 2

2. 270/180 = 1.5, r = 2, 2 x 6 = 12

3. 3

4. 2 + 12 + 3 = 17, 17 x 195 = 3315 bytes

```
 3315
+250
-----
 3565 I/O Buffer space for 1 opened file
```

Example #2 - Non-Indexed file, 1 buffer, block size = 180

1. 1/4 = .25, r = 1

2. 180/180 = 1, r = 1

3. 1

4. 1 + 1 + 1 = 3, 3 x 195 = 285 bytes

```
  285 file
+250 base
-----
 535 bytes I/O space for 1 open file
```

Example #3 - Both 1 and 2 opened

```
 3315 for file 1
  285 for file 3
+250 base
-----
 3850 bytes needed
```

OS buffer memory is organized in sections. Currently each section is 180 bytes in size (plus control fields). These sections are linked together in four categories:

File table entries
Key file table entries
System I/O descriptors (Not yet implemented)
Available sections

These are controlled by a set of pointers located in the first 34 bytes of the buffer memory as follows:

```
DA_BUF_POINTER POINTER      4 BYTES
AVAILABLE_HEAD              4 BYTES
AVAILABLE_TAIL              4 BYTES
FILE_TABLE_HEAD             4 BYTES
FILE_TABLE_TAIL             4 BYTES
KEY_FILE_TABLE_HEAD         4 BYTES
KEY_FILE_TABLE_TAIL         4 BYTES
SYSIO_DESC_HEAD             4 BYTES
SYSIO_DESC_TAIL             4 BYTES
AVAILABLE_COUNT             2 BYTES
```

A sample of these entries in an actual memory dump is shown in figure 12-1.

```
*****************************
******* BUFFER MEMORY ********
*****************************

          DA BUF POINTER POINTER  0301 1B60
                  AVAILABLE HEAD  0301 6169
                  AVAILABLE TAIL  0301 1107
                 FILE TABLE HEAD  0301 9265
                 FILE TABLE TAIL  0301 0A84
            KEY FILE TABLE HEAD   FFFF FFFF
            KEY FILE TABLE TAIL   FFFF FFFF
                    SYS IO HEAD   0301 3D60
                    SYS IO TAIL   0301 3D60
                AVAILABLE COUNT   5F00
```

Figure 12-1. Sample Initial Entry in Buffer Memory Section of a System Dump

All but the Available Count are full system addresses, laid out as follows:

```
7      0  7        0  7        0  15       8
[      ]  [        ]  [        ]  [        ]
  Page        Bus             Offset
```

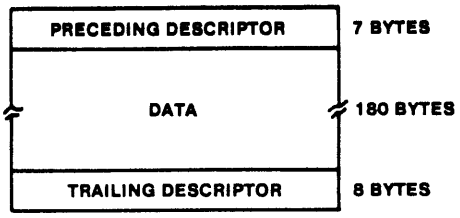
This layout is due to the byte reversal of the NBDS.

The exact usage of these fields is as follows:

DA_BUF_POINTER POINTER	Contains the address of the start of OS Buffer memory.
AVAILABLE HEAD	Contains the address of the first 180-byte section in the chain of those sections that are available for use.
AVAILABLE TAIL	Contains the address of the last available section.
FILE TABLE HEAD	Contains the address of the first file table in the file table chain.
FILE TABLE TAIL	Contains the address of the last file table in the file table chain.
KEY FILE TABLE HEAD	Contains the address of the first key file table in the key file table chain.
KEY FILE TABLE TAIL	Contains the address of the last key file table in the key file table chain.
SYS IO HEAD	Not maintained.
SYS IO TAIL	Not maintained.
AVAILABLE COUNT	Contains the number of sections that are currently available.

SECTION LAYOUT

Each section includes three parts: a preceding memory descriptor, the actual data, and a trailing descriptor:



ET2652

There are two links, a preceding link and a trailing link, because PIMOVE expects the link to immediately precede the section and to point to the next link, while the I/O device controllers require the size of the next section and the link to follow the section, and the link to point to the first byte of the next section (for performance reasons). A sample section of a memory dump containing this information is shown in figure 12-2.

```
**********************          PRECEEDING      TRAILING
FILE TABLE ENTRY (FTE)    FLAGS SIZE  LINK      SIZE  LINK       ADDRESS      FTE
**********************    ***** **********    **********       *******      ***
                          01    B400  03301A4CB B400  03015566   03019265     0000
```

Figure 12-2. Memory Dump Sample Showing Section Layout

PRECEDING MEMORY DESCRIPTOR FORMAT

The preceding memory descriptor contains three fields:

FLAGS : 1 byte
SIZE : 2 bytes
LINK : 4 bytes

These fields are used as follows:

FLAGS — Defines the current use of this section:

- @80@ : Available section
- @01@ : File table section
- @02@ : File table extension section
- @03@ : Key file table section
- @04@ : Key file table extension
- @05@ : SYSIO descriptor table section
- @10@ : Buffer section

SIZE — This entry contains the size of the section that follows. In this implementation this field always equals 180. This field is implemented so that if the size of buffer sections change, code does not have to change.

LINK — This entry contains the address of the preceding descriptor for the next structure of a similar type, except for buffer sections where this link points to the next link in the preceding descriptor of the next section of the buffer. The degree of linking varies according to the section type and use. All sections reserved for buffers are not linked together. Only those sections associated with a particular buffer are linked. Conversely, all file table sections, key table sections, available sections, and system I/O descriptor sections are linked together. All file table extension sections belonging to the same file table are linked together. The link entry of the last section of any chain contains @FFFFFFFF@.

TRAILING MEMORY DESCRIPTOR FORMAT

The trailing descriptor consists of three fields. These are as follows:

```
SIZE:              2 bytes
LINK:              4 bytes
FTE (VALIDATION):  2 bytes
```

SIZE

This entry contains the size of the next section of the file table or the particular buffer. If the preceding section is the last section of the file table or the buffer, this entry contains @FFFF@.

LINK

For buffer sections, the link entry in the trailing memory descriptor is used to address the first byte of the next section of that particular buffer. The link entry of the last section for that particular buffer contains @FFFFFFFF@.

For file table sections, this entry points to the preceding descriptor of the file table extension section associated with the file table. If no file table extension section exists for this file table, this entry contains @FFFFFFFF@.

For file table extension sections, this entry addresses the preceding descriptor of the next file table extension section for the file table. This entry in the last file table extension section for the file table contains @FFFFFFFF@.

For all other section types, this entry addresses the preceding descriptor of the next section of the same type. The last section of a particular type contains @FFFFFFFF@.

ANALYZER LISTING

The analyzer lists the descriptor fields in a standard manner for all sections. This layout consists of one line of titled information comprising the flag field, the preceding descriptor information, the trailing descriptor information, the address of the section, and the validation field.

FILE TABLE STRUCTURE

File table information is kept in two structures, the file table and the file table extension.

A file table entry is built when a CMS data file is first opened and it is maintained until a close reduces the user count to zero. Each file table entry (FTE) has one or more file table extensions linked to it. Each extension has four entries (FTEE). Each entry is a buffer descriptor that points to and controls a data buffer.

The exact layouts are described in the following subsections.

FILE TABLE ENTRY

A file table entry section contains the following information:

DA_FILE_TAG	This 4-byte field is an internal identifier for the file.
COMMON_EOF	This 3-byte field holds the shared EOF for all users.
TOTAL_DESC	This 1-byte field holds the number of descriptors allocated.
NUM_USERS	The number of users of this file.
SHARED_RECORD_SIZE	This 2-byte field holds the common record size for shared file users.
SHARED_BLOCK_SIZE	This 2-byte field holds the common block size for shared file users.
AREA INFORMATION	There are 16 entries, one for each possible area (disk only). Each entry includes the following:

UNIT	1 byte	The DA_CT index of the device
ADDRESS	3 bytes	Disk address
SIZE	3 bytes	Number of blocks in this area

FILLER	55 bytes reserved for future use.

FILE TABLE EXTENSION ENTRY

Each file table extension contains up to four buffer descriptors. The FTEE includes:

Map	These four 1-byte fields contain the task IDs of the owners of these descriptors. If a descriptor is unassigned, the corresponding byte will be @FF@. The first byte refers to the first descriptor, etc.
Buffer Descriptors	These are 40 bytes each, and are described in detail in the next subsection.

A sample from an actual memory dump is shown in figure 12-3.

BUFFER DESCRIPTOR FORMAT

Each buffer descriptor is laid out and analyzed as follows:

Descriptor	This number is developed by the analyzer.
Block ID	This 3-byte field contains the block number in the file of the information in this buffer.
FIB ID	The first byte is the FIB ID, the second the owner.
Lock	Contains the ID of the owner of the block when it is locked. If it is @FFFF@, the block is not locked.
Flags	These define the state of the buffer as follows: MSB 7 : Buffer allocated 6 : Move outstanding 5 : Shared buffer 4 : Buffered ahead 3 : Waiting 2 : Updated 1 : Printer backup LSB 0 : Record outstanding
Unit No	Unit number for this I/O (index into DA_CT).
Queue Status	Status of descriptor in queue (see PHT section).
Action ID	Action ID of requestor if waiting.
Return Status	The I/O status (see PHT section).
Forward Link, Backward Link	Links for queueing. These are I/O descriptors. They are two-way links for possible future use. Presently, everything is linked in the forward direction.
Opcode	Operation requested. (See PHT section for normal operations, and Command Table section for specialized disk operations).
Buffer Address	Address of first section of buffer: Size 2 bytes Bus & page 2 bytes Address 2 bytes
Size Parameter	Total size of buffer in bytes.
Device Dependent Parameters	Device-dependent parameters, 9 bytes.

Note that the address in I/O.DESC = @FFFF@ if no buffer exists.

```
**********************************
FILE TABLE EXTENSION ENTRY (FTEE)
**********************************
                                        PRECEEDING      TRAILING
                                  FLAGS SIZE  LINK      SIZE  LINK      ADDRESS   MAP       FTE  FTEE
                                  ***** ********** ********** ********  ***       ***  ****
                                  02    8400  FFFFFFFF 8400  FFFFFFFF  03015566  000610FF  0000 0000

                        DESCRIPTOR  01
                          BLOCK ID  8300 00
                            FIB ID  1205
                              LOCK  FFFF
                             FLAGS  80
                       UNIT NUMBER  04
                      QUEUE STATUS  03                  COMPLETE AND DEQUEUED
                         ACTION ID  0F
                     RETURN STATUS  0100 00             COMMAND COMPLETE     NO ERROR
                      FORWARD LINK  0301 5377
                     BACKWARD LINK  0001 CA4A
                           OP CODE  00                  READ
                    BUFFER ADDRESS  8400 0301 1F67
                    SIZE PARAMETER  8400
          DEVICE DEPENDENT PARAMETERS  001D 3302 0004 0000 00
                                  ALLOCATED           UNCONDITIONAL I/O   NOT SHARED          NOT BUFFERED AHEAD
                                  NOT IN USE          NOT UPDATED         NOT PRINT BACKUP    NOT OUTSTANDING

                                        PRECEEDING      TRAILING
                                  FLAGS SIZE  LINK      SIZE  LINK      ADDRESS
                                  ***** **** ****       **** ****      ********
                                  10    8400  FFFFFFFF 8400  FFFFFFFF  03011F67

                               DATA  092E 3448 6984 90A9 9A4C 414E 4755 4147 4520 5459 5045 2048 4153 2054 4F4F 204D
                                     414E 5920 4348 4152 4143 5445 5253 454E 4420 FFC0 FFAU 2056 4552 4220 454E 5452
                                     4945 532J 5550 4441 5445 4454 4849 5320 4954 454D 2043 414E 464F 5420 4245 2054
                                     5241 4E5S 4C41 5445 444D 4953 2D4D 4154 4348 2046 4F55 4E44 2049 4E20 4E45 5720
                                     5445 5354 5245 434F 5645 5259 2043 4F4D 504C 4554 4544 FFC0 2057 4F52 4B20 4649
                                     4C45 204C 4F41 4445 4453 5420 FFC0 494E 5320 2D00
```

Figure 12-3. Sample File Table Extension Entry

KEY FILE TABLE FORMAT

KFILETAG	4 bytes	Unique identifier for file
NUM_USERS	1 byte	# of users with keyfile open
MAX_KEYFILE_SIZE	3 bytes	Write EOF
KEY_SIZE	1 byte	Actual size of key
KEY_ENTRY_SIZE	1 byte	Key size reserved (5,13,21,29)
AREA_INFO		Array [16] of record
Address	3 bytes	
Length	2 bytes	
INDX_REG_EXISTS	1 byte	Set if index region exists
OVRFLW_REG_EXISTS	1 byte	Set if overflow region exists
RUF_TBLE	10 bytes	See KEYFILE_REG_FRMT below
INDX_REG	10 bytes	See KEYFILE_REG_FRMT below
OVRFLW_REG	10 bytes	See KEYFILE_REG_FRMT below
HIGHEST RECORD		Read EOF
Area	1 byte	
Sector	3 bytes	
Offset	1 byte	(offset in the sector)
Key		Array [28] of byte
		(last key value end)
SPARE	25 bytes	
KEYFILE_REG_FRMT		Record
Starting		
Sector	3 bytes	
Size	3 bytes	Number of sectors in starting area
Starting Area	1 byte	
Distance to		
End Area	3 bytes	Starting sector in starting area

KEY FILE EXTENSION FORMAT

Keyfile		Array [7] of record
User	2 bytes	FIB_ID and TID
Updated	1 byte	Flag for this user
Lock	1 byte	TID of next user (begin of list)
End List	1 byte	Next available slot in lock list
List Waiting Lock		Array [7] of record
Action ID	1 byte	
Take Lock	1 byte	
Spare	143 bytes	

SYSIO DESCRIPTOR TABLE FORMAT

Not Maintained in 3.04 releases.

AVAILABLE TABLE

Lists available buffer space.

APPENDIX A
OS ERROR INDICATORS

BANKS C AND D

The display on these control panel banks is an indication of processor business (normally a communicate count).

CLEARSTARTS

When the operating system encounters an error condition, it performs an error-handling routine which terminates with a clearstart. This causes a 3-byte field to be displayed on the hex display and on the ODT if its data path is still runable. The three bytes displayed are:

BANK E -Error number as described below.

BANK F -Index into the AM.ACTION.LIST for the action in error.

BANK G -Id of the action in error.

BANK E -CLEARSTART NUMBERS

@00@	DEADLOCK ERROR - An action is attempting to get on a queue behind its parent which is waiting for it to finish.
@01@	NO NUMBERS ERROR - A normal action was invoked when there were already one hundred normal actions in the mix, and Activity Management could not assign it a unique ID; only 100 ACTION numbers are available.
@02@	BAD ACTION ID ERROR - A non-existing action was invoked.
@03@	BAD SEG ID ERROR - An attempt to access a non-existing segment (work block, link block, data block, or code block) occurred.
@04@	TOO MANY WB ERROR - An action tried to get a work block when it already had four.
@05@	CALL FROM FS ERROR - An action called an Activity Management routine from force state which is not allowed in force state.
@06@	CALL BY CONT ERROR - A continuous action called an Activity Management routine which is not allowed for continuous actions.
@07@	CALL NOT NORM ERROR - An action attempted to get on a queue when it was already on a queue.
@08@	WB AT FINISH ERROR - An action attempted to finish when it still owned a work block.

@09@ BAD POST ERROR - An action called "POST-RESULT" passing an action ID of a non-existing action, or of one which is not awaiting a post.

@0A@ BAD DEQUEUE ERROR - An action attempted to dequeue from a queue where it was not present.

@0B@ BAD STOP INT ERROR - An action called "STOP-INTERRUPTS" when it was not proccessing interrupts (not in force state).

@0C@ BAD AWAIT INT ERROR - An action called "AWAIT-INTERRUPT," specifying a channel which was being serviced already by some other action.

@0D@ ZERO DIVIDE ERROR - An action called "ABS.DIVIDE" passing a zero divisor.

@0E@ PARITY ERROR - A parity error was detected by the OS processor. The State Save of a 2-MHz machine will have the address of the offending memory location listed under "IC - First Error Register" in the dump.

@0F@ BAD CLEAR ERROR - Software sent micro execution to address zero on any page.

@10@ PREV MARKED ERROR - An attempt was made to mark a work or link block with the state which it was already in (jelled, frozen, unjelled, or unfrozen).

@11@ MARK COUNT ERROR - An attempt was made to freeze or jell a code or data block which already had 15 outstanding freezes or jells against it.

@12@ MEM CORRUPT ERROR - A virtual memory descriptor has been corrupted.

@13@ BAD CHANNEL ERROR - A call to "AWAIT-INTERRUPT" or "STOP-INTERRUPTS" was made which specified an illegal channel number.

@14@ STK OVERFLOW ERROR - An action called Activity Management with its stack pointer pointing beyond its allocated space.

@15@ CONT VM ERROR - A continuous action required virtual memory for any reason.

@16@ BAD ENQUEUE ERROR - An action attempted to enqueue to a non-existing queue.

@17@ CB FREEZE ERROR - An attempt was made to freeze a Code Block.

@18@ VM REQUEST ERROR - The function which virtual memory

was requested to perform does not exist.

@19@ VM LOCATE ERROR - The Work Block or Link Block which virtual memory was requested to load in memory does not exist.

@1A@ VM IO ERROR - An I/O error was reported in an attempt to read/write to the MCP file.

The exact error and the sector in error can be found as follows:

Determine the action running in the OS (AM.ACTION.LIST). The action should be AM_VM. The comm field can be decoded as follows:

Drive number	Byte 1
MTR status	Bytes 2-3
Sector address	Bytes 4-6
Unused	Bytes 7-9
Retries	Byte 10

Drive number is as in disk processor, where 0 is A, 1 is B, ..., 7 is H. MTR status is defined in the Disk Management section of this manual (section 11).

The same information exists in the disk processor memory. If necessary, it can be decoded.

The normal recovery action is to utilize a different system disk to warmstart from, then take appropriate action with the failing disk, depending on the nature of the failure. FE assistance may be required.

@1B@ NO AVAIL MEM ERROR - Sufficient available memory cannot be obtained to satisfy the virtual memory request due to the number of jelled or frozen memory blocks.

@1C@ COMPACT VMFIL ERROR - Virtual memory was unable to relocate a work or link block within the Backing Store portion of the MCP file.

@1D@ OS VMFIL FULL ERROR - The Backing Store portion of the MCP file is completely filled.

@1E@ COMPACT MEM ERROR - Virtual memory was unable to create a block of contiguous available space of sufficient size to satisfy the request, due to the presence of frozen memory blocks.

@1F@ BAD BLK TYPE ERROR - An unrecognizable block type was encountered by virtual memory in the process of compacting memory.

@20@ WARMSTART ERROR - An error encountered by the Data

Access activity of the operating system during the startup process.

@21@ SYSCONFIG ERROR - An error was encountered while trying to access the SYSCONFIG file. An open or read error results in this error.

@22@ POWER OFF ERROR - A call to SYSTEM_ERROR was made to stop the system because a Power Off command was received rather than because of an error. No error display is made.

@23@ ATTACH BLOCK ERROR - An action attempted to attach a non-existent structure, attach a structure with one already attached, or locate a data block with one already attached.

@24@ DETACH BLOCK ERROR - An action attempted to detach a structure that was not attached.

@25@ INTRANSIT ERROR - Virtual memory cannot locate a section of memory it just allocated.

@26@ MEMORY LIMIT ERROR - The hardware detected that the processor attempted to address a memory (RAM or ROM) that is not in the system; often indicates a disk drive and/or media problem.

@27@ BOUNDARY ERROR - The hardware detected that the processor tried to access the IC memory map when it did not intend to, or that the processor tried to increment through address @FFFF@ on any page.

@28@ FILE TYPE ERROR - This error is generated when the Data Access activity, SYS_OPEN, is requested to open a file that is not a system file.

@29@ NO SIZE VMFL ERROR - This error is generated when an attempt is made to open a VM file which has a file size of zero.

@2A@ VMFL IS OPEN ERROR - This error is generated when an attempt is made to open a VM file which is already open.

@2B@ SYSFL OPEN ERROR - This error is generated when an attempt is made to open a system file which has a file type other than SYSTEM, LOG, CODE, or INTERPRETER, which is already open.

@2C@ NAMTAB OVER ERROR - This error is generated when an attempt is made to open a system file with the maximum number of files already opened for that file type.

@2D@ Reserved.

@2E@ ILLEGAL WRITE ERROR - This error is generated when an attempt is made to perform a SYS-IO write to a file that is not opened with a usage of read/write.

@2F@ INVALID OP ERROR - This error is generated when an attempt is made to perform a SYS-IO operation with an opcode other than read, write, or search.

@30@ Reserved.

@31@ CLOSE ADV ERROR - This error is generated when a SYS-CLOSE with purge of a file with type other than miscellaneous, list, NDLSYS, or VM-FILE is attempted.

@32@ NOT VMFILE ERROR - This error is generated when an attempt is made to add an area to a system file other than a VM file, or to dump to a file other than a VM file.

@33@ ADD ZERO SIZE ERROR - This error is generated when an attempt is made to add an area of size zero to a VM file.

@34@ QUEUE MAINT ERROR - This error is generated if an illegal descriptor was sent to a physical I/O module, or if an invalid unit number was detected.

@35@ IO DESC ERROR - This error is generated if the I/O handler sent an illegal descriptor.

@36@ Reserved.

@37@ DA PHT OVER ERROR - This error is generated when there are more than 20 devices on the system.

@38@ KEYFILE FRMT ERROR - An error was encountered in accessing the keyfile (of an index pair) that could only occur as a result of the keyfile being formatted incorrectly.

@39@ DA OUTCOME ERROR - This error is generated when an unrecognized output from the I/O subsystem is received.

@3A@ DA BUFF MEM ERROR - This error is generated when an internal data access error is detected during a buffer memory operation.

@3B@ DA SCL ERROR - This error is generated when an unrecognized output from a utility or the I/O subsystem procedure is received.

@3C@ Reserved

@3D@ Reserved.

@3E@ START OI ERROR - A problem has occurred in "START-OI" (operator interface); probably a bad outcome in trying to access a system file (SYSMCP, SYSCONFIG, SYSLANGUAGE).

@3F@ Reserved.

@40@ Reserved.

@41@ Reserrved.

@42@ Reserved.

@43@ Reserved.

@44@ Reserved.

@45@ Reserved.

@46@ JM START OPEN ERROR - An interpreter file open has failed.

@47@ JM START TP ERROR - The load of an interpreter to a task processor has failed.

@48@ JM START READ ERROR - A read from an interpreter file has failed.

@49@ JM OPEN PROG ERROR - Open of a program file has resulted in the return of an unexpected failure value.

@4A@ JM OPEN INTRP ERROR - Open of an interpreter file has resulted in an unexpected outcome value for failure.

@4B@ JM OPEN VM ERROR - Open of a VM file has resulted in an unexpected failure value being returned.

@4C@ JM PAUSE ERROR = The state flags for a job being paused do not make sense.

@4D@ JM TP ERROR - The task ID or mix number given by the invoker of the task procesor error action is illegal, or a request made of a task processor has been rejected by it.

@4E@ JM ACCESS ERROR - Information passed to the MIX-ACCESS action is invalid (in all cases, the task id).

@4F@ JM MX NAME ERROR - Information used to get a task name from DA (through DA.NAME.FIBID) gave a bad result. Error could be in JM or DA software.

@50@ Reserved.

@51@ JM ST ERROR - This error is detected by the STOP action

indicating that a specified job is in an invalid state.

@52@ JM SUPER ERROR - A startup time detected error indicating that the initial load of SYSSUPERUTIL was unsuccessful.

@53@ JM DSDPFLAGS ERROR - A job being DS-ed or DP-ed has the state flags in disarray.

@54@ JM EOJ ERROR - The end of job code has been entered for a job whose state flags make no sense.

@55@ JM TERMFLAG ERROR - The terminate flag was set with no terminate condition existing.

@56@ Reserved.

@57@ JM DCJOBREM ERROR - The DC job removed action has been called for a job whose flags say it is not terminating.

@58@ JM JOBRUN ERROR - A TP has notified the OS that a newly loaded job is running when no such notification was expected.

@59@ Reserved.

@5A@ JM SELECT ERROR - The processor selected for the load of a job has failed for unexpected reasons.

@5B@ JM TASK ID ERROR - A task processor sent a task ID to the OS that does not exist.

@5C@ JM SYSCONFIG IO ERROR - An IO error occurred while JM was accessing the SYSCONFIG file during warmstart.

@5D@ Reserved.

@5E@ Reserved.

@64@ DC RWL ERROR - The DCA (Data Comm Activity) attempted to access one of the queues, but was stopped because the READ-WITH-LOCK flag was set.

@65@ DC IOI Error - The result function has found an invalid message type on the system result queue. This problem could be caused by running with an incompatible version of MCP- and NPC-generated code files.

@66@ Reserved.

@6E@ PI PROCESSOR ERROR - An attempt was made to send a message to an illegal bus address (X15, for example).

@6F@ PI TP TASK ID ERROR - The TP was asked to do something for a task that is not running on that TP.

@70@ MOVE RESULT ERROR - An unidentified result was returned from PI_MOVE by the TP.

@71@ Reserved.

@72@ Reserved.

@78@ Reserved.

@79@ Reserved.

@96@ BAD DFH LOCK ERR - OPEN or CLOSE attempted to read a disk file header without having acquired the disk file header lock.

@97@ ZERO ALLOC ERR - The system attempted to allocate a 0-sized area on disk.

@98@ BAD ALLOC ERR - The system attempted to allocate an area with an area number greater than 16.

@99@ BAD INDEX ERR - The system attempted to access a disk file header with a directory index greater than 2805.

@9A@ Reserved.

@9B@ BAD DFH ERR - The system attempted to write a disk file header with an improper value in the FLAGS or EOF field.

@9C@ BAD PACK TAG ERR - A PPIT search operation returned an invalid pack tag to the system.

@C8@ Reserved.

@F0@ DISK PROC ERROR - The disk processor detected an error. To determine the reason for the error, a ROM dump must be taken and analyzed. The Save Area of the dump contains the IC Status which will indicate the type of error. Figure 8-1 of this manual should be helpful.

@F1@ SUZIP FAIL ERR - On an unattended system, the program to be zipped at system startup time encountered a Load failure. The following two bytes of the display give the FCM value for the zip failure.

APPENDIX B
TASK PROCESSOR SAVE ERRORS

The task processor error numbers defined below are reported as the "Error Number" in a processor entry of the error log or in the dump's State Save Area for a task processor.

@00@	Hardware detected error.
@01@	FS YIELD PROC - Force state has been detected while entering PM function YIELD PROCESSOR. This function should not gain control while in force state.
@02@	ILL CALL AP - Two actions are allowed to call PM function ASSIGN PRIORITY; these are JC INTERP ACTION and JC COMMAND HANDLER. These errors indicate the caller was neither.
@03@	FS ASG PRIOR - Force state has been detected while entering PM function ASSIGN PRIORITY. This function can not gain control while in force state.
@04@	FS INVOKE - Force state has been detected while entering PM function INVOKE. This function can not gain control while in force state.
@05@	FS FINISH - Force state has been detected while entering PM function FINISHED. This function can not gain control while in force state.
@06@	FINISH ON Q - PM FINISHED has been invoked to take a status block (SB) off the linked lists; but the SB can not be located. Assume that the SB is still on a queue which is in error.
@07@	Reserved
@08@	Reserved
@09@	ENQ ILL Q - An invalid queue name has been passed to PM function ENQUEUE.
@10@	Reserved
@11@	DEQ ILL Q - An invalid queue name has been passed to PM function DEQUEUE.
@12@	ILL POST - A PM POST has been invoked on a SB that does not have a status of WAITING POST.
@13@	DEQ NOT HEAD - A SB has invoked DEQUEUE to get off a queue while it is not at the head of the queue.

@14@ FS PRE POST - Force state has been detected while entering PM function PRE POST CALLER. This function cannot gain control while in force state.

@15@ FS MEMSPACE - Force state has been detected while entering PM function MEM SPACE. This function cannot gain control while in force state.

@16@ FS MARK - Force state has been detected while entering PM function MARK. This function cannot gain control while in force state.

@17@ MEM SP NO WB - A JC or PI action has called MEM SPACE to allocate a WB when the maximum limit of four has already been allocated.

@18@ FS FREE SPACE - Force state has been detected while entering PM function FREE SPACE. This function cannot gain control while in force state.

@19@ ILL FREE SPACE - A SB has called FREE SPACE to deallocate a structure, and FIND WB was unsuccessful in locating the structure.

@1E@ TASK LIMIT EXCEEDED - More than the legal amount of tasks allowed in the mix have been invoked.

@20@ FS LINK WB - Force state has been detected while entering PM function LINK WB. This function cannot gain control while in force state.

@21@ ILL INVOKE - An action has been called by INVOKE, but the action name does not exist (is out of bounds).

@22@ MARK CODE SEG - MARK has been called to modify a structure of type SCODE. Only VM is authorized to do this.

@23@ INVOKE VM - VM has been called by INVOKE; VM cannot be invoked.

@24@ INV TOO MANY ACT - An action being invoked is non-critical; yet, due to lack of space in the SB ADDR ARRAY, it has overflowed into the area reserved for critical actions.

@25@ AP NOT ON LIST - A search of the list heads by the action ASSIGN PRIORITY was unsuccessful in locating the SB of the task.

@26@ DEALLOC TYPE - Work block being deallocated was neither type Normal, SB, ICB, TCB, or PCB. Invalid use of DEALLOCATE.

@27@ ILL TRANSFER WB - TRANSFER WB was called to give up

ownership of a WB,but was not successful in locating it.

@28@ ILL LINK WB - LINK WB was called to delink a WB and relink it, but WB was not found.

@29@ Reserved

@30@ WB NOT FOUND - FIND WB has not found a specified WB, and the error flag was on. In some cases, not finding a WB is not an error, but it is in this case.

@31@ FS LOCATE SSEG - Force state has been detected while entering PM function LOCATE SSEG. This function cannot gain control while in force state.

@32@ ILLEGAL JOB COMMAND - The OS processor has requested the TP to perform an invalid job command.

@33@ ILLEGAL IN REQ - The OS requested an invalid function of the task processor.

@34@ INTERPRETER NOT LOADED - The OS requested the task processor to run a task for which there is no interpreter.

@35@ ILL MARK REQ - MARK has been called in order to mark a WB, but FIND WB was not able to locate the WB.

@36@ FS GET SEGS - Force state has been detected while entering PM function GET SEGS. This function cannot gain control while in force state.

@37@ WS FATAL - No entry in the warmstart table (planted at startup time) has been found for this task processor during START PM.

@38@ INV ILL INT TASK - An interpreter action has been detected having an illegal action ID. This means the ID is not in the range of SB ADDR SLOTS set aside for INTERP ACTIONS. (INVOKE)

@39@ MARK DATA SEG - MARK has been called to modify an s-data type structure. Only VM is authorized to do this.

@3C@ GET SEGS MANY - An interpreter requested an invalid number of segments (0 or more than 3) or requested more than one code segment.

@3D@ GET SEGS KIND - The interpreter requested an invalid type of segment (not a data or a code segment).

@3E@ INVALID INTERPRETER NEED - The interpreter requested an invalid NEED. Valid NEEDs are: Communicate, Get Segment, Yield, Program Error, or System Message.

@40@ Reserved

@41@ CLEAR START ERR - Indicates that START PM has been entered for a second time. START PM is not reentrant. (HARD ERROR)

@42@ Reserved

@46@ SOFTWARE ERROR - Task processor execution led to an illegal software operation.

@47@ HARDWARE ERROR - The TP went logically Not Ready due to a hardware-detected error: Parity Error, Memory Limit Error (the TP attempted to address a nonexistent memory), Boundary Error (the TP tried to increment addressing through the end of a memory page: @FFFF@).

@50@ WS PAGE 0 SIZE - The size of page 0 is too small to support the execution of the ICP.

@51@ WS END OF TABLE - the Warmstart Table loaded by the OS is corrupted.

@52@ VM TID ERROR - FIND WB was unable to find the TCB of the task. (VM ACT CODE of VM)

@53@ VM ERROR - Descriptors exist for data and code segments in a TCB, PCB, or ICB; if the type passed is neither of these, error is noted. (FIND SSEG DESC of VM)

@54@ DISK IO ERROR - FIND SSEG DESC was unsuccessful in locating the segment descriptor. (DO DISK IO of VM)

@55@ GETSEGSP ERROR - FIND SSEG DESC was unsuccessful in attempting to relocate the seg table. (GET SEG SPACE)

@56@ SETUP DISK ERROR - The number of segments requested to be brought in was greater than the maximum of 3. (SET UP DISK of VM).

@57@ SETUP2.DSK.ERROR - In procedure SET.UP.DISK, which is building a SYSIO COMM to bring in segs, the FIND SSEG DESC rountine was unsuccessful. (SET UP DISK of VM)

@58@ CHECK VIC ERROR - VM victim segment has no associated descriptor; FIND SSEG DESC failed.

@59@ VM NO ROOM - VM was invoked for other than LOCATESEGS. (CHECK NO ROOM). Segments can not fit in memory.

@60@ NO MEMORY - MAKE SPACE has been unsuccessful. (VM ACT CODE)

@61@ VICTIM SEG ABS - A segment chosen as a victim by VM has

an associated descriptor which shows the segment absent. (CODE DATA VICTIM)

@62@ DO COMP ILL TYPE - During the actual compacting part of VM, the next block of memory has been found to have an illegal type. (DO COMPACTING)

@63@ SEG COMP NO WB - In DO COMPACTING, SEG COMPACT has been called, which called FIND SSEG DESC, and failed to find the descriptor table. (SEG COMPACT)

@64@ GETSPACE TYPE ERR - A segment type is out of range, meaning it is neither VM FILE, PROG FILE, INT TYPE, ZERO, or GARBAGE FILL. (GET SEG SPACE in VM)

@65@ Reserved

@66@ VM GETSEGS NOTCB - VM has called FIND SSEG DESC to find the ICB/TCB/PCB for a segment, and it was unsuccessful. (VM GET SEGS in VM)

@67@ Reserved

@67@ Reserved

@68@ SRCH UPSEGS ADDR - The last pass of the virtual memory algorithm has searched through memory, and was not successful in locating space.

@69@ SRCHUPSEG NOTCB - FIND SSEG DESC has failed. (SEARCH UPDATED SEGS)

@80@ JC INTERP COMM - TCB flags are zero. (JC INTERP ACTION)

@81@ JC ILL OP CODE - Command handler action has detected that a command is out of range. Bad parameter.

APPENDIX C
SYSANALYZER ERROR MESSAGES

An error detected in the dump file by the SYSANALYZER, or an error in the analysis, causes an error message to be generated within the analysis printout. These error messages are identified by one string of three dollar signs before the message, and another after the message: "$$$ ERROR MESSAGE $$$." Messages without three dollar signs are merely warnings.

$$$ NO xx ID.TABLE ENTRY $$$

No ID.TABLE entry (as documented in the dump file map) was found for the memory type requested. The memory types ("xx") are defined as follows:

OS -Operating System
TP -Task Processor
DC -Data Comm Processor
DK -Disk Processor
BM -Buffer Memory

$$$ CIRCULAR LIST ERROR $$$

A list that circled back on itself did not circle back to the beginning.

$$$ REVERSE LINK ERROR $$$

In a list that has forward and reverse links, a forward link was followed from entry <a> to entry <b>, but the reverse link in entry <b> did not point to <a>.

$$$ QUEUE TAIL ERROR $$$

While analyzing a queue having a head and a tail, the tail was found not to be pointing at the last entry reached via the forward links.

$$$ TABLE SIZE NOT MODULO 8 $$$

If this message occurs during the TCB analysis, it means that the difference between the stack base and the DST base in a certain TCB is not modulo 8.

If this message occurs during the ICB/PCB analysis, it means that the difference between the Code Segment Table (CST) limit and the CST base in a certain PCB is not modulo 8.

$$$ QUEUE LINK GREATER THAN QUEUE SIZE $$$

The queue link points beyond the end of the table of status blocks through which the queue is linked.

$$$ STE LENGTH NOT EQUAL SD SIZE $$$

The segment table entry length is not equal to the segment descriptor size.

$$$ SD OWNER NOT EQUAL PD/BD OWNER $$$

The segment descriptor owner is not equal to the physical descriptor/block descriptor owner.

$$$ SD WB.ID NOT EQUAL PD/BD WB.ID $$$

The segment descriptor work block ID is not equal to the physical descriptor/block descriptor work block ID.

$$$ SD SEGMENT NUMBER INCORRECT $$$

The segment table entry number does not equal the segment number.

$$$ LIST INDEX GREATER THAN LIST SIZE $$$

List index is greater than the list size.

$$$ DYNAMIC ARRAY DECLARATION ERROR $$$

An attempt to declare data space dynamically in order to perform an analysis resulted in a value either zero or greater than 5000 bytes. The data declaration does not occur, and the specific subsection of analysis is skipped.

$$$ LIST SHOULD NOT BE CIRCULAR $$$

A straight list has been detected to circle back on itself.

$$$ DISK READ ERROR $$$

An "ERROR" status has been returned to SYSANALYZER while it was trying to read from disk. Analysis continues at the next logical startup point.

$$$ SEGMENT TABLE ID MISMATCH $$$

The value of a segment-id, as passed to the SYSANALYZER in the dump file's header record, does not match the value built into SYSANALYZER in order to print the names of the various segments in the segment table.

$$$ ADDRESS ERROR $$$

When the value of the indicated address is added to the indicated length or offset, the sum exceeds the last address for the page of memory involved. (The last address for the page is determined from the Dump File Map.)

$$$ TYPE INCORRECT $$$

The least digit of the physical descriptor type is incorrect. For TCB analysis, it should be 7; for PCB analysis, 8; and for ICB analysis, 9.

$$$ P.STATUS /=WAIT.CRIT.INV $$$

The status field of the status block does not equal @04@.

$$$ ID'S DO NOT MATCH $$$

The preceding and trailing IDs do not match.

$$$ SIZES DO NOT MATCH $$$

The preceding and trailing sizes do not match.

$$$ SIZE LESS THAN 7 $$$

The preceding size is less than 7.

$$$ FORWARD PD ID ERROR $$$

The forward physical descriptor ID is not @06@.

$$$ REVERSE PD ID ERROR $$$

The reverse physical descriptor ID is not @06@.

$$$ FORWARD PD SIZE ERROR $$$

If 6 is added to the status block size array entry that is indexed by the action name, the result does not equal the forward physical descriptor size.

$$$ REVERSE PD SIZE ERROR $$$

If 6 is added to the status block size array entry that is indexed by the action name, the result does not equal the reverse physical descriptor size.

APPENDIX D
USER TASK ANALYSIS

To assist in tying a user job to all of the various control and parameter blocks associated with it, use the following outline:

VIRTUAL MEMORY

1. From the mix table, locate the desired job and record the following information:

   1) Task ID.

   2) Mix Processor.

   3) Program FIB ID (least significant byte = PCB ID). The PCB ID can be used to locate the PCB in the task processor.

TASK PROCESSOR <PROCESSOR NUMBER>

1. Scan the TCB.heads for BH.ID = <task ID>. NOTE: scan all memory pages.

2. Beneath each TCB are the FPBs associated with that task, if any.

   The least significant byte of the "SD" number of each FPB is the associated FIB segment number (FIB ID).

OPERATING SYSTEM

1. Scan the WB.LB.HEAD for "owner" of DA.TASK <task ID>.LIST.
   NOTE: Convert <task id> to decimal.

2. Scan the FIB List below the WB.LB.HEAD for:

   1) "Owner" as determined in step 1 above.

   2) "ID" (FIB ID) as determined in step 2 for Task Processor number above.

   3) Assemble a DA.FILE-TAG from FIB entries:

      a. File tag info (1 byte).

      b. File tag directory (2 bytes).

      c. File tag packtag (1 byte).
         NOTE: DA.FILETAG = <file tag info><file tag directory><file tag packtag>.

BUFFER MEMORY

Scan FTE entries for DA.FILETAG as assembled in step 3) above.

The FTEEs associated with the FTE are immediately following it.

NOTE

Four FTEE descriptors will be listed, and the map of the FTE will indicate the number of them that are active. The associated 180-byte buffer sections and their contents will follow each descriptor. The buffer sections are linked together by the "trailing link," which points to the address of the next buffer section. The last buffer section of an FTEE will be linked to @FFFF FFFF@. The FTEE itself will have a trailing link of Fs unless there is more than one FTEE associated with an FTE.

APPENDIX E
A GUIDE FOR TROUBLESHOOTING SYSTEM FAILURES

INTRODUCTION

Some initial steps should be taken in any trouble-shooting effort. It is not always possible to do all of the things suggested but the more of these suggestions that are followed the easier the task of trouble-shooting will be.

1. Record all information which appeared on the ODT and on the lighted displays when the error occurred.

2. Get a dump.

If the system clearstarted and created a dump file in the process, this will be indicated by messages which appear on the ODT. If no messages have appeared and the system will still respond, take a dump using the GT MD command. IF the ODT is not communicating, a ROM dump should be taken.

Once the dump exists it is ready to be converted into a readable format.

If the dump is a system dump (created automatically or by GT MD), it can be converted using the SYSANALYZER program.

If the dump is taken using the Hex keypad (a ROM dump), it should have ROMCONVERT run against it. The output of this procedure is then used as input for the SYSANALYZER program.

There are times when ROMCONVERT can not be run. In this case, run ROMANALYZER against the ROM dump.

(Directions for all these procedures are contained in the System Operations Guide.)

3. Acquire the log file of the activities on the system when the problem occurred (use the PL command).

4. Make a copy of the SYSCONFIG file.

5. Talk with the personnel at the site and make a note of any unusual activity when the failure occurred.

6. If a clearstart occurred and the error message is available (the number in Bank E of the Hex display), the meaning of the message should be established using Appendix A to decode the number.

SUGGESTIONS/HINTS

The suggestions which follow have been accumulated from successful field engineers and are included here in the hope that they may be of help to others.

IF THE SYSTEM CAN'T BE RE-WARMSTARTED

Try the redundancy switch(es), OS and/or DISK, to confirm that the problem is in the suspected primary processor.

Coldstarting is a good test of the hardware. It is a good idea to:

Replace the system files (RP).

If it will not damage the customer's operation too much, re-initialize the fixed disk, and then load the system files from the B900RL1 disk using the LD command.

PERSISTENT PROBLEM

Run the appropriate MTRs. Be sure to check the MTR configuration against the configuration shown when CONFIGURER is run.

A LARGE NUMBER OF CHIPS SEEM TO BE FAILING

The problem may be a wire to that card/circuit or a track on that card.

ENVIRONMENT IS IMPORTANT

If the room is too hot the disk platters may expand and the disk can not be read properly.

If the air is too dry, static may be causing problems and static mats may be needed.

Review the log files for the date and time of failures. It may turn out that the customer is working after hours or on weekends when in-house air conditioning is switched off.

CHECK CONNECTIONS

Check the power supply. Hot or discolored fuses indicate that the connection is bad. This can cause low line voltages which can, in turn, affect the system memory.

All I/O cables and I/O cable plugs should be checked to be sure that they are firmly seated.

The examples which follow are intended to be helpful for problems on different system configurations. It should not be assumed that all types of problems have been addressed.

In following the outlines for the different system configurations, follow the instructions only to the point where a reason for the failure is found.

SYSTEM CONFIGURATION 1

System aborts with a SPO and a printer

CONDITION 1: CLEARSTART - DUMP CREATED

1. Analyze clear start message.

2. Will system WARMSTART?

   1) If no:
      a. Analyze WARMSTART failure code.
      b. Replace system files, and rewarmstart.
      c. Run MTRs.

   2) If yes:

      a. Print and analyze log files.
      b. Print and analyze KA and LR of all disks on line at time of CLEAR/START.

c. Run MTR.
d. Print dump file.
e. Analyze appropriate areas for hardware errors.

3) If reason for CLEAR/START is not found in steps a or b, submit FTR.

CONDITION 2: CLEARSTART - DUMP NOT CREATED

1. Analyze clear start message.

2. Take ROM dump.

3. Will system WARMSTART?

   1) If no:

      a. Analyze WARMSTART failure code.
      b. Relace system files, and rewarmstart.
      c. Run MTRs.

   2) If yes:
      a. Print and analyze log files.
      b. Print and analyze KA and LR of all disks on line at time of CLEAR/START.
      c. Run ROMANALYZER and analyze IC status words.
      d. Run ROMCONVERT and SYSANALYZER.
      e. Run MTRs.
      f. Analyze appropriate areas for hardware errors.

If reason for CLEAR/START is not found in steps a or b, submit FTR.

CONDITION 3: SYSTEM HUNG - NO RESPONSE TO SPO MESSAGES

1. Attempt a ROM dump.

   1) If ROM does not take dump, run MTR.
   2) If ROM dump is taken, attempt to WARMSTART.
2. Will system WARMSTART?
   1) If no:

      a. Analyze WARMSTART failure code.
      b. Replace system files.
      c. Run MTRs.
   2) If yes:
      a. Print and analyze log files.
      b. Print and analyze KA and LR of all disks online at time of CLEAR/START.
      c. Run ROMANALYZER, and analyze IC status words.
      d. Run ROMCONVERT and SYSANALYZER.

e. Run MTRs.
a. Analyze appropriate areas for hardware errors.

3) If reason for "hang" is not found in steps a or b, submit FTR.

CONDITION 4: TASK PROCESSOR MARKED NOT READY - DUMP CREATED

Does the system have an MPL processor available?

1. If yes:
   1) Print and analyze log files.
   2) Print and analyze a KA and LR of all disks online at time of failure.
   3) "PO" system and run MTR.
   4) Run SYSANALYZER and analyze appropriate areas.
   5) If the reason for the task processor being marked not ready is not found in steps 1), 2), or 3), submit FTR.
2. If no:
   1) "PO" system and WARMSTART.
   2) Will the task processor run "PL"?
      a. If yes:
         a) Print and analyze log files.
         b) Print and analyze a KA and LR of all disks online at time of failure.
         c) Run MTR.
         d) Run SYSANALYZER and analyze appropriate areas.
         e) If the reason for the task processor being marked not ready is not found in above step, submit FTR.
      b. If no:
         a) Replace system files.
         b) Run MTRs.

SYSTEM CONFIGURATION 2

System aborts on a system without a printer

CONDITION 1: CLEARSTART - DUMP CREATED

1. Analyze CLEAR/START message.

2. Will system WARMSTART?

   1) If no:
      a. Analyze WARMSTART failure code.
      b. Replace system files.
      c. Run MTRs.
   2) If yes:
      a. Run PL with display option and analyze any entries.
      b. Run MTR.
      c. If the reason for the CLEAR/START is not found by steps 1)

or 2), submit FTR.

a) Run KA and LR to printer backup.

b) Copy dumpfile, printer backup files, and log files to a removable disk, and transport to a system with a printer for further analysis.

c) If the reason for the CLEAR/START is not found by the above step, submit FTR.

CONDITION 2: CLEARSTART - DUMP NOT CREATED

1. Analyze CLEAR/START message.

2. Set the OPERATE/MEM DUMP switch to MEM DUMP.

3. Interrogate the contents of the OS processor page 0 offsets 000F through 001C memory. (Refer to CMS SOG.)

   See NOTE 2 for OS save state area.

4. Take ROM dump.

5. Will system WARMSTART?

   1) If no:
      a. Analyze WARMSTART failure code.
      b. Replace system files.
      c. Run MTRs.

   2) If yes:
      a. Run PL with display option and analyze any entries.
      b. Run MTR.
      c. Run KA and LR to printer backup.
      d. Copy printer backup files and log files to a different removable disk than used for the ROM dump, and transport to a system with a printer for further analysis.
      e. If the reason for the CLEAR/START is not found in the above step, submit FTR.

CONDITION 3: SYSTEM HUNG - NO RESPONSE TO SPO MESSAGES

1. Attempt to display the contents of the OS processor page 0 offsets 000F through 001C memory. (Refer to CMS SOG.)

   1) If ROM does not display, run MTR.

   2) If ROM does display, analyze using layout described in step 3 of the previous condition.

2. Attempt a ROM dump.

   1) If dump not taken, run MTR.

2) If dump taken, attempt to WARMSTART

a. If no:
   a) Analyze WARMSTART failure code.
   b) Replace system files.
   c) Run MTRs.

b. If yes:
   a) Run PL with display option and analyze any entries.
   b) Run MTR.
   c) Run KA and LR to printer backup.
   d) Copy printer backup files and log files to a different removable disk than used for the ROM dump, and transport to a system with a printer for further analysis.
   e) If the reason for the "hang" is not found in the previous step, submit FTR.

CONDITION 4: TASK PROCESSOR MARKED NOT READY - DUMP CREATED

Does the system have an MPL processor available?

1. If yes:
   1) Run PL with "display" option and analyze any entries.
   2) Run MTR.
   3) Run KA and LR to printer backup of all disks online at time of failure.
   4) Copy printer backup files, log files and memory dumpfile to removable disk, and transport to a system with printer for further analysis.
   5) If the reason for the failure was not found in the above step, FTR.

2. If no:
   1) "PO" system and WARMSTART.
   2) Will the task processor run PL with the display option?
      a. If yes:
         a) Analyze any entries.
         b) Run MTR.
         c) Run KA and LR to printer backup of all disks online at time of failure.
         d) Copy printer backup files, log files, and memory dumpfile to removable disk and transport to a system with printer for further analysis.
         e) If the reason for the failure is not found in the above step, submit FTR.
      b. If no:
         a) Replace system files.
         b) Run MTRs.

SYSTEM CONFIGURATION 3

System aborts with no SPO and no printer.

It will be very difficult to determine the time and type of failure on this system cofiguration. Once a failure has occurred, the following steps can be followed:

1. Attempt to display the contents of OS and disk processors save state areas. See NOTE 2.

   1) If ROM does not display, run MTR.

   2) If ROM displays, analyze using the layout provided in system configuration 2 condition 2.

2. Attempt a ROM dump, then run the MTR.

3. Will the system WARMSTART?

   1) If no:
      a. Analyze the WARMSTART error code.
      b. Replace system files.

   2) If yes:
      a. Run KA and LR to printer backup of all disks on line at time of failure via the remote SPO.
      b. Copy printer backup files and log files to a different removable disk.

NOTE 1: If there was not sufficient disk space at WARMSTART time to create an empty dump file, a dump will not be taken in the event of a CLEAR/START, even if the system had the ability to do so. This case does not apply for this part of the outline because a ROM dump would not reflect the condition of the memory at the time of the failure.

NOTE 2. Disk Processor Hard Errors: When causing freeze, these will display "F00000" on the hex indicators: an unexpected "not ready", (select lock, 211 relocation table full, for example) can cause the system to eventually hang waiting on virtual memory.

IC status is stored at the end of disk processor memory, indicated by "DP-SIZE" in a dump. It is stored as status 1, error status, status 3, status 2.

"DP-SIZE" is located at memory address @003A@ on page 0 of the disk processor.

NON-OS PROCESSORS

Errors on these processors cause values to be saved at the following addresses in their own memories:

Address	Length	Stored Item
000B	1	ERROR INDICATOR
000C	1	B0
000D	1	REQ
000E	2	B1FL
0010	2	B32
0012	1	AD
0013	2	J
0015	2	K
0017	2	L
0019	8	XY
0021	2	M1
0023	2	M2
0025	2	WR
0027	2	MAX
0029	2	MXA
002B	2	MXB
002D	2	UMAR3
002F	2	UMAR4
0031	2	UMAR5
0033	2	UMAR6

NOTE

ERROR INDICATOR = 00 = Hardware detected error.

OS PROCESSOR

Hard errors on the OS processor cause values to be saved at the following addresses on page 0:

Address	Stored Item	Length
000F	M1	2
0011	M2	2
0013	MAX	2
0015	UMR2	2
0017	UMRX2	2
0019	STATUS1	1
001A	ERROR STATUS	1
001B	STATUS3	1
001C	STATUS2	1

APPENDIX F
GLOSSARY OF COMMON TERMS

This glossary only includes terms and abbreviations that are global to the system. Many terms are explained in the appropriate place within the "SYSANALYZER output" sections of this document. Throughout this document, the first definition will be implied unless explicitly specified otherwise.

NOTE

Several terms are described as "CMS standard." These are further described in the CMS MCP Manual.

ABP	Available Buffer Pool. The pool of available data comm buffers. Managed by DCA.
Action	A discrete unit of code achieving a single function.
Activity	A logically related group of system functions comprised of a set of interdependent actions.
AM	Activity Management. An OS activity. See Section 8 for a concise description.
AM.ACTION.LIST	A linked list of status blocks, one block per active OS action.
AVR	Automatic Volume Recognition. CMS standard.
BDS	Basic Data System . The underlying microprocessor see "NBDS" and "NPRO."
BD	Block Descriptor. See Task Processor, Section 10.
BSMD	Burroughs Super Mini-Disk.
CDC	Cartridge Disk Control.
CMS	Computer Management System. CMS standard.
CPA	Communicate Parameter Area. CMS standard.
Critical	An action is known as critical if it is not re-entrant. Area of an action, known as a region, may be critical without the whole action being critical. Both actions and regions may be critical because they invoke a critical action.
CST	Code Segment Table. CMS standard.
DA	Data Access. The OS activity handling all data operations. See Section 8.
DA_CT	Data Access Configuration Table.

DC	Data Comm. Usually used with reference to data storage areas of the data comm activity.
DCA	Data Comm Activity. OS activity for handling all data comm communicates. See Section 8.
DCI	Data Communications Interface control. That part of the DCP hardware that enables the connection of an external line to the data comm processor.
DCP	Data Communications Processor.
DMTR	Display and Maintainance Test Routine control. That part of the DS&M processor hardware responsible for handling the hexadecimal keypad, the displays, and the MTR switches.
DS&M	Data Storage and Maintenance processor.
DP	Disk Processor. Same as DS&M.
DSCP	Data Storage Control Program. The firmware that executes in the DS&M.
FDC	Fixed Disk Control.
FIB	File information Block. For a user file this is a reflection of a user FPB.
FPB	File Parameter Block. CMS standard.
Frozen	1. A frozen processor is an NPRO that has deliberately stopped itself. 2. A frozen segment is an element of managed memory (in any processor) that cannot be swapped out or moved.
.HCT	History Communicate Table.
IC	Interface Control. The hardware interface to the Processor Interconnect Bus.
ICB	Interpreter Control Block. CMS standard.
ICMD	Industry Compatible Mini-Disk.
ICP	Interpreter Control Program. The firmware that runs in the task processor.
JC	Job Control. The ICP activity that schedules and controls the execution of user jobs within the task processor.

Jelled	A jelled segment is an element of managed memory that may not be swapped out, but which may be moved. See frozen.
JM	Job Management. The OS activity for job related OS functions. See Section 8.
LB	Linked Block.
LSB	Least Significant Bit or Byte (depending on the context).
Mailbox	The communication area of two processors. The content and use of mailboxes is covered in sections 2 and 10.
MCP	Master Control Program. Generic Burroughs term for an operating system. See CMS MCP Manual for specific CMS implementations.
MCS	Message Control System. A user program responsible for controlling all or part of a data comm network.
Mix Number	External user reference to a running job. Mix numbers are assigned from 1 to 99 in a round-robin manner.
	NOTE
	Internally the system uses a (different) task ID. Both are kept in the mix table.
MN	Monitor.
Monitor	The OS activity responsible for system and error logging and the detection of peripheral device errors. See Section 8.
MSB	Most Significant Bit or Byte, depending on the context.
NBDS	An n-channel BDS processor. See also BDS and NPRO.
NPRO	The standard processor module. An extended n-channel BDS microprocessor.
ODT	Operator Display Terminal.
OI	Operator Interface. The OS activity handling all operator messages. The activity also handles I/O to the ODT directly. See Section 1.
OS	Operating system. That portion of the MCP that runs in the OS processor, and the processor itself. See Section 8.

PCB	Program Control Block.
PD	1. Physical Descriptor. Used in relation to task and buffer memories. 2. MCP utility to report all or part of a disk directory.
PHT	Peripheral Handling Table. Used by DA to manage all peripherals attached to the system (other than data comm).
PI	Processor Interface. A group of related modules of code appearing as one activity in all processors (other than the DS&M) that communicate with one another. PI provides a processor to processor communication capability via the processor interconnect bus.
PIB	Processor Interconnect Bus. The hard bus connecting the IC modules of each processor subsystem.
PIC	Processor Interface Control. Same as PIB and IC.
PM	Processor Management. The ICP activity responsible for the control of the task processor.
PO	Power Off. The MCP intrinsic used to request the system to logically release a disk peripheral.
Port	The I/O path of the NBDS. Each NBDS has the capability of addressing eight ports. Port 0 (and port 1 in the OS processor and DS&M) is reserved for interprocessor communication.
Region	A sub portion of an action. The term is normally only used in the context of critical.
RSO	Remote SPO Operation. A system capability to allow the ODT function to be transferred to a remote device via an MCS.
RTC	Real Time Clock. Also known as TOD (Time Of Day).
SB	Status Block. See status block.
SCL	System Control Language. Standard CMS.
SD	Segment Descriptor.
Section	A portion of buffer memory. Usually a part of a buffer of user data. Currently a section is 180 bytes.

SPO	Alternative mnemonic for ODT.
Stackframe	An internal save area maintained for every action on a soft stack.
Status Block	A standard storage area reflecting the current status of an action. Active entries appear in the AM.ACTION.LIST.
STE	Segment Table Entry.
Task ID	The internal reference to a job. This number is a direct index into the mix table. See mix number.
TCB	Task Control Block. CMS standard.
TOD	Time Of Day. Alternative name for the real time clock.
TP	Task Processor.
VM	Virtual Memory.
WB	Work Block.
WLP	Wide Line Printer. I/O control for line printers.
Work Block	A Work Block is owned by an action, and is used for a variety of data storage purposes.

Documentation Evaluation Form

Title: B900/CP 9500 Systems Memory Dump Users Guide

Form No: 1118478

Date: November, 1982

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 _____

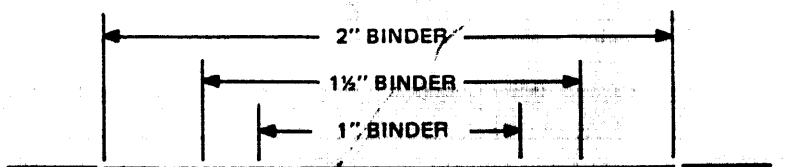
Address _____

_____

Phone Number _____ Date _____

Remove form and mail to:

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



B 900/CP 9500 Systems

MEMORY DUMP USER'S GUIDE

1118478

Printed in U.S.A.

Printed in U.S.A. November 1982 1118478