Burroughs

Functional Description Manual

B 1000 Systems Memory Dump Analysis

(Relative to Mark 12.0 System Software Release)

Distribution Code SC

Priced Item
Printed in U.S.A
March 1985

1152055

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.

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the CLASS specified as 2 (S.SW: System Software), the Type specified as 3 (DOC), and the product specified as the 7-digit form number of the manual (for example, 1152055).

LIST OF EFFECTIVE PAGES

Page	Issue
Title	Original
ii	Original
iii	Original
iv	Blank
v thru xii	Original
1-1 thru 1-11	Original
1-12	Blank
2-1 thru 2-40	Original
3-1 thru 3-2	Original
4-1 thru 4-3	Original
4-4	Blank
5-1 thru 5-7	Original
5-8	Blank
6-1	Original
6-2	Blank
A-1 thru A-7	Original
A-8	Blank
B-1 thru B-5	Original
B-6	Blank
C-1 thru C-9	Original
C-10	Blank
D-1 thru D-35	Original
D-36	Blank
E-1 thru E-7	Original
E-8	Blank
F-1 thru F-11	Original
F-12	Blank
G-1 thru G-22	Original
H-1 thru H-6	Original
I-1 thru I-6	Original
1 thru 4	Original

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF TABLES

PREFACE

This Burroughs B 1000 Systems Memory Dump Analysis Functional Description Manual describes techniques that a support person can use to locate the immediate and fundamental causes of system halts and system hangs, as well as certain performance problems. The manual provides a detailed description of the information contained in a system memory dump and a program memory dump and shows how to determine the state of the processor, the software, and the input/output operations at the time the system memory dump is taken.

Following are summaries of the contents of each section of the manual.

Section 1: System Memory Dump
The various types of system halts and system hangs are defined. Also, procedures for dumping the state of the system into the SYSTEM/DUMPFILE file are described, GISMO debugging aids are explained, and the Fault Dockets, forms to be completed at the time of a system halt or hang, are exhibited.

Section 2: The SYSTEM/IDA Program
Explains how to produce a formatted analysis of the SYSTEM/DUMPFILE file using the Interactive Dump Analyzer (SYSTEM/IDA) program.

Section 3: Operating System Components
Lists the components of the operating system and describes the functions of each component.

Section 4: Problem Analysis Overview
Provides an overview of the steps required to analyze system halts and system hangs.

Section 5: State of the Software
Describes how to determine the status of each operating system component and each job in the mix.

Section 6: State of the Input/Output Operations
Describes how to determine the status of each input/output operation represented in the system memory dump.

The appendixes cover the following topics:

Appendix A: examples of SYSTEM/IDA program execution.
Appendix B: hardware organization.
Appendix C: control of processor allocation by GISMO.
Appendix D: memory organization.
Appendix E: memory management.
Appendix F: input/output operations.
Appendix G: disk organization.
Appendix H: tape organization.
Appendix I: RPG memory dumps.

RELATED DOCUMENTATION

The following B 1000 manuals contain information related to the topics in this manual.

B 1000 Systems System Software Operation Guide, Vol. 1, form number 1169000.

B 1000 Systems System Software Operation Guide, Vol. 2, form number 1169091.

B 1870/B 1860 Systems Reference Manual, form number 1090644.

B 1000 Series Product Support Information Manual, form number 1137890.

SECTION 1
SYSTEM MEMORY DUMP

If the B 1000 system halts or hangs, several actions may be taken to isolate, analyze, and solve the problem. In the paragraphs that follow, along with definitions of the various types of system halts and system hangs, the steps to be taken by the system operator on a halt or hang are given. Also in this section, Fault Dockets, forms to be completed at the time of a halt or hang, are exhibited, and procedures for performing a CLEAR/START system memory dump and for using the GISMO trace routine are outlined.

SYSTEM HALT

A system halt is indicated when a system running under MCP control stops performing work, and the RUN indicator goes off. There are two classes of system halts: software-controlled system halts and undefined system halts.

Software-Controlled System Halt

A system halt with the ERROR indicator off and a defined value in the L register is a software-controlled system halt. This type of halt may occur when an MCP component (GISMO, the Micro MCP, or the SDL2 Interpreter when running the SMCP) detects an invalid value in a data field or register, a condition in which further processing will result in data corruption or loss. When this happens, the contents of the L, T, X, and Y registers of the master processor) are written into the HINTS record in lower memory, and a HALT microinstruction is then executed. The L, T, X, and Y registers of the slave processor are written in a memory area allocated for the slave processor.

Defined L register values are those with any of the following patterns in the sixteen leftmost bit positions (bits 0-15) of the register.

Pattern	Routine or program that halted
@0000@	SDL2 Interpreter on behalf of the MCP
@0200@	Micro MCP
@0D00@	GISMO
@0D01@	GISMO
@00F0@	SYSTEM/INIT
@000F@	CLEAR/START

For more information on the use of the L register, refer to section 8 of the *B 1000 Systems System Software Operation Guide, Volume 1.* All relevant values for the 24-bit L register are listed there.

When a software-controlled system halt occurs, the operator should take the following action:

1. Fill in a Fault Docket. See Fault Dockets, later in this section.

2. Perform a CLEAR/START operation with a system memory dump. (Exception: No dump is required if HALT occurred during CLEAR/START or SYSTEM/INIT as indicated by LC or LD = F.) The CLEAR/START program and memory dump procedure are described in detail in section 4 of the *B 1000 Systems System Software Operation Guide, Volume 2.* Also see CLEAR/START and System Memory Dump, later in this section.

3. Package the system memory dump.

Undefined System Halt

A system halt with an undefined value in the L register and the ERROR indicator either on or off is an undefined system halt. An undefined L register value is one that is not listed in section 8 of the *B 1000 Systems System Software Operation Guide, Volume 1.*

The MCP components are designed to halt in a defined manner. An undefined system halt occurs when a processor or memory error leads to the execution of a HALT microinstruction. This may result from the transfer of processor control out of the defined instruction sequence, from the execution of one or more corrupted microinstructions, or from the detection of an irrecoverable hardware problem by the processor.

When an undefined system halt occurs, the operator should take the following action:

1. Fill in a Fault Docket.
2. Perform a CLEAR/START operation with a system memory dump.
3. Package the system memory dump.

SYSTEM HANG

"Hang" is the name given to the condition in which a system running under MCP control does not respond to ODT commands even though the RUN indicator is on. If the system in this condition comes to a halt when the INTERRUPT button is pressed, an interruptible system hang has occurred. If the system does not halt when the INTERRUPT button is pressed but does halt when the HALT button is pressed, a non-interruptible system hang has occurred. If neither the INTERRUPT button nor the HALT button cause the system to halt, a processor hang has occurred.

Interruptible and Non-Interruptible Hangs

In an interruptible system hang, an MCP component or a job of very high priority is in a loop, preventing jobs of lower priority from running. Once the hang is detected, the operator should wait at least 60 seconds before pressing the INTERRUPT button because the MCP may wait that long for some I/O operations to complete.

If the INTERRUPT button has no effect, the condition may be a non-interruptible system hang or a processor hang.

A non-interruptible hang occurs when a microcoded MCP component is in a microinstruction loop and is not leaving the loop to test for the interrupt. In this case, pressing the HALT pushbutton brings the system to a halt.

If the HALT button has no effect, a processor hang condition exists. See Processor Hang.

For an interruptible or non-interruptible hang, the operator should take the following action:

1. Fill in a Fault Docket.

2. If the system hang is reproducible, set up for a GISMO trace of the disk channel(s), GISMO, port, MMCP, SMCP, scheduler, interrupt.handler, and timer.interrupt (TG = @0041B8@ for a single disk channel on channel 9, @0061B8@ for disk channels 9 and 10), and perform the sequence leading up to the hang again. A full GISMO trace is rarely required.

3. Perform a CLEAR/START operation with a system memory dump.

4. Package the system memory dump.

Processor Hang

A processor hang has occurred when the system is hung and neither the INTERRUPT nor the HALT button brings the system to a halt. This type of hang is caused by a hardware malfunction. In this situation, the operator should perform the following steps:

1. Push the HALT and CLEAR buttons at the same time.

2. Perform a clear/start operation and resume processing if possible.

3. Notify a Burroughs Field Engineer even if processing resumes.

There is no need to fill in the Fault Docket, set up a full GISMO trace, or take a system memory dump.

FAULT DOCKETS

Figure 1-1 shows the Fault Docket form for dual-processor systems, and figure 1-2 shows the form for single-processor systems. The forms are self-explanatory. Table 1-1 identifies the indicators and registers of interest.

B 1000 Dual Processor Fault Docket

OPERATOR:_____ SYSTEM:_____ DATE:_____ TIME:____

In the event of a system halt or hang, please answer the following (circle the appropriate response):

1. Are the RUN lights ON?
   Master: ON OFF Slave: ON OFF
   If both are OFF, then go to step 6.

2. Enter the HALT system command. Are the RUN lights ON?
   Master: ON OFF Slave: ON OFF
   If both are OFF, then go to step 6.

3. Press the INTERRUPT pushbutton. Are the RUN lights ON?
   Master: ON OFF Slave: ON OFF
   If both are OFF, then go to step 6.

4. Press the HALT pushbutton. Are the RUN lights ON?
   Master: ON OFF Slave: ON OFF
   If both are OFF, then go to step 6.

5. Press HALT and CLEAR. Call Field Engineer for assistance.

6. Is the ERROR light on?
   Master: ON OFF Slave: ON OFF

7. Is the STATE light on?
   Master: ON OFF Slave: ON OFF

8. What are the values in the following registers?

Master		Slave	
L	LR	L	LR
T	CC	T	CC
X	CD	X	CD
Y	PERM	Y	PERM
A	PERP	A	PERP
FA		FA	

9. Write the halt definition below:
   Master:

   Slave:

10. Comments:

    Master:

Figure 1-1. Fault Docket for Dual-Processor Systems

B 1000 Single Processor Fault Docket

OPERATOR:_____ SYSTEM:_____ DATE:_____ TIME:____

In the event of a system halt or hang, please answer the following (circle the appropriate response):

1. Is the RUN light on?
   ON OFF
   If OFF, then go to step 6.

2. Enter the HALT system command. Is the RUN light ON?
   ON OFF
   If OFF, then go to step 6.

3. Press the INTERRUPT pushbutton. Is the RUN light ON?
   ON OFF
   If OFF, then go to step 6.

4. Press the HALT pushbutton. Is the RUN light ON?
   ON OFF
   If OFF, then go to step 6.

5. Press HALT and CLEAR. Call Field Engineer for assistance.

6. Is the ERROR light on?
   ON OFF

7. Is the STATE light on?
   ON OFF

8. What are the values in the following registers?

L	LR
T	CC
X	CD
Y	PERM
A	PERP
FA	

9. Write the halt definition below:

10. Comments:

Figure 1-2. Fault Docket for Single-Processor Systems

Indicators:

Name	Usage
STATE	Indicates CC(0) is TRUE, used by system performance monitor.
RUN	Indicates processor is running, as opposed to halted.
ERROR	Indicates one or more bits in PERM or PERP is TRUE.

Registers:

Name	Usage
L	Working storage
T	Working storage
X	Working storage
Y	Working storage
A	Address Register for microinstructions
FA	Field address, the absolute bit address used to access a main memory data field.
LR	Limit Register
CC	CC(0) = STATE light on console CC(1) = Real time clock interrupt, set by hardware every 100 milliseconds CC(2) = I/O service request by one or more controls CC(3) = INTERRUPT button on console pushed
CD	CD(0) = M-register micro fetch parity error Cache Key parity error Cache double hit Console cassette parity error Uncorrectable S Memory parity error PERM register has changed S Memory field out of bounds S Memory microinstruction time out CD(1) = Memory Write/Swap out of bounds override CD(2) = Read out of bounds (FA < BR or FA > LR) CD(3) = Write or Swap out of bounds (FA < BR or FA > LR)
PERM	Parity ERror Memory PERM(0) = S Memory microinstruction time out PERM(1) = Read, Write, or Swap out of memory (FA > MAXS) PERM(2) = Error Log register changed PERM(3) = Uncorrectable CPU access error to S Memory If it occurs during fetch, the processor halts.
PERP	Parity ERror Processor PERP(0) = Cache double hit PERP(1) = Cache Key parity error PERP(2) = M Register parity error PERP(3) = Cassette read error

Table 1-1. Indicators and Registers

CLEAR/START AND SYSTEM MEMORY DUMP

After the Fault Docket form has been completed, a CLEAR/START operation with a system memory dump is called for on all halts and hangs.

DUMP System Option

The DUMP system option must be set if a CLEAR/START with a system memory dump is desired. If this option is reset, the SYSTEM/DUMPFILE file does not exist, its address in the COLD START VARIABLES is zero, and, therefore, a system memory dump operation is not possible. As standard operating procedure, it is advisable to run with the DUMP system option always set.

Memory Dump Details

The system memory dump must be completed successfully in the first CLEAR/START operation following the halt. This is because the first step in the process writes the entire contents of memory to the SYSTEM/DUMPFILE file on disk, and the second step (the CLEAR/START itself) clears memory, writing zeros and correct parity throughout. Thus, the state of the system when the problem occurred is no longer reflected in memory after a CLEAR/START operation.

The contents of the SYSTEM/DUMPFILE file are valid only if the system is running under MCP control at the time the system memory dump operation is performed. This process cannot be used for analyzing problems when the CLEAR/START, SYSTEM/INIT or STANDALONE programs are in control.

A special situation may arise when a system appears to hang, for example, printers and tapes stop, but it still responds to system commands. It is acceptable, in this case, to enter a DM system command with no mix number to get a system memory dump. However, this action sometimes causes the actual problem data to be lost --the processing associated with reading, recognizing, and executing the DM system command rearranges memory and changes the state of the system. Either of the following procedures avoids this problem:

1. Interrupt the system and take a CLEAR/START system memory dump, thus preserving the exact state of the machine at the time of the hang.

2. Run system performance monitoring to determine what the system is actually doing. System performance monitoring is described in appendix B of the *B 1000 Systems System Software Operation Guide, Volume 1.*

Packaging the System Memory Dump

If the analysis of the contents of the SYSTEM/DUMPFILE file is not to be performed immediately following the system memory dump operation, the current file must be packaged for later analysis. This is done by entering the PM system command to generate a packaged dump file with the default name of DUMPFILE/PM<nnn>, where <nnn> is the next number from the BACKUP stream.

The packaged dump file includes the contents of the SYSTEM/DUMPFILE file plus layout tables for the SMCP, the network controller, and the DMS access routines. Object code segments from memory at the time of the system memory dump are compared with the corresponding object code files on disk, and code segment comparison error information is included in the packaged file.

This packaged file is to be submitted with the Fault Docket and an FCF describing the problem.

GISMO TRACE

The GISMO trace facility is a powerful and useful debugging tool. It can be used to pinpoint I/O subsystem failures, as well as to track down system software problems. Invoking the GISMO trace facility incurs an overhead of 3 percent to 20 percent depending upon the functions being traced.

The GISMO tracing code is in a discardable segment that may be requested at CLEAR/START time. If it is not requested, the segment is discarded. If either the BR register or the TG register is non-zero at CLEAR/START time, the segment is included.

Setting the Trace Parameters

The TG command is used to permanently specify the trace parameters normally entered in the BR register and results in the same action as would have occurred had the parameters been loaded manually into the BR register during a CLEAR/START operation. Trace parameters that are specified by means of the TG command are overridden if the BR register is non-zero at CLEAR/START time.

Trace parameters that have been established during the CLEAR/START operation may be changed by means of the TG command; that is, a second CLEAR/START operation is not required to change the trace parameters.

On B 1990 systems, the trace flags are entered at CLEAR/START time by the TEXT TG command, followed by six hexadecimal digits representing the trace flags. No @ symbols are used. On the other B 1000 systems, the trace flags are entered at CLEAR/START time by loading the BR register after the TAPE mode finishes but before RUN mode commences. The L register is equal to @AAAAAA@ at this point.

The 24 bits comprising the trace parameters are defined in table 1-2.

Table 1-2. Trace Parameters

Bits	Function
0-14	Trace physical I/O on the corresponding channel Dispatch through channel table Reference address, op code, and disk sector address Service request Reference address and result descriptor Extended result descriptor Data transfer (when bit 22 is also set) Pocket select Seek complete Missing device Bad reference address
15	Trace GISMO scheduling and interrupt operations Block slave Unblock slave Block slave complete DCPU dispatch Set event for interrupt queue or I/O complete Save interrupt Fetch interrupt Communicate Rehang program Hang program Wait Cause program Set event index for waiting program Reinstate job Mark in queue Queue out top Adjust interpreter Communicate with GISMO Run MMCP MMCP page zero fault MMCP return
16	Trace GISMO port activity Port dispatch Port interrupt Port lockout Port missing device
17	Trace user interpreters (debug versions only)

Table 1-2. Trace Parameters (Cont)

Bits	Function
18	Trace MMCP (must use MICRO-MCP/DEBUG) CONDITIONAL.HALTS Logical I/O Interprogram communication
19	Trace SMCP CONDITIONAL.HALTS (sets bit 2 of SEGMENT_HALT)
20	Trace GISMO scheduler and interrupt handler Lock scheduler Unlock scheduler Run scheduler Interrupt handler Timer interrupt
21	Trace time stamp Include time stamp in trace table
22	Trace data transfers for selected channel(s)
23	Used internally by GISMO.

To trace all the non-I/O functions, bits 15, 16, 18, 19, 20, and 21 must be turned on. Enter:

TG @0001BC@

To trace only the disk channel, assumed to be located on channel 9, bit 9 must be turned on. Enter:

TG @004000@

To display the settings of the trace flags, enter:

TG

To reset the trace flags, enter

TG 0

Tracing of data transfers uses up the trace table very quickly, and should not be used unless advised by Burroughs.

Some of the Trace GISMO functions concerning processor allocation are described in appendix C.

Printing the Trace Table

Printing of the GISMO trace table is part of the function of the analyzer (the SYSTEM/IDA program). Thus, a system memory dump is required to capture the trace table contents. The analyzer causes the table entries to be printed in a readable format. It combines multiple entries onto single print lines for ease of use, where applicable, and prints the trace table, maintained by GISMO in a "wrap-around" fashion, in chronological order with the most recent entries at the end of the printout.

The output listing of the GISMO trace table produced by the analyzer contains four columns of information. Column contents are as follows:

Column Identification	Contents
MASTER...SLAVE EVENT	Name of the event traced An event on the slave is preceded by "..."
CHANNEL	Channel affected
REF-ADDR	Address of the I/O descriptor
DESCRIPTION	Further information and parameters

INVOKING MICRO-MCP/DEBUG

MICRO-MCP/DEBUG is the debug version of the Micro MCP. It is invoked by entering the following CM system commands and then performing a CLEAR/START operation.

```
CM MMX MCPII/MICRO-MCP
CM MM MICRO-MCP/DEBUG
```

The debug version of the Micro MCP is placed into the standard Name Table entry. The non-debug version is placed into the experimental Name Table entry and is available for restoring the system to a non-debug state.

SECTION 2
THE SYSTEM/IDA PROGRAM

The Interactive Dump Analyzer (SYSTEM/IDA) program replaces the ISSA program and the DUMP/ANALYZER program. Neither of those programs is included in the Mark 12.0 B 1000 System software release.

The SYSTEM/IDA program analyzes system memory dump files, program dump files, and programs within system memory dump files.

The PM system command causes the SYSTEM/IDA program to transform the SYSTEM/DUMPFILE file into a packaged dump file that includes the contents of system memory, layout tables, and code segment comparison error information. Layout tables contain the SDL2 data declarations used by SYSTEM/IDA to interpret the contents of memory. Packaging the SYSTEM/DUMPFILE file is essential if later analysis or transmission of the file to another site for analysis is contemplated. Only SMCP, NDL, and DMS access routine layouts are added to the packaged dump file.

Direct execution of the SYSTEM/IDA program allows interactive analysis of selected portions of a dump file. Direct execution also enables a formatted analysis of the entire dump file or portion of it to be printed.

SYSTEM/IDA commands belong to two categories, control commands and object commands. The object commands may be further subdivided into two types: program object commands and system object commands.

Control commands are active; in addition to returning information, they may be used to change the current job, current environment, current file, and so forth. Object commands are passive; they return information but have no influence on subsequent actions.

Multiple commands separated by semicolons may be included in a single transmission.

Example:

```
GET SYSTEM/DUMPFILE; PRINT IOAT; PRINT CHANNELS; PRINT DISK
```

All commands are described under the headings Control Commands, Program Object Commands, and System Object Commands in this section.

USING SYSTEM/IDA

The SYSTEM/IDA program may be initiated from a remote terminal as well as from the ODT.

Initiation from a Remote Terminal

There are three ways to initiate a run of the SYSTEM/IDA program from a remote terminal: (1) EXECUTE, a program control instruction, (2) PASS, an SMCS command, or (3) ON, an SMCS command.

Using EXECUTE

The user transmits EXECUTE (or EX), receives the BOJ, REMOTE FILE OPENED, and Welcome messages, transmits a GET command, and receives the output from the GET command.

Example:

```
EX SYSTEM/IDA
  SYSTEM/IDA =1900 BOJ. PP=4, MP=4 TIME =13:19:31.7
  REMOTE FILE OPENED BY "SYSTEM/IDA",  SIGNAL = *
  -- Ready for INPUT (type HELP for help) --
GET SYSTEM/DUMPFILE
  [output resulting from entry of GET command]
```

Using PASS

The PASS command may be used if the SYSTEM/IDA program has been entered in the SMCS Jobs file. The user transmits the PASS command with a GET command appended. A MESSAGE QUEUED message followed by the output from the GET command is received.

Example:

```
PASS SYSTEM/IDA GET SYSTEM/DUMPFILE
  MESSAGE QUEUED FOR "SYSTEM/IDA": WAITING OPEN
  [Output resulting from entry of GET command]
```

Using ON

The ON command may be used if the SYSTEM/IDA program has been entered in the SMCS Jobs file. The user transmits the ON command with a GET command appended. A Welcome to the SYSTEM/IDA program message followed by the output from the GET command is received.

Example:

```
ON SYSTEM/IDA GET SYSTEM/DUMPFILE
  -- Welcome to SYSTEM/IDA --
```

Initiation from the ODT

There are three ways to run the SYSTEM/IDA program from the ODT. One way is by entry of the PM system command. The other two ways are two different modes of using the EXECUTE program control instruction.

Using PM

The user transmits a PM command with a FILE program control instruction appended, directing the packaged dump file to a user disk.

Example:

```
PM; FILE PM NAME PACKX/PACKAGE/DUMPFILE
```

Using EXECUTE

There are two ways of using the EXECUTE program control instruction from the ODT.

1. The user transmits an EXECUTE (EX) command with a FILE program control instruction to rename the DUMPFILE file as the previously packaged system dump file. An AC program control instruction containing a pair of SYSTEM/IDA program control commands is included to print the disk descriptor chain and then terminate.

Example:

```
EX SYSTEM/IDA; FILE DUMPFILE NAME PACKX/PACKAGE/DUMPFILE;
AC PRINT DISK;BYE
```

2. The user transmits an EX command and receives a BOJ and a Welcome to SYSTEM/IDA message. The SYSTEM/IDA program waits for transmission of an AC or AX system command containing an SYSTEM/IDA command and then displays the requested information.

Example:

```
EX SYSTEM/IDA;
  SYSTEM/IDA =1566 BOJ. PP=4, MP=4 TIME = 09:45:10.6
  % SYSTEM/IDA =1566 -- Welcome to SYSTEM/IDA --
```

Scroll Mode

The SYSTEM/IDA program enters scroll mode when a SYSTEM/IDA command is given to display information that requires more than one display page. In this mode, the OPTION LINESPERSCREEN control command is enabled.

The commands listed in table 2-1 are recognized in the scroll mode. All other commands terminate scroll mode.

Table 2-1. SYSTEM/IDA Scroll Mode Commands

Command	Function
+ [<increment>] – [<decrement>]	Move forward (backward) one page or the number of lines specified by <increment> (<decrement>).
1	Display the first page.
$	Display the last page.
<line number>	Display the page that begins at <line number>.
HELP	Display the current and ending line number and the command menu.
OPTION (O)	Display the current option settings.

CONTROL COMMANDS

Control commands are used to control the execution of the SYSTEM/IDA program. They may be used to select a current version of a current job, current environment, current file, and so forth, as well as to provide formatted displays of portions of the dump file and to to terminate execution of either a single command or the overall SYSTEM/IDA program. They also provide user assistance by enabling the display of a command menu as well as syntaxes of individual commands.

Table 2-2 is a list of control commands that may be used to select items for display. Following the table are the individual control command descriptions.

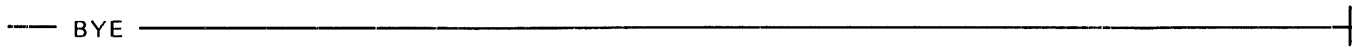
Table 2-2. Control Commands for Use in Displaying Items

Command	Item Selected
GET	Dump file
JOB	Job and environment
ENVIRONMENT	Environment of current job
FILE	File of current job
LAYOUT	Procedure frame and variable of current environment
MEMORY	Memory address in data for current environment
OPTION	Options

BYE

The BYE control command terminates the SYSTEM/IDA program. The dump file is not removed.

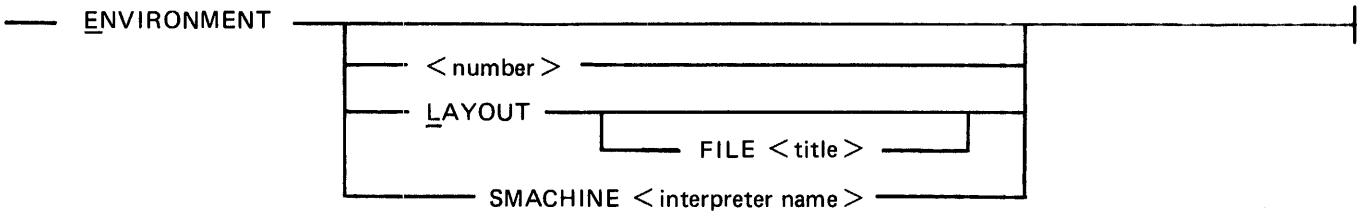
Syntax:



ENVIRONMENT

The ENVIRONMENT (E) control command selects a current environment and displays information about the environments of the current job. If no parameters are included, the environment dictionary for the current job is displayed.

Syntax:



Semantics:

<number>
This field is used to select a current environment.

LAYOUT
The LAYOUT parameter specifies that the layout table is to be loaded from the codefile named in the Run Structure Nucleus (RSN) for the current job, rather than from the dump file. This capability is useful when entering the LAYOUT command results in the message:

** Error: Job <Number>: LAYOUT TABLES NOT PRESENT

If FILE <title> is included, the layout table is to be loaded from the codefile specified by <title> rather than from the dump file. The <title> must be in the form A/B ON C.

SMACHINE <interpreter name>
SMACHINE <interpreter name> specifies the first name of the interpreter for the current job. It is entered to facilitate analysis of a program that used an interpreter with a non-standard first name.

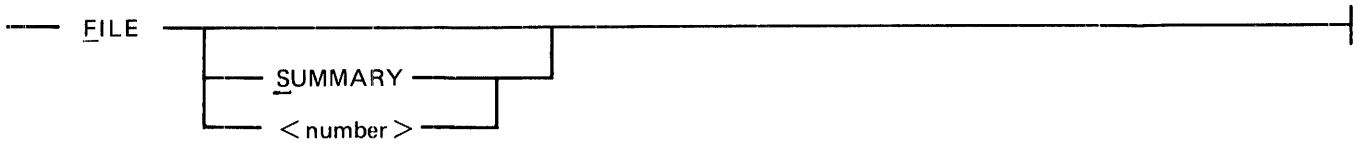
Examples:

E 1
E LAYOUT

FILE

The FILE control command selects a current file and displays information about files associated with the current job. If no parameter is included, the first page of the File Information Block (FIB) of the current file is displayed.

Syntax:



Semantics:

SUMMARY or S
The SUMMARY (or S) parameter causes the file dictionary to be displayed.

<number>
This field is used to select a current file and causes the first page of the FIB to be displayed.

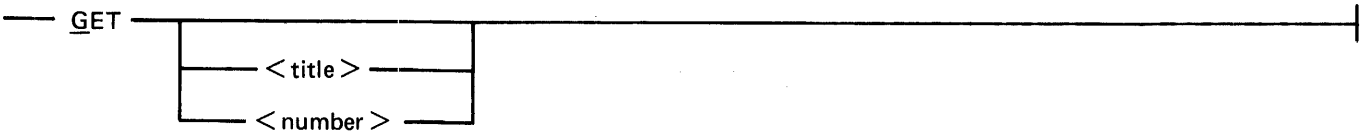
Examples:

```
F S
FILE 2
```

GET

The GET control command selects a current dump file, selects the first job as the current job, and displays some general information about the dump file. If a system dump file is selected, the MCP (JOB 0) is selected as the current job. If no parameter is included, general information about the currently selected dump file is displayed.

Syntax:



Semantics:

<title>
This field is used to select a current dump file. The SYSTEM/IDA program determines whether it is a system dump file, program dump file, or packaged system dump file, and performs version checking. <title> must be in the form A/B ON C.

<number>
When this field has a zero value, the SYSTEM/DUMPFILE file is selected as the current dump file. When the field has a non-zero value, DUMPFILE/<number> is selected as the current dump file.

If a prior dump file had been selected, it is closed but not removed.

Examples:

```
G 0
GET DUMPFILE/PM ON S
```

HELP

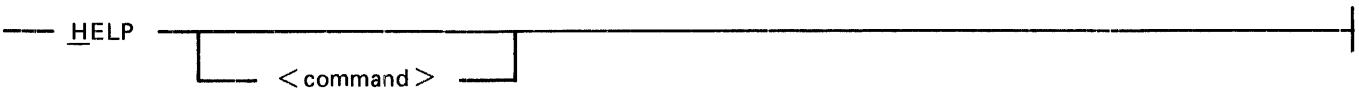
The HELP control command displays either the menu of commands or the syntax of the specified command.

If the SYSTEM/IDA program is in the scroll mode, the following message is included, where <current> is the number of the current line and <last> is the number of the last line in the scroll buffer.

YOU ARE SCROLLING AT LINE <current> OF <last>

If no parameter is included, the menu of commands is displayed, along with the current patch level compile date.

Syntax:



Semantics:

<command>
This field may contain any control command, program object command, or system object command.

Examples:

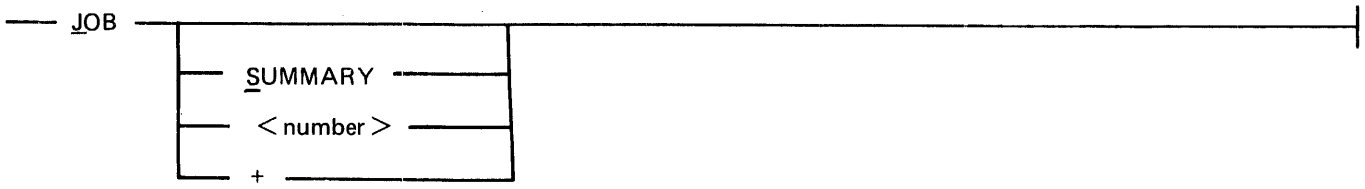
```
HELP
HELP LAYOUT
```

JOB

The JOB control command selects a current job and a current environment and displays information about the job. This command is only available in system dumps. The job may be either the MCP or one of the jobs in the mix.

If no parameter is included, the mix summary is displayed.

Syntax:



Semantics:

SUMMARY
The SUMMARY parameter displays the mix summary.

<number>
This field is used to select a current job. The SYSTEM/IDA program selects the active environment for that job as the current environment and displays the state of the job.

When this field has a zero value, the MCP is selected.

When this field has a non-zero value, a job in the mix is selected.

+
The + parameter is used to select the next job as the current job. If the MCP was the current job, the first job in the mix is selected. If a job in the mix was the current job, the next job in the mix is selected.

Examples:

```
J
J 581
J +
J 0
J S
```

LAYOUT – SDL2 Program

The LAYOUT control command is used to select a current procedure frame, to select a current variable, and to display a history of procedure calls and information about the variables and arrays accessible to the procedures of the current job.

The layout tables are a representation of the variable and array declarations. They provide the template for the SYSTEM/IDA program to interpret the contents of a dump file.

This command includes the concept of a current procedure frame and a current variable. A current procedure frame is selected by transmitting LAYOUT <number>. A current variable is selected by transmitting LAYOUT VAR <name>. If no current variable has been selected, subsequent LAYOUT commands analyze all variables in a frame.

The LAYOUT command does many things, depending on the state of the analysis. These things can best be described programmatically:

1. If the layout tables for the current job are not loaded, load the layout tables.
2. If a SUMMARY parameter is present, display the layout summary.
3. If a SUMMARY parameter is not present, consider the following:

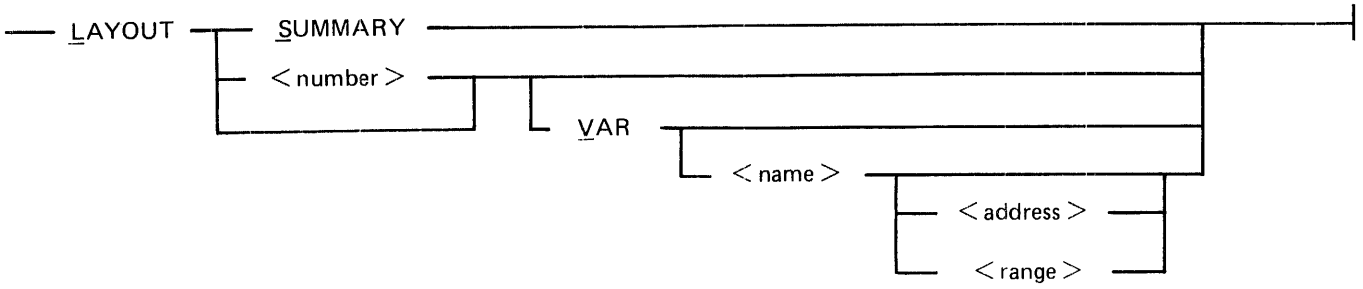
   If a <number> field is present, select a current procedure frame.

   If a VAR <variable name> is present, select a current variable. If VAR parameter is present without a variable name, clear the current variable.

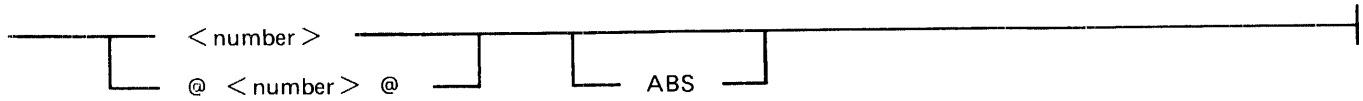
   If there is a current variable search for the current variable within the current procedure frame, display the value. If the ADDRESSES switch is set, the addresses are displayed.

If the SIMPLETYPES switch option is on, it is temporarily turned off for variable analysis by the LAYOUT VAR form of this control command.

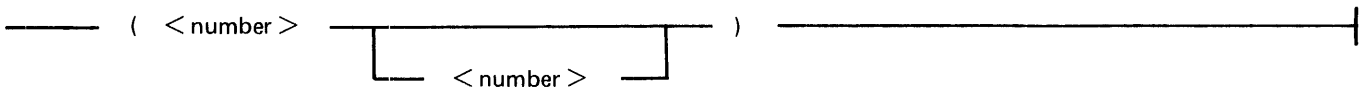
Syntax:



Syntax for <address>:



Syntax for <range>:



Semantics:

SUMMARY
The SUMMARY parameter displays a history of procedure calls that shows the flow of control from the global procedure to the last procedure called.

<number>
This field is used to select a current procedure frame.

VAR
This parameter is used to select a current variable and turn on the search mode, or to clear the current variable and turn off the search mode.

<name>
When this field is present, a current variable (or array) is selected.

When this field is not present, the current variable is cleared.

<address>
When this field is present, the <name> field is used as a template to interpret the contents of location <address>.

The <address> field is a decimal address if it is not bounded by at-sign (@) characters. It is a hexadecimal address if it is bounded by at-sign characters.

The <address> field is relative to the base register of the current job if it is not followed by the ABS keyword. It is an absolute address if it is followed by the ABS keyword.

<range>
When this field is present, it is used to specify a single array element or a range of array elements to be displayed.

When a single number is included, it specifies the subscript of the array element to be displayed.

When two numbers are included, they specify the subscripts that bound the range of array elements to be displayed.

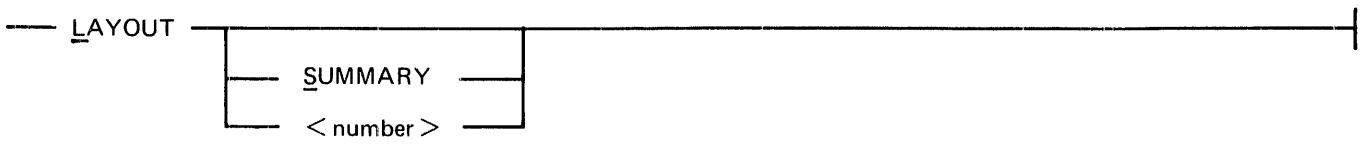
Examples:

```
L S
L 6
L V
L V CSV
L 0 V HINTS 0 ABS
L V TABLE (1 10)
```

LAYOUT – WFL Program

A subset of the LAYOUT control command is available for Work Flow Language (WFL) program analysis. The VAR parameter is not available.

Syntax:



Semantics:

SUMMARY
The SUMMARY parameter displays a history of procedure calls that shows the flow of control from the global procedure to the last procedure called.

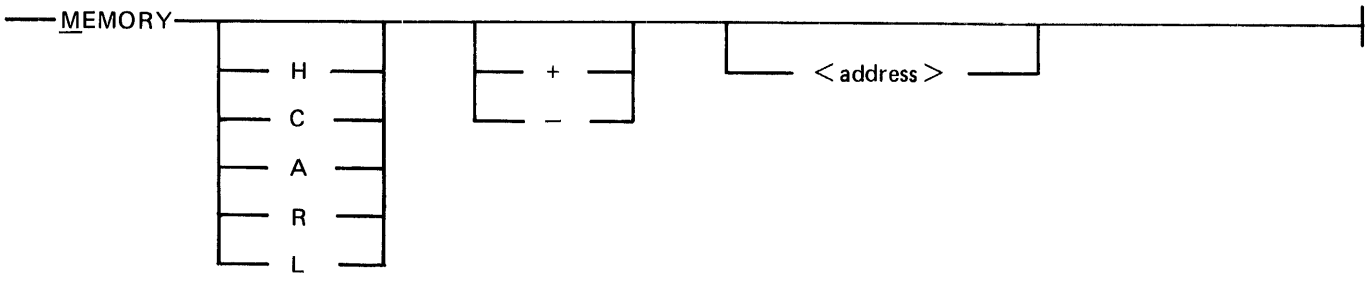
< number >
This field is used to select a current procedure frame.

MEMORY

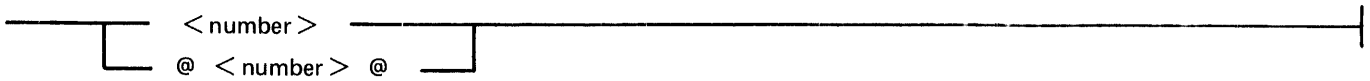
The MEMORY control command selects a current memory address, displays an unformatted analysis of memory, and displays a formatted analysis of a memory link.

If no parameter is included, the display shows the contents of memory at the current address.

Syntax:



Syntax for <address>:



Semantics:

address
This field establishes a current address. The address may be specified in decimal or hexadecimal notation. Hexadecimal addresses are enclosed in at sign (@) characters.

H
The H keysymbol specifies that memory is to be displayed in hexadecimal representation.

C
The C keysymbol specifies that memory is to be displayed in character representation.

A
The A keysymbol specifies that all subsequent addresses are absolute. This is the default mode for system dump files and invalid for program dump files.

R
The R keysymbol specifies that all subsequent addresses are relative to the base register of the current job. This is the default mode for program dump files.

L
The L keysymbol displays the system memory link for the current address (system dumps only).

+
The + keysymbol increments the current address by either one page or by the number of bits specified by the following <address>.

–

The – keysymbol decrements the current address by either one page or by the number of bits specified by the following <address>.

L +

The L + keysymbols change the current address to the address of the forward memory link and display the link.

L –

The L – keysymbols change the current address to the address of the backward memory link and display the link.

Examples:

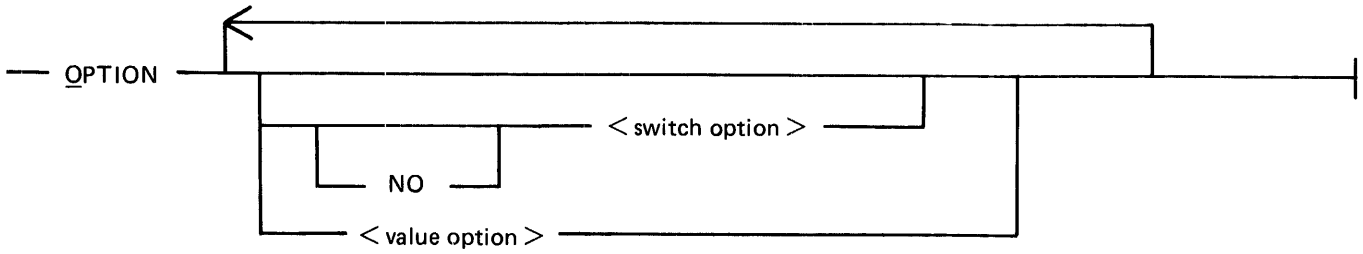
```
M C 1400
M H + 100
M R 0
M C +
M L @345678@
```

OPTION

The OPTION control command allows the entry of options for displaying information. The entry of OPTION alone displays the current option settings. The options are of two types, switch options and value options. See Switch Options and Value Options.

For further information on options, see Switch Settings, later in this section.

Syntax:



Semantics:

NO
Turns off the switch option that follows.

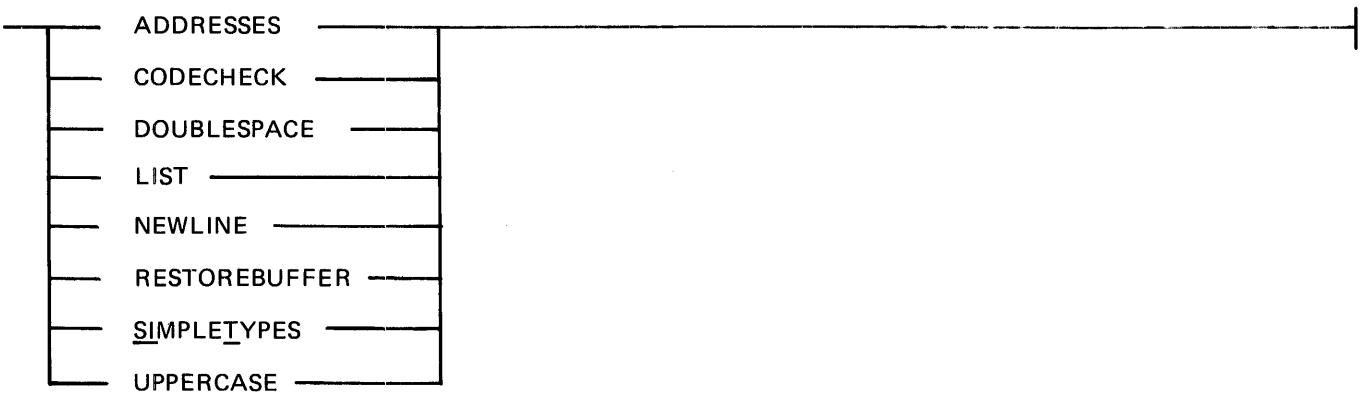
<switch option>
See Switch Options.

<value option>
See Value Options.

Switch Options

All switch options except UPPERCASE and CODECHECK are initially off.

Syntax:



Semantics:

ADDRESSES
The ADDRESSES switch option adds the memory addresses of variables to the output of the LAYOUT control command.

CODECHECK
The CODECHECK switch option enables comparison of code files in the dumpfile with the corresponding code files on disk by the PM command.

DOUBLESPACE
The DOUBLESPACE switch option specifies that subsequent printer output is to be double spaced.

LIST
The LIST switch option specifies that all succeeding output is to be written to the LINE file in addition to the USER file or ODT.

NEWLINE
The NEWLINE switch option specifies that the cursor is to be left at the start of the second line instead of at the HOME position. This capability facilitates repeated + commands when in scroll mode.

RESTOREBUFFER
The RESTOREBUFFER switch option restores the prior scroll buffer if any.

SIMPLETYPES
The SIMPLETYPES (may be abbreviated ST) switch option suppresses analysis of record fields and array elements by subsequent LAYOUT commands.

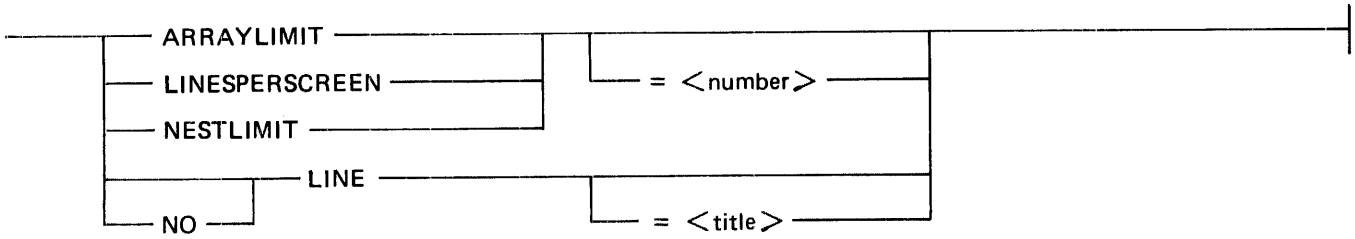
Note that if the SIMPLETYPES switch option is on, it is temporarily turned off for variable analysis by the LAYOUT VAR form of the LAYOUT control command.

UPPERCASE
The UPPERCASE switch option specifies that output is to be in uppercase only.

Value Options

Each value option has a default value, specified in its description.

Syntax:



Semantics:

ARRAYLIMIT

The ARRAYLIMIT value option either interrogates or changes the number of array elements analyzed by the LAYOUT command. The default number of array elements analyzed is 64. If <number> is not specified, the current value of the option is interrogated and displayed. The <number> parameter is used to change the number of array elements analyzed.

LINESPERSCREEN

The LINESPERSCREEN value option is valid only in scroll mode. The option either interrogates or changes the number of lines per page in scroll mode. The default value of the option is 22. If <number> is not specified, then the current value of the option is interrogated and displayed. The <number> parameter is used to change the number of lines per page.

NESTLIMIT

The NESTLIMIT value option controls the format of variables analyzed by the LAYOUT command by restricting analysis to a maximum nest level.

LINE

The LINE value option either interrogates or changes the title of the LINE file. The default title of the LINE file is LINE. If <title> is not specified, the current title of the LINE file is interrogated and displayed. The <title> parameter is used to change the title of the LINE file. The <title> must be in the form A/B ON C. A NO entry preceding the LINE option closes the LINE file.

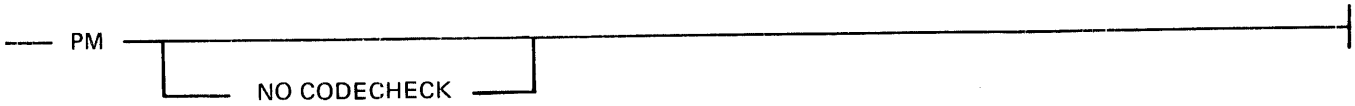
PM

The PM control command reads the current dump file and writes the PM file in order to produce a packaged dump file. The default name of the PM file is DUMPFILE/PM<nnn>, where <nnn> is the next BACKUP file number assigned by the system.

The default name of the PM file can be overridden by inclusion of a FILE program control instruction when the SYSTEM/IDA program is executed.

A packaged dump file is used when the analysis is not scheduled to be performed immediately or when analysis is scheduled to be performed on a different system.

Syntax:



Semantics:

NO CODECHECK
The NO CODECHECK keywords eliminate checking for code segment comparison errors between the code segments in the dump file and the corresponding code segments on disk. This option should not be used for dumps submitted with Field Communication Forms (FCF).

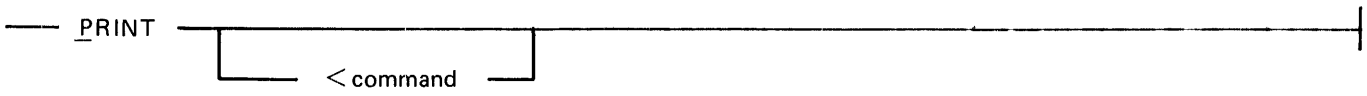
Examples:

```
PM
PM NO CODECHECK
```

PRINT

The PRINT control command either writes a formatted analysis of the current dump file or writes the output of the specified command to the LINE file.

Syntax:



Semantics:

<command>
This field may contain any command except PRINT. All output associated with the command is written to the LINE file.

Pragmatics:

When certain commands are included with no parameters, their normal display output is enhanced. Those commands and the corresponding enhancements follow.

Inclusion of the FILE command prints the file dictionary and the FIB for each open file.

Inclusion of the JOB command prints the job summary and the state of each job in the mix.

Inclusion of the LAYOUT command prints the layout summary and the contents of each procedure frame.

Inclusion of the MEMORY command prints an unformatted analysis of memory for the current job, along with any memory links present.

Inclusion of the SMACHINE command for SDL2 job prints the SDL2 S-machine summary and the output from the CONTROL, DISPLAY, LAYOUT, NAME, PROCEDURE, and VIRTUAL parameters.

Examples:

```
P
P L S
P DISK
P JOB 567
```

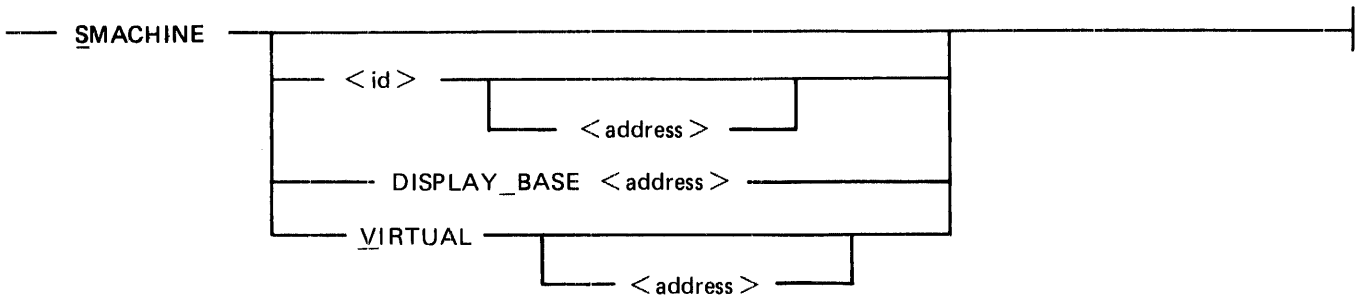
SMACHINE

The SMACHINE control command displays SMACHINE information about the current environment of the current job.

For SDL2 environments, if no parameter is included, a SDL2 S-machine summary is displayed giving the addresses of the value, evaluation, procedure, and control stacks and of the name and display arrays, followed by a brief analysis of any virtual memory usage.

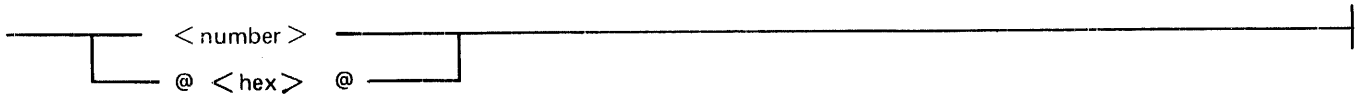
For non-SDL2 environments, no parameters are used. The local data area is analyzed, as it was in the past, by the DUMP/ANALYZER program.

Syntax:



Syntax for <id>:

Syntax for <address>:



Semantics:

CONTROL
The CONTROL parameter displays a formatted analysis of the control stack. The <address> field may be used to change the address of the control stack when the starting address has been corrupted.

DISPLAY
The DISPLAY parameter displays a formatted analysis of the display array. The <address> field may be used to change the address of the display array when the starting address has been corrupted.

LAYOUT
The LAYOUT parameter displays a formatted analysis of the data described by the name array entries.

NAME
The NAME parameter displays a formatted analysis of the name array. The <address> field may be used to change the address of the name array when the starting address has been corrupted.

PROCEDURE
The PROCEDURE parameter displays a formatted analysis of the procedure stack. The <address> field may be used to change the address of the procedure stack when the starting address has been corrupted.

DISPLAY_BASE <address>
The DISPLAY_BASE parameter changes the address of the base of the display array to <address>, thus changing the procedure stack base as well.

VIRTUAL
The VIRTUAL parameter displays a formatted analysis of virtual memory.

VIRTUAL <address>
When the VIRTUAL parameter is followed by an <address> field, a formatted analysis of the memory link at <address> is displayed.

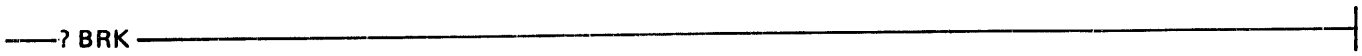
Examples:

```
S P
S VIRTUAL
S V @A5B6C7@
```

?BRK

The ?BRK control command terminates processing of the current command.

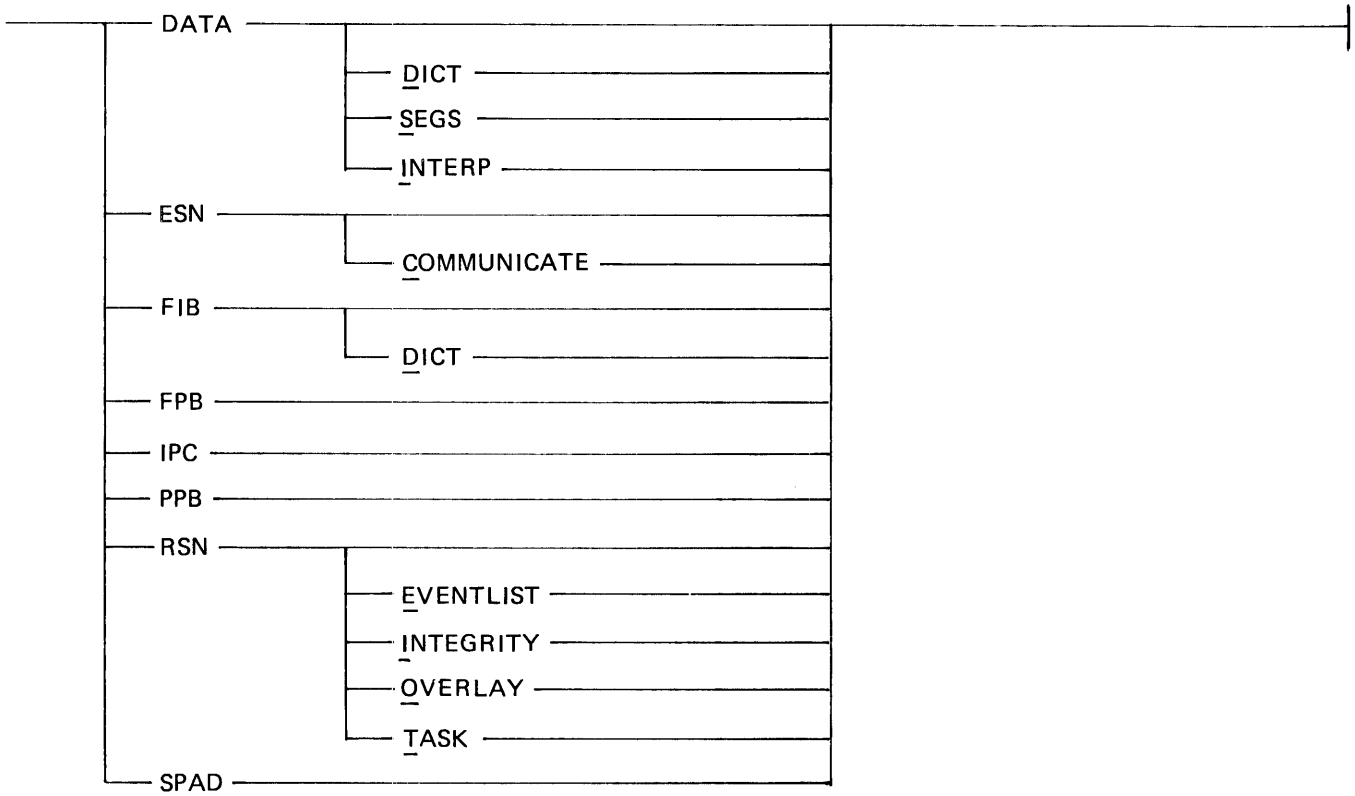
Syntax:



PROGRAM OBJECT COMMANDS

Program object commands display data that is available both in program dump files and in system dump files. Program object commands that require more than one display page force SYSTEM/IDA into scroll mode. All program object commands apply to the current job.

Syntax:



DATA

Displays the Data Dictionary, Data Segments, or Interpreter Data for the current environment. Data Segments are only available in a program dump.

ESN

Displays the Environment Structure Nucleus for the current environment.

ESN COMMUNICATE
Displays the Communicate message for the current environment.

FIB

Displays the File Information Block for the current file. The FIB command displays the same information as the FILE <number> control command.

FIB DICT
Displays the FIB Dictionary. The FIB DICT command displays the same information as the FILE SUMMARY command.

FPB

Displays the File Parameter Block for the current file. The FPB command is only available in program dump files.

IPC

Displays the Inter-Program Communication information for the current job.

PPB

Displays the Program Parameter Block.

RSN

Displays the Run Structure Nucleus.

RSN EVENTLIST
Displays the event list.

RSN OVERLAY
Displays the overlay descriptor.

RSN INTEGRITY
Displays the results of a RSN integrity check.

RSN TASK
Displays the task variable table.

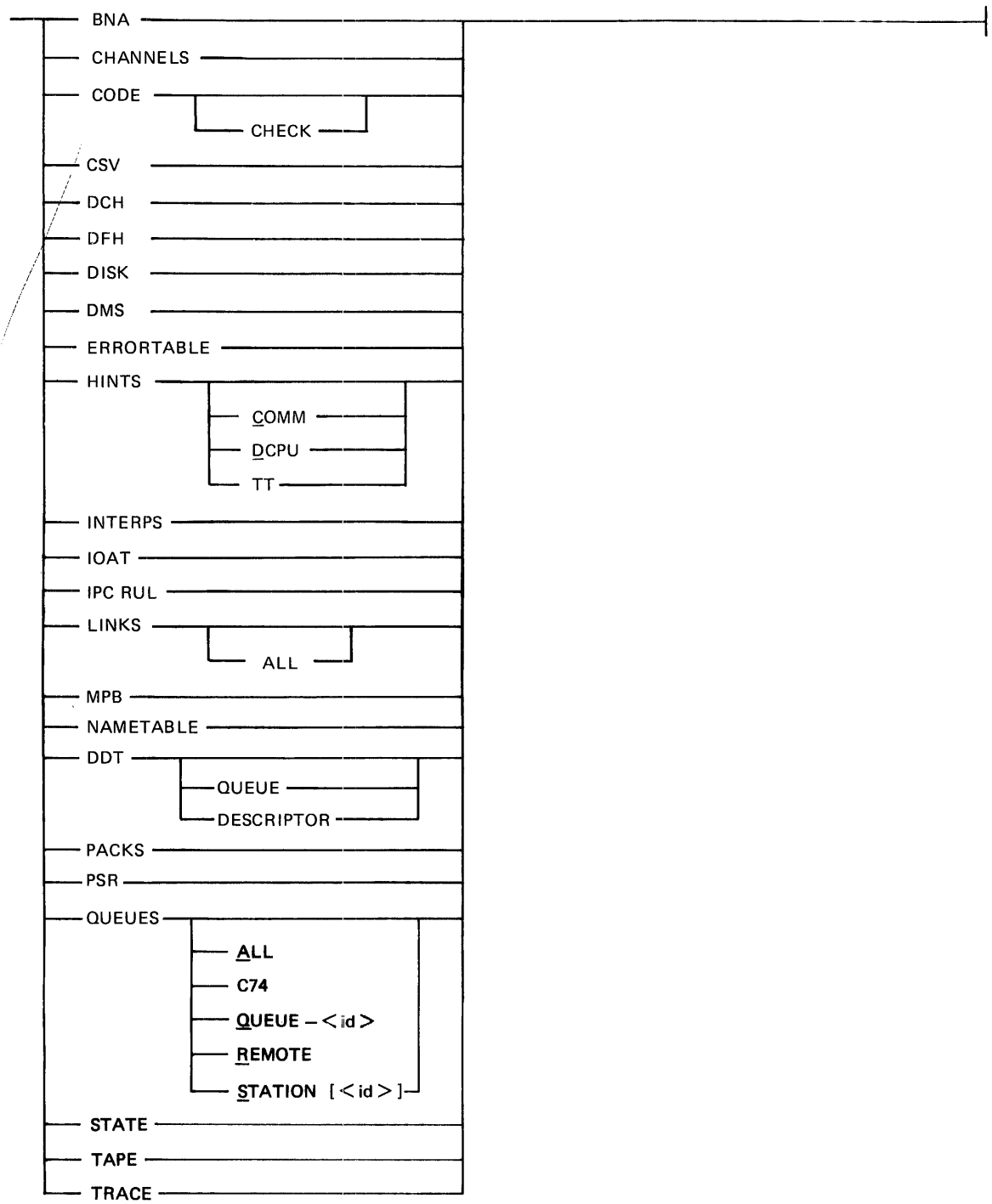
SPAD

Displays the Scratchpad for the current environment.

SYSTEM OBJECT COMMANDS

System object commands display data that is available only in system dump files. System object commands that require more than one display page force SYSTEM/IDA into scroll mode.

Syntax:



BNA

Displays the BNA memory area.

CHANNELS

Displays the I/O channel tables.

CODE

Displays the code dictionary for the current environment.

CODE CHECK
Displays the code comparison errors.

CSV

Displays the cold start variables.

DCH

Displays the active data communication channels and network controller information.

DFH

Displays the disk file header dictionary.

DISK

Displays the disk descriptor chain and extended result descriptor chain.

DISK S

Displays only disk descriptors those not yet completed and those in error.

DMS

See DMS Commands.

ERRORTABLE

Displays the correctable error table.

HINTS

Displays the Hints record.

HINTS COMM
Displays the communicate splitter mask for routing program communicate messages to either the SMCP or the MMCP.

HINTS DCPU
Displays the DCPU information and the lock management data.

HINTS TT
Displays the truth table for marking the patch level of the SMCP.

INTERPS

Displays the Interpreter Dictionary.

IOAT

Displays the Input-Output Assignment Table (IOAT).

IPC RUL

Displays the Inter-Program Communication Run Unit List.

LINKS

Displays a general analysis of system memory links, a memory usage summary, and a specific analysis only of bad system memory links.

LINKS ALL
Displays a general analysis of system memory links and a specific analysis of each system memory link.

MPB

Displays the MCP Parameter Block.

NAMETABLE

Displays the name table for system software names and disk addresses.

ODT

Displays the ODT queue starting with the most recent entries.

ODT QUEUE
Displays the same information as the ODT command.

ODT DESCRIPTOR
Displays the ODT descriptor chain, the ODT buffer, and the ODT/SQUASH globals.

PACKS

Displays the disk cartridge/pack information tables.

PSR

Displays the Pseudo Reader Information.

QUEUES

Displays the queue information global parameters and queue descriptors.

QUEUES ALL
Displays all queue descriptors.

QUEUES C74
Displays only COBOL74 datacomm queue descriptors.

QUEUES QUEUE-<id>
Displays all queues with the given multifile-id.

QUEUES REMOTE
Displays only file queue descriptors.

QUEUES STATION [LSN]
Displays station queues. If an LSN is specified, displays that station only.

STATE

Displays the processor state including the interrupt queue, the GISMO work area, the master and slave processor scratchpad registers and A-stack, and the master and slave processor MMCP data.

TAPE

Displays all tape descriptor chains.

TRACE

Displays the GISMO trace table.

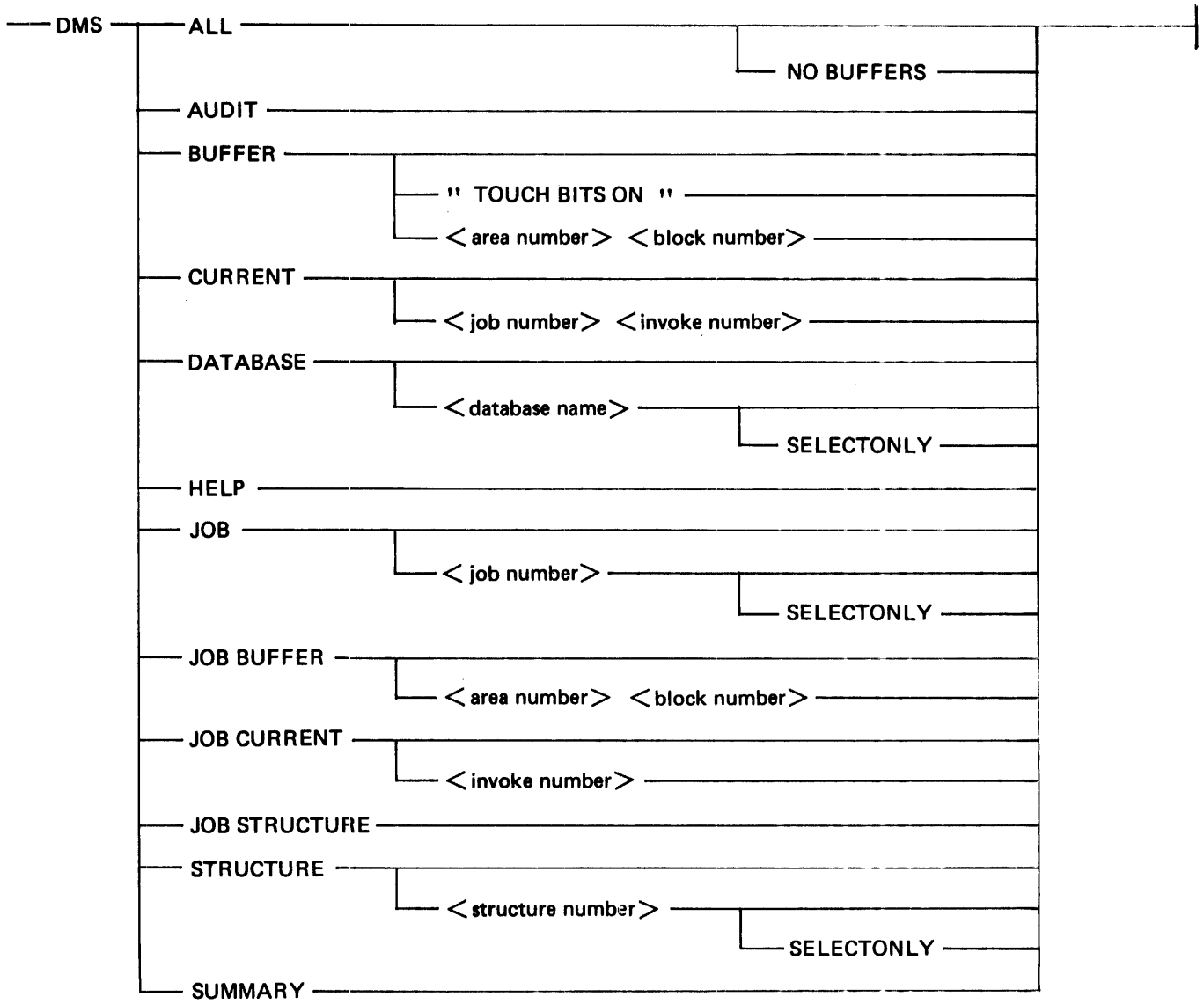
DMS COMMANDS

DMS alone was the only valid DMS command in the previous release. Now, DMS entered alone returns the following message:

WARNING the DMS command is no longer valid. Try DMS HELP

For the 12.0 release, there are twelve DMS commands, as shown in the syntax diagram and the descriptions that follow it.

Syntax



DMS ALL

For each opened data base, this command displays the DMS globals and the audit FIB as well as the two I/O descriptors belonging to the data base. For each opened structure, the following is displayed:

- The structure record.
- All the current records.
- If SWITCH2 is less than 3, all the buffer descriptors. Moreover, if the structure is an index (INDEX SEQUENTIAL or INDEX RANDOM), the tail part of each buffer is displayed.
- If SWITCH2 equals 0, and NO-BUFFERS has not been specified in the DMS command, and a print has been requested (PRINT DMS ALL), all the buffers are printed in hexadecimal.

DMS AUDIT

The audit command displays the audit FIB and the memory address of the buffers. If disk is the audit file medium, the disk file header is also displayed.

DMS BUFFER

To use this command, a data base and a structure must first be selected. (See the DMS DATABASE and DMS STRUCTURE commands.)

DMS BUFFER
Displays, for each buffer belonging to the selected structure, the logical address, the touch bits, the user count, and the status flags: media (Bd__in__memory), to be written, control point, and I/O error. Figure 2-1 is an example of the display.

DMS BUFFER "TOUCH BITS ON"
Displays all buffers with touch bits not equal to zero.

DMS BUFFER <area number> <block number>
Displays the front part of the buffer (Buffer__descriptor) and the address of the buffer itself. If the structure is an index, the tail part of the buffer (Buffer__end__descriptor) is also displayed.

To see the contents of the buffer, enter the MEMORY control command. (See MEMORY, in this section.)

DMS CURRENT

To use this command, a data base and a structure must first be selected. (See the DMS DATABASE and DMS STRUCTURE commands.)

DMS CURRENT
Displays each current with its job number, invoke number, and locks. Record number is relevant for lists and datasets only. Entry number is relevant for lists and indexes only. Figure 2-2 is an example of the display.

DMS CURRENT <job number> <invoke number>
Displays the entire current.

DMS DATABASE

Displays each data base with a list of the jobs using it. For each job listed, the full name (pack id, mfid, fid) and the job number are displayed. Figure 2-3 is an example of the DMS DATABASE display.

DMS DATABASE <database name>
Displays a single data base and provides additional information about it. The pack name is not needed because DMS prohibits two data bases with the same name, even if they are on different packs. If the data base name contains odd characters, for example: 2MWDB, the data base name must be enclosed in quotes.

The named data base is selected as context for subsequent commands, and the DMS globals record is displayed.

DMS DATABASE <database name> SELECTONLY
Allows the selection of a data base without the DMS globals record. The following is displayed: DATABASE <database name> SELECTED

```
"              to be written control point    "
"              media_____  ! !  _I/O error    "
"<--addr--> touch     ! ! ! !                 "
"area block  bits user V V V V                "
"-------------------------------------------  "
"123  12345  @11@  16  Y N Y N                "
"012  01234  @01@  01  N N N N                "
"001  00123  @00@  03  Y Y Y Y                "
```

Figure 2-1. Example of DMS BUFFER Display

```
"                                                          restart lock__  "
"                                                          record lock_ !" 
"                                                          user lock_  ! !"
"                                                                    ! ! !"
"                                                                   V V V"
"          <-------user current------><---working current------->       "
"          <-------address-->points at<--address------->points at       "
"job invoke area blk rec entry         area blk rec entry                "
"-------------------------------------------------------------------------"
"1234  64   101  345  -   256  nothing 001  001  -   123   next   Y N Y"
"4321  63   001  001 010   -   current 101  345 123   -   prior   N Y N"

"                                                        user lock   ! !"
```

Figure 2-2. Example of DMS CURRENT Display

```
"             ***** Data base <database name> *****                    "
"                                                                      "
"                       user count : 5                                 "
"                 update user count : 3                                "
"                           updated : YES                              "
"                      programs ok  : YES                              "
"                 recovery in process : NO                             "
"              reorganization running : NO                             "
"                        globals lock : NO                             "
"                          audit lock : YES                            "
"                     transaction lock : YES                           "
"                 recovery in process : NO                             "
"           reorganization in process : NO                             "
"<---job name-----> job nbr                                            "
"--------------------------                                            "
" a/b/c               1234                                              "
" d/e/f               4321                                              "
```

Figure 2-3. Example of DMS DATABASE Display

DMS HELP

This command is context sensitive; that is, its output depends on preceding commands. Syntax as well as instructions are provided.

DMS JOB

To use this command, a data base must first be selected. (See the DMS DATABASE command.)

DMS JOB
Displays the names and numbers of the jobs using the selected data base. For each job, the full name (pack id, mfid, fid), number, actual environment, job status, and relevant flags are displayed. Figure 2-4 is a sample display. If the job is rolled out, the flags are not displayed; instead, the message "Job Rolled Out" is displayed.

DMS JOB <job number>
Displays the DMS interface area of the specified job.

DMS JOB <job number> SELECTONLY
Selects the specified job, without displaying the interface, and responds with

JOB <job number> SELECTED

DMS JOB BUFFER

To use this command, a data base, a job, and a job structure must first be selected. (See the DMS DATABASE, DMS JOB, and DMS JOB STRUCTURE commands.) The command is similar to DMS BUFFER but operates in a different context.

DMS JOB BUFFER
Displays all the buffers belonging to the currents for the selected job and job structure. For each buffer, the logical address, the touch bits, the user count, the flags, and the entry number are displayed. For indexes, only the entry number is relevant; for lists, both record number and entry number are relevant.

DMS JOB BUFFER <area number> <block number>
Displays the contents of the buffer specified.

DMS JOB CURRENT

To use this command, a data base, a job, and a job structure must first be selected. (See the DMS DATABASE, DMS JOB, and DMS STRUCTURE commands.) This command is similar to the DMS CURRENT command but operates in a different context.

DMS JOB CURRENT
Displays all the currents and, for each current, the invoke number, the logical address (consisting of the area number, block number, and record number), the entry number, and the locks.

DMS JOB CURRENT <invoke number>
Displays the entire current.

DMS JOB STRUCTURE

To use this command, a data base and a job must first be selected. (See the DMS DATABASE and DMS JOB commands.) The DMS JOB STRUCTURE and DMS STRUCTURE commands are similar but work in different contexts.

DMS JOB STRUCTURE
Displays all the structures that are in use by the job specified. For each structure, the structure number, user count, update user count, structure type, parent structures, object structures, and some locks are displayed.

DMS JOB STRUCTURE <structure number>
Causes the specified structure to be selected as context for subsequent commands and displays the structure record, file record, and disk file header.

DMS JOB STRUCTURE <structure number> SELECTONLY
If the user wants to select a structure and does not want any more information, the reply is:

STRUCTURE <structure number> SELECTED

DMS STRUCTURE

To use this command, a data base must first be selected. (See the DMS DATABASE command.)

DMS STRUCTURE
Displays all opened structures, giving the structure number, user count, update user count, structure type, parent structure number, object structure number, and some locks. Figure 2-5 is an example of the display.

DMS STRUCTURE <structure number>
Displays the structure record, the file record, and the disk file header. The structure is selected as the context for subsequent commands.

DMS STRUCTURE <structure number> SELECTONLY
Selects a structure and gives the following response:

STRUCTURE <structure number> SELECTED

```
"              fatal error _____________________                "
"              in transaction_________________  !               "
"              I am aborting _______________  ! !               "
"              aborted _____________________  ! ! !             "
"              backing out_________________  ! ! ! !            "
"              updating ___________________  ! ! ! ! !          "
"              opened update _____________  ! ! ! ! ! !         "
"                                         ! ! ! ! ! ! !        "
"                                         V V V V V V V        "
"  name nbr   env <--status--->                                  "
"------------------------------------------------------------"
" a/b/c 1234 usr waiting DS/DP  Y N N N N N N                   "
" d/e/f 4321 dms executing      Y Y Y Y Y Y Y                   "
" g/h/i 0123 mcp executing      Y N Y N Y N Y                   "
                                ROLLED OUT
```

Figure 2-4. Example of DMS JOB Display

```
"<------structure-------> <--users--> <----------locks-------------->"
" type  nbr object parent all    upd  memory buffer cur_link currents"
"                                                                    "
"  MSS   4   1023   1022  10     3      Y      Y       N        N    "
" RSDDS   7   1021   1020   0     0      Y      Y       Y      Y(EX)  "
"  EDS   1   1019   1018  11    11      N      N       N      Y(12)  "
"IDXSEQ  6   1017   1016   3     1      N      Y       N      Y(02)  "

   Y(EX) means exclusive lock.
   Y(xx) means non-exclusive lock with xx users.
```

Figure 2-5. Example of DMS STRUCTURE Display

DMS SUMMARY

This command produces a display that provides an overview of what was going on when the dump was taken. The display includes the summary information for data bases, structures, and currents.

Each data base name is displayed. For each, the user count and other summary information is included, along with the locks (global, audit, and transaction).

For each data base name, all the structures in memory are shown. For each structure, the locks (buffer lock, current lock, current link lock), the type, user count, update user count, object structure, parent structure, and the currents are shown.

For each current, the job number, the invoke number, the logical address, and the user lock are shown.

Figure 2-6 is an example of the display.

```
"             ***** Data base <database name> *****                    "
"                                                                      "
"                       user count : 5                                 "
"                 update user count : 3                                "
"                           updated : YES                              "
"                      programs ok  : YES                              "
"                 recovery in process : NO                             "
"              reorganization running : NO                             "
"                        globals lock : NO                             "
"                         audit lock  : YES                            "
"                    transaction lock : YES                            "
"                                                                      "
"               ** Structure <structure number> **                     "
"     type= INDX RANDOM  user cnt=12   update user count=10            "
"     buffer lock=Y  current lock=Y(EX) current link lock=Y             "
"     object structure = 7 parent structure = -                        "
"                       * Currents *                                   "
"                                                    restart lock__    "
"                                                    record lock_  !   "
"                                                    user lock_  ! !   "
"                                                              ! ! !   "
"                                                              V V V   "
"            <-------user current------><---working current------->    "
"            <-------address-->points at<--address------->points at    "
"job invoke area blk rec entry          area blk rec entry             "
"-------------------------------------------------------------------   "
"1234  64    101  345  -   256  nothing 001  001  -   123   next   Y N Y"
"4321  63    001  001 010   -   current 101  345 123   -   prior   N Y N"

   Y(EX) means exclusive lock.
   Y(xx) means non-exclusive lock with xx users.
```

Figure 2-6. Example of DMS SUMMARY Display

SWITCH SETTINGS

Following are the SYSTEM/IDA program switch settings. In all cases, zero is the default setting.

Switch	Settings
0	0 = Analyze only the first 64 elements of each array. 1 = Analyze all elements of each array. 2-15 = Do not analyze arrays.
1	0 = Allow comparison of resident code, interpreter, and microcode segments to their copies on disk. 1-15 = Suppress code segment comparison.
2	0 = Display DMS and ISAM buffers. 1 = Suppress DMS and ISAM buffer data only. 2-15 = Suppress DMS and ISAM buffer descriptors and data.
3	0 = Suppress printing of certain (already analyzed) memory areas during the hexadecimal dump. 1-15 = Print all areas of memory during the hexadecimal dump.
4	0 = Remain in loop until exit conditions are satisfied. 1-15 = Decrement SW 4 and exit loop unconditionally. The semantics of this switch are functionally the same as those of the corresponding switch in the MCPII/ANALYZER program.
5	0 = Analyze all ODT queue entries. 1-15 = Analyze only the last 25 percent of the ODT queue.
6	0 = Print available memory areas during hexadecimal dump. 1 = Suppress printing of available memory areas.
7	Reserved for future use.
8	0 = If incorrect MCP level, abort analysis. 1-15 = Attempt analysis regardless of MCP level.
9	0 = Default option UPPERCASE = ON. 1 = Default options UPPERCASE and DOUBLESPACE = OFF 2 = Default options UPPERCASE and DOUBLESPACE = ON 3 = Default option DOUBLESPACE = ON 4-15 = Same as setting 1.

EXCEPTION CONDITIONS

The SYSTEM/IDA program informs the user upon the detection of corruption in the dump file. The user may then decide how much analysis is worthwhile.

When an exception situation is encountered, a message describing the problem is displayed, followed by a menu of possible user responses, and a request for input.

FILE NAMES

Table 2-3 shows the internal and external file names of the files associated with the SYSTEM/IDA program and their functions.

Table 2-3. SYSTEM/IDA File Information

Internal File Name	External File Name	Function
LINE	LINE	The printer output file.
USER	USER	The user's remote file.
DUMPFILE	SYSTEM/DUMPFILE	The current dumpfile.
DISK	IDA/DISK	A temporary work file.
PM	DUMPFILE/PM<nnn>	The packaged dump file written by the PM control command, where <nnn> is the next BACKUP file number assigned by the system. File equation to tape is allowed.
CEFILE	CEFILE	A temporary work file.
CDFILE	CDFILE	A temporary work file.
TEXTFILE	IDA/TEXTFILE	A temporary work file for scrolling.
LINKFILE	LINKFILE	A temporary work file.

SECTION 3
OPERATING SYSTEM COMPONENTS

The Burroughs B 1000 operating system is a modular supervisory program that takes charge of frequently used functions and thus simplifies and expedites the preparation and running of programs and the overall operation of the system.

The operating system consists of the following three separate components.

Component	Abbreviation
Master Control Program II	SMCP
MCPII/MICRO-MCP	MMCP
GISMO3 (or GISMO2)	GISMO

These three components perform the following functions:

- Scheduling, initiating, executing, monitoring, and terminating programs as requested by users.
- Providing a symbolic means of communicating with the system while shielding users from the details of the hardware.
- Identifying, managing, and retaining memory and disk areas for maintenance of programs and files.
- Managing the system resources for optimum utilization by many concurrent operations.

The B 1000 operating system components are programs written in the System Development Language (SDL2) and the Micro Implementation Language (MIL).

SDL2 is a high level language used for writing B 1000 system software. SDL2 program instructions are performed by the SDL2 interpreter, as opposed to being executed directly on the B 1000 computer system hardware. An example of a SDL2 program is the MCPII program, the portion of the operating system written in SDL2.

MIL is the Micro Implementation Language on the B 1000. MIL is a machine level language used for writing B 1000 system microcode. MIL programs execute directly on the B 1000 computer system hardware. An example of a MIL program is the MCPII/MICRO-MCP program, the portion of the operating system written in MIL. Other examples of MIL programs are the SDL2 interpreter and GISMO.

The operating system components are described in the following paragraphs.

GISMO (GISMO3 AND GISMO2)

Gismo is a MIL program, executed directly, which contains procedures for the following operating system functions:

- Processor scheduling.
- Physical input and output operations.
- System interrupt handling.
- Memory allocation under prioritized memory management.
- Communicate message routing.
- Complex wait (COBOL68, COBOL74, SDL2 WAIT) statement.
- System performance monitoring.
- Tracing.

In a dual processor system, Gismo can execute on either processor. A system of memory locks and conventions allows shared access to memory structures.

MMCP (MCPII/MICRO-MCP AND MICRO-MCP/DEBUG)

The MMCP is a MIL program, executed directly, which contains procedures for processing the following communicate messages:

- Logical input and output operations (record blocking and unblocking).
- COBOL74 interprogram communication.

The MMCP is also called when a reader/sorter operation completes.

In a dual processor system, the MMCP can execute on either processor. A system of memory locks and conventions allows shared access to memory structures.

SMCP (MCPII)

The SMCP is an SDL2 code file, interpreted by the SDL2 interpreter, which contains code for all operating system functions except those listed for the MMCP and GISMO.

The SMCP can only execute on the master processor.

SECTION 4
PROBLEM ANALYSIS OVERVIEW

The purpose of a system memory dump operation is to find the fundamental cause of a system problem. For example a symptom such as a system halt with the L-register = @0D0055@ has, as its immediate cause an attempt to read or write outside the bounds of memory. Further investigation is needed to determine the underlying (fundamental) cause. Further investigation includes a request for a system memory dump and the analysis of the resulting system memory dump file. This may (but does not always) reveal the fundamental cause.

When a B 1000 system is not performing any work, the first step is to determine whether it is hung or halted. If the system is hung, the next step is to determine what is required to halt it.

The following paragraphs provide an overview of the steps required to analyze a malfunctioning system.

IMMEDIATE CAUSE OF A PROBLEM

The immediate cause of each software controlled system halt is listed in the system halt table contained in section 8 of the *B 1000 Systems System Software Operation Guide, Volume 1*. The hardware L-register provides the primary halt definition. For many halts, the T, X, and Y registers contain additional information which assists in locating the fundamental cause of the problem.

The immediate cause of a system hang can be determined by following the instructions that are presented next.

FUNDAMENTAL CAUSE OF SYSTEM HANG

When a system hangs, the fundamental cause may be determined by going through a sequence that takes advantage of what the system does or does not do in response to specific inputs:

1. System responds to ODT.
2. System responds to INTERRUPT.
3. System responds to HALT.
4. System responds to HALT/CLEAR.

In the paragraphs that follow, this sequence is described in the order in which the steps should be taken.

System Responds to ODT

GISMO runs the SMCP, the network controller when using a datacomm ODT, and the SYSTEM/ODT program, but no other work is being performed. A probable cause of the problem is that a high-priority job with no input or output operations is taking all the processor time. When such a job wants the processor, it takes precedence over all jobs of lower priority.

System performance monitoring or the MX ALL system command followed by repeated <mix number> TI system commands can be used to determine which job has control of the processor. Adjusting the priorities of the jobs in the mix may cause other work to resume. If this does not identify the cause, interrupt the system, take a clear/start system memory dump, and locate the job whose QUEUE_ID is NOT_QUEUED.

System Responds to Interrupt

The system does not respond to ODT input.

One possible cause is that the SMCP program is busy and cannot accept requests to perform any other services. A job, including the SYSTEM/ODT program, that needs service from the SMCP program will wait in the SMCP communicate queue (S_COMM_Q) until the SMCP can respond. As time goes by, more and more jobs cease performing work while they wait for the SMCP.

Another possible cause is that the priority of either the SYSTEM/ODT program or the network controller program when using a datacomm ODT has been set lower than some job with no input or output operations and the latter is taking all the processor time. For this reason SYSTEM/ODT and the network controller should be at the highest priority in the mix. Other programs, such as SMCS and SYSTEM/MONITOR, can share that priority.

Another possible cause is that the SYSTEM/ODT program is hung. If SMCS and RD are running, it may be possible to use commands from a remote ODT via SMCS and RD to determine the problem. Zip input goes directly to the MCP and can bypass an ill SYSTEM/ODT.

To determine the cause of the problem, interrupt the system and perform a clear/start system memory dump. Execute the SYSTEM/IDA program, get the SYSTEM/DUMPFILE file, and verify that the SMCP QUEUE_ID is NOT_QUEUED. Then enter the SYSTEM/IDA command LAYOUT SUMMARY to see the history of SMCP procedure calls.

System Responds to HALT

The system does not respond to ODT input or to activation of the console INTERRUPT push button.

One of the interpreters listed in the interpreter dictionary had control of the processor, was stuck in a loop, and could not get out to test for any interrupt. A possible cause is a malfunction in the I/O subsystem. It may be sending service requests at a rate that does not give GISMO time to exit the module that handles service requests to test for other interrupts.

To determine the cause of the problem, halt the system, record the registers specified in the Fault Docket, and perform a clear/start system memory dump operation. The value in the A register is the address of the next microinstruction to be executed. That value, together with the information in the interpreter dictionary, determines which program segment listed in the dictionary had control of the processor. The STATE A @hhhhhh@ system object command, using the interpreter dictionary, will decode the A register. The values in the other registers provide useful information for determining what caused the program segment to loop.

If the value in the A-register indicates that GISMO had control of the system, look through the I/O descriptor chains in the SYSTEM/IDA listing to determine which descriptors had been initiated. When there is a system hang with GISMO in control of the system, the memory dump often fails to show the cause of the problem. If the I/O subsystem is suspect, the next step in trying to isolate the cause is to run a GISMO trace of all I/O channel activity.

System Responds to HALT and CLEAR

When HALT and CLEAR must be activated simultaneously to halt the processor, a processor malfunction has occurred, and a Burroughs Field Engineer must be notified. A system memory dump is unnecessary. Perform a clear/start operation, and try to resume processing.

FUNDAMENTAL CAUSE OF SYSTEM HALT

If the halt occurred in GISMO, the MMCP, or the SDL2 interpreter, the L-register contains a value listed in section 8 of the *B 1000 Systems System Software Operation Guide, Volume 1.* That value, together with any additional information specified as being included in the T, X, and Y-registers, establishes starting points for analyzing the system memory dump. For example, when the L-register indicates a GISMO halt associated with an I/O problem, the starting points are (1) the appropriate I/O descriptor chain, (2) the most recent procedure in the SMCP layout summary, (3) the communicate message of the program performing the I/O operation, and (4) the FIB for the file being referenced.

An L = @0D0055@ (D-55) system halt occurs when GISMO discovers an attempt to read out of the bounds of addressable memory. This event usually is not detected until some time after it occurs. At the time of the halt, if the Y-register contains zero, the error occurred in GISMO. If it contains a value other than zero, the X-register contains the contents of the limit register of the program that was running when the error occurred.

WHEN IS ADDITIONAL INFORMATION NEEDED?

A system memory dump does not always show the fundamental cause of a problem. Examples of such cases include problems resulting from to the corruption of a disk address and system hangs for which the memory dump fails to show a cause.

Problems associated with invalid disk addresses are usually caused by an address that became invalid long before its reference caused the immediate problem. Frequent execution of the SYSTEM/PANDA program may help isolate the time frame within which the corruption occurred. Printouts of formatted analyses of the SYSTEM/LOG and the SYSTEM/ELOG may show a pertinent activity sequence and any I/O errors that occurred during that time.

Locating the cause of a system hang frequently is not possible without a GISMO trace. If an I/O channel is suspected of requesting the undivided attention of GISMO, a GISMO trace of that channel will show it. If there is no specific suspect, a GISMO trace with all trace flags set is the first step in trying to locate the cause of the problem.

WHAT TO DO WHEN THE PROBLEM IS LOCATED

If hardware is the fundamental cause of a problem, then a Burroughs Field Engineer should be called. If the problem is software, the *B 1000 Product Support Information Manual (PSIM)* may contain a corrective measure.

On the first occurrence of a problem, write a description on a Field Communication Form (FCF) and send it, with appropriate supporting documentation, to a Burroughs Software Support representative.

SECTION 5
STATE OF THE SOFTWARE

The state of the software at the time a system memory dump is taken provides information on the job and the operating system component under which it was running. All software-controlled halts occur when an operating system component detects invalid data in memory or in a register. The pertinent operating system components are GISMO, the MMCP, the SMCP, and the components that are active during a clear/start operation (CLEAR/START and SYSTEM/INIT). Knowing which version of an operating system component was in use is a key for determining whether the problem has been corrected with a later release of the system software.

When the SDL2 interpreter halts the system on behalf of the SMCP, the first two digits of the L-register are 00. If the SMCP was performing a service for one of the jobs in the mix, the Job queue for the job will be NOT_QUEUED. The SYSTEM/IDA program object command ESN COMMUNICATE applies to the current job and displays the communicate message sent to the SMCP. The control command LAYOUT displays the procedure call history and the contents of the SMCP name stack.

When the MMCP halts the system, the first two digits of the L register are 02. The MMCP firmware performs services for a job that sends a communicate message. The job queue for the job is NOT_QUEUED, and the communicate message is displayed when ESN COMMUNICATE is entered.

When GISMO halts the system, the first two digits of the L register are OD. For many of the GISMO halts, GISMO has been doing some service for the SMCP or the MMCP, and the X register contains the contents of the Limit register of the applicable component. The cause of the problem can be traced from GISMO back to component and, if necessary, back to the job that sent the communicate message.

VERSION AND STATE OF THE SMCP

The SMCP version number and compilation date are displayed in the outputs of the SYSTEM/IDA control commands GET SYSTEM/DUMPFILE and JOB 0, as shown in the following example:

```
MCP VERSION = MARK 12.0.00 (10/26/84)
```

The SMCP state is found in the same display in the field labeled Job queue, which may have any of the following values:

NOT_QUEUED
: The SMCP is running, or GISMO is running on the master processor.

READY_Q
: The SMCP is ready to resume execution.

WAIT_Q
: The SMCP is waiting for one or more events in the event list. When the event is TRUE, the SMCP is moved to the queue specified by the Next queue field.

SMCP EVENT LIST

The SMCP event list is displayed in the output of the SYSTEM/IDA program object command RSN EVENTLIST. The event list is a list containing, typically, two or five entries.

When the SMCP QUEUE ID is WAIT_Q, and the event list contains five entries, the list shown next appears in the SYSTEM/IDA display.

```
        *** EVENT LIST ***
EVENT          ADDRESS
    0          @7D8C68@      FALSE    (RS_TIME_EVENT)
    1          @000554@      FALSE    (S_M_Q_EV)
    2          @7C379B@      FALSE    (Q_NOT_EMPTY, TO_THE_MCP)
    3          @000552@      FALSE    (S_C_Q_EV)
    4          @00031F@      FALSE    (CHANGE_BIT)
```

The meanings of the entries in a five-entry event list are as follows:

RS_TIME_EVENT
The SMCP requested GISMO to wake it up after a 5-second to 60-second time interval. The time interval is proportional to the number of jobs in the mix.

The following fields in HINTS and the SMCP run structure nucleus are used.

HINTS.SYCOUNTER	current time
HINTS.SY_CNTR_MSK	time at which wait time expires
HINTS.SY_PRIOR_TIME	last time at which wait time expired
RS_PAUSE	unexpired wait time
RS_TIME_EVENT	caused when time to wake up the job

When the time interval has elapsed, RS_PAUSE goes to zero, RS_TIME_EVENT goes TRUE, and the SMCP performs some housekeeping functions, including the following.

** Handle exterminations
** Perform CHECK_CHANGE_BIT procedure (explained below)
** Perform N_SECOND procedure as follows:
** Roll out jobs
Update date and time
** Initiate any delayed random I/O operations
** Load pseudo readers
** Transfer ELOG
** Garbage collect queues
** Fire up SYSTEM/ODT
** Fire up SYSTEM/BACKUP for autoprint
If no remote files open for the third time, QC the network controller
DS any job that has exceeded its maximum time
** Update LOG mix information on the system disk

** = conditional

Q_NOT_EMPTY, ODT_QUEUE_FILE
An input message was transmitted from the ODT, and the SYSTEM/ODT program put the message in the ODT queue. The SMCP deciphers the message and takes the appropriate action.

S_M_Q_EV (SMCP interrupt Message Queue Event)
GISMO has placed one or more of the following possible entries into the Interrupt queue:

An I/O operation completed, and the SMCP had requested notification by setting the RESULT_STATUS INT_REQ bit in the I/O descriptor, and M_EVENTS INT_M_OR_S bit was FALSE.

An I/O operation completed, and the SMCP had requested notification by setting the M_EVENTS S_INT_REQ bit in the I/O descriptor, and M_EVENTS INT_M_OR_S bit was FALSE.

An I/O operation completed, and the SMCP had requested notification by setting the M_EVENTS INT_S bit in the I/O descriptor.

An I/O operation had completed with any exception except datacomm I/O and tape streamer tape mark only. All other tape streamer exceptions are routed to the SMCP.

An S-memory parity error occurred.

The system is thrashing.

The SMCP deciphers the interrupt, takes the appropriate action, and then conditionally performs the CHECK_CHANGE_BIT procedure.

Q_NOT_EMPTY, TO_THE_MCP
An input message was transmitted from the ODT, and the SYSTEM/ODT program put the message in the ODT queue.

S_C_Q_EV (SMCP Communicate Queue Event)
A communicate message has been routed to the SMCP program.

Either GISMO routed a communicate message from a program to the SMCP, or the MMCP needs help from the SMCP because of a problem described in the RS_M_PROBLEM_TYPE field.

The SMCP performs a Q_OUT_TOP operation on the S-communicate queue to cause GISMO to locate the job that did the communicate.

The SMCP performs the operation indicated by the communicate message.

CHANGE__BIT
A job has either entered the schedule, gone to BOJ, or gone to EOJ.

The SMCP performs the CHECK__CHANGE__BIT procedure, which was also referenced above with RS__TIME__EVENT and S__M__Q__EV bits fields, as follows.

If there is a disk squash (SQ system command) operation to be done and no jobs are running, execute the SYSTEM/SQUASH program.

If an OPEN operation has been performed on a remote file and the network controller is not running, execute the network controller program.

If there are jobs in the schedule, and if any jobs in the mix are waiting because of no memory, reinstate the jobs; otherwise, execute a job in the schedule.

When the SMCP Job queue is WAIT__Q, and the event list contains two index-address pairs, the RS__TIME__EVENT field is in the event list as described above, and the M__EVENTS.IOC field is in the disk descriptor chain as described below.

The event list could have the following appearance in the SYSTEM/IDA display:

```
        *** EVENT LIST ***
EVENT          ADDRESS
    0         @7ED718@      FALSE     (RS_TIME_EVENT)
    1         @7DAC4F@      FALSE     (M_EVENTS.IOC)
```

The SMCP dispatched a disk I/O operation to GISMO , and then performed a GISMO communicate specifying that the SMCP was to be put into the Wait queue until either 30 seconds elapsed or the physical disk I/O operation completed. When the event list contains two entries, the entries have the following meanings.

RS__TIME__EVENT
This event becomes TRUE when 30 seconds have elapsed (RS__PAUSE field goes to zero) and the operation has not completed.

M__EVENTS.IOC
This event becomes TRUE when the disk I/O is physically complete.

VERSION AND STATE OF THE MMCP

The MMCP version number is found by entering the SYSTEM/IDA command INTERPS. In the following example, the version number is @12000101@.

```
----> INTERPRETER 0
  NAME = MCPII/MICRO-MCP
  VERSION = @12000101@
```

Also included here are various local parameters to certain MMCP segments.

For a system halt, if the MMCP was running, the first two digits in the L register are 02, and the next four digits specify what the MMCP was doing when it discovered the bad value and halted.

For an interruptible system hang, if the interrupt button halted the processor, the MMCP did not have control. The MMCP responds to service request and timer interrupts only.

For a non-interruptible system hang, the MMCP had control of the processor if the LR value written in the Fault Docket matches the M__MCP__LR field in HINTS.

SYSTEM/IDA STATE Command

Further information about the state of MMCP, specifically, the contents of the Interrupt queue and the MMCP work area, is displayed by the SYSTEM/IDA command STATE.

VERSION AND STATE OF GISMO

The GISMO version number is found by entering the SYSTEM/IDA command INTERPS and locating interpreter 1. In the following example, the version number is @12000101@.

```
----> INTERPRETER 1
  NAME = GISMO3/DEBUG
  VERSION = @12000101@
```

The GISMO state is displayed in the L-register. If the first two digits in the L register are @0D@, the next four digits specify what GISMO was doing when it discovered the bad value and halted. Frequently, the other registers provide additional information.

GISMO State Flags

Upon entry to GISMO, the contents of PERM register bits 1 and 3 are tested. Upon leaving GISMO, the bits are tested again. PERM(1) indicates a read or write outside the bounds of memory (D-55 halt). PERM(3) indicates an uncorrectable CPU access error to S-Memory (D-54 halt).

If the STATE.FLAGS field in the Y-register contains a non-zero value, the error was detected upon entry to GISMO, indicating the error occurred in the program that just called GISMO. The X-register contains the limit register of that program.

If the STATE.FLAGS field contains zero, the error was detected upon leaving GISMO, indicating the error occurred while GISMO had control of the processor.

SYSTEM/IDA STATE Command

Further information about the state of GISMO, specifically, the contents of the Interrupt queue and the GISMO work area, is displayed by the SYSTEM/IDA command STATE.

STATE OF EACH JOB IN THE MIX

The state of each job in the mix is found by entering the SYSTEM/IDA command JOB SUMMARY to obtain a job summary display.

To find the value of a next instruction pointer of the job, enter:

```
JOB <number>
ESN
```

and read the value in the NIP field.

To find the contents of a communicate message field of the job, enter:

```
JOB <number>
ESN COMMUNICATE or ESN C
```

To find more information about a job that is waiting for one or more events to become TRUE, enter the following:

```
JOB <number> or J <number>
```

and read the values for Next queue and Job status.

To find the addresses and values of the events for which a job is waiting, enter the following:

```
JOB <number>
RSN EVENTLIST or RSN E
```

The kinds of events upon which jobs in the mix typically wait include an interval of time and other conditions specified in a WAIT statement.

JOB QUEUE IDENTIFIERS

Following are interpretations of the various Job queue identifiers:

NOT QUEUED
The job is running on either the master or slave processor, or its communicate message is being processed, or GISMO is running. On a dual processor system, if the RSN address of the job matches HINTS.DCPU_SLAVE_LAST_REIN, then the job is running on the slave processor, or its communicate message is being processed, or the slave GISMO is running.

READY_Q
The job is ready to resume execution on either the master or slave processor.

EXTERMINATE_Q
The job is ready for the SMCP program to discontinue the job on the master processor.

S_COMM_Q
The job is ready for the SMCP to begin or resume processing a communicate message on the master processor.

M_COMM_Q
The job is ready for the MMCP to resume processing a communicate message on the master processor. Processing could have been interrupted if a MMCP code segment was not present, or if a new disk area was needed for a read or write operation.

IOC_Q
The job is ready for the MMCP to resume processing a communicate message on either the master or slave processor. Processing was interrupted to wait for the completion of some event, such as an I/O operation.

WAIT_Q
The job is waiting for one or more events in its event list. For better resolution on what the job is waiting for, see the Next queue, job status, and event list. When the event becomes TRUE, the job is moved to the queue specified by the Next queue field.

The Next queue, job status, and event list are in a defined state only when the Job queue contains WAIT_Q. When the Job queue has any other value, these fields exist but are in an undefined state, and only the value in the Job queue defines the state of the job.

SECTION 6
STATE OF THE INPUT/OUTPUT OPERATIONS

The state of the input/output operations at the time a system memory dump is taken provides additional information about the state of the system. For every input/output operation there is an associated result descriptor (I/O descriptor) defining the status of the operation. Since GISMO handles all physical soft input and output on the master processor, analysis of the I/O descriptors will provide information about the state of the GISMO on the master processor. Hard input and output for MLCs can be initiated by either processor. Dual processor systems also use port to port communications to send messages to each other. This information is useful in analyzing system hangs or halts.

The I/O descriptors may be divided into two groups: those that are associated with devices on multiple unit channels, such as disk and tape, and those that are associated with devices on single unit channels, such as card reader and printer.

The I/O descriptors for devices on multiple unit channels are found by entering a DISK command to display the disk descriptor chain and by entering a TAPE command to display the tape descriptor chain.

The I/O descriptors for devices on single unit channels are found as follows:

1. Enter an IOAT command to display the Input-Output Assignment Table. This table contains an entry for every device physically connected to the processor through the I/O subsystem.

2. Locate the entry for the device of interest.

3. If a file has been opened on the unit assigned to that device, the unit is connected to a job in the mix.

    1) Locate the JOB NUMBER, and note the corresponding FIB ADDRESS.

    2) Enter a JOB <number> command followed by a FILE SUMMARY command.

    3) Locate the file number by finding the FIB ADDRESS in the ADDRESS column, and read the corresponding file number in the SG column.

    4) Enter FILE <number> to see the File Information Block for the file.

    5) The I/O Descriptors are found within the File Information Block.

4. If a file has not been opened on the unit assigned to that device, the unit is not connected to a job in the mix.

    1) The I/O descriptors are found in the Unit Test Descriptor chain at the end of the IOAT.

    2) Note that these are the only I/O descriptors in the Unit Test Descriptor chain that are in a defined state at the time of the system memory dump.

For further information, refer in this manual to appendix F, Input/Output Operations, and to I/O Subsystems in section 2, System Elements. Also see section 6, I/O Subsystems and Device Controls, in the *B 1870/B 1860 Systems Reference Manual.*

APPENDIX A
SYSTEM/IDA EXAMPLES

A sample analysis of a system memory dump using the SYSTEM/IDA program is presented in the following pages. A system memory dump operation was performed by entering a DM system command with the DBUG system option set. The execution of the SYSTEM/IDA program, the entry of various SYSTEM/IDA commands that display the state of the system at the time of the dump, and the corresponding output are shown and described.

EXECUTING THE SYSTEM/IDA PROGRAM

The following analysis of a system memory dump was performed at a remote terminal that was signed on to the SMCS program when the EXECUTE system command was entered.

The SYSTEM/IDA program opened a remote file. While that file was open, messages that did not begin with the signal character were routed to the SYSTEM/IDA program. Messages beginning with the asterisk (*) signal character were routed to the SMCS program.

After the remote file was opened, the program displayed the message "--Ready for input (type HELP for help) --" and waited for a SYSTEM/IDA command. The command entered, OPTION LIST, caused the list to appear in a printer backup file.

```
EX SYSTEM/IDA
    SYSTEM/IDA =1900 BOJ. PP=4, MP=4 TIME =13:19:31.7
    REMOTE FILE OPENED BY "SYSTEM/IDA",  SIGNAL = *
    -- Ready for input (type HELP for help) --"
OPTION LIST
    Options enabled:  LIST  ARRAYLIMIT = 64
    --
```

GETTING THE DUMPFILE

The SYSTEM/IDA command GET SYSTEM/DUMPFILE causes the SYSTEM/IDA program to open the SYSTEM/DUMPFILE file for analysis and to display general system version information. This dump file resides on the DL DUMP system command's designation pack. Therefore, the ON keyword followed by a pack-id must be used when the SYSTEM/DUMPFILE file is not on the system pack. Refer to the *B 1000 Systems System Software Operation Guide, Volume 1,* for more information about the DL command.

The information displayed includes the date and time the system memory dump operation was performed, the host name of the system, the version and compile date of the SMCP, and the contents of the L, T, X, and Y registers. The contents of the registers are decoded.

MCP STATUS

In this example, the SMCP was running (NOT_QUEUED) because it was processing the DM system command. If a CLEAR/START system memory dump operation had been performed, the contents of the L, T, X, and Y registers would have appeared on this screen.

Example:

```
GET SYSTEM/DUMPFILE
         System dumpfile title = SYSTEM/DUMPFILE
          11/05/84 12:37:38.6  :Date of dump
                   "PAASSBPC"  :HOSTNAME
           MCP VERSION = MARK 12.0.00 (10/30/84)
            PSEUDO MEMORY SIZE = 2097152 BYTES
                         MCPII  :MCP NAME
              MCPII/MICRO-MCP   :MICRO MCP NAME
                   SDL2/INTERP  :INTERPRETER NAME
                        GISM03  :GISMO NAME
                   SYSTEM/INIT  :INITIALIZER NAME

                    *** MCP = MCPII ***
                   Job queue: NOT_QUEUED
                   Job status: Executing
    MCP: Next S-op at P= 1,S= 0,D=   2896 = @00B50@
   Options enabled:  LIST  ADDRESSES  ARRAYLIMIT = 64
   -- End GET.
```

HELP COMMAND

The HELP command displays the list of available commands or the syntax of a specific command.

Example:

```
HELP
 Help:
  'HELP' followed by a command will show the syntax for that command.
  Commands are:
    ?BRK BYE DATA ?DP ENVIRONMENT ESN
    FILE FIB FPB GET IPC HELP LAYOUT MEMORY OPTION PPB PRINT
    RSN SMACHINE SPAD
    The following commands are not available in program dumps:
    BNA CHANNELS CODE CSV DCH DFH DISK DMS ERRORTABLE HINTS INTERPS I
    JOB LINKS MCP MPB NAMETABLE ODT PACKS PM PSR QUEUES STATE TAPE TR
  Lengthy output is written to a buffer that may be SCROLLed.
  Scroll commands are line numbers and:  + - . $
  The ?BRK command will terminate an endless loop.
  HELP SW or HELP SWITCHES will explain IDA's program switches.
  The manual gives the complete command semantics.
  Current level = 12.0, patch 0, compiled on 11/09/84 04:30 PM
  --END HELP.
```

ODT QUEUE

The ODT command displays the most recent entries in the ODT queue. These are the entries at the end of the queue. All other SYSTEM/IDA commands display entries from the beginning of their corresponding data structures, with the exception of the TRACE command.

In the following example, the date field which normally appears on the left side of the output has been deleted to facilitate getting the example on the page. The last entry in the list shows the DM system command that caused this system memory dump.

Example:

```
ODT

  16:26:46.0 0       12/15/83.
  16:26:48.0 0       % (ODT) SYSTEM/ARCH =1795 PLEASE STOP IF INCORRECT
  16:26:54.5 0       % (ODT) SYSTEM/ARCH =1795 ENTER PACKS TO
  16:26:54.5 0        ARCHIVE WHEN ACCEPT APPEARS
  16:26:54.8 0       CANDE =1044 S/(DAN)/QRSTREAM REPLACED
  16:27:01.0 0       (ODT) SYSTEM/ARCH =1795 ACCEPT.
  16:27:08.9 I ODT 1795AX  DB
  16:27:12.8 I ZIP REDB/ARCHIVE/FINDPACK
  16:27:13.7 0       DB/ARCHIVE/FINDPACK REMOVED
  16:27:18.5 0 RMT   (SYSTEM/IDA) SYSTEM/IDA  =1784 S/(SYSTEM/IDA)/DWH1
                      RELEASE
  16:27:21.9 0 RMT   (SYSTEM/IDA) SYSTEM/IDA =1784 EOJ. TIME= 16:27:21.7
  16:27:23.3 0       % (ODT) SYSTEM/ARCH =1795 ENTER SPECS:SPAN
  16:27:23.3 0        INCL SELECT TYPE EXCL
  16:27:23.5 0       % (ODT) SYSTEM/ARCH =1795  FOR DB PACK
  16:27:23.7 0       (ODT) SYSTEM/ARCH =1795 ACCEPT.
  16:27:31.1 0 RMT   COBOL74 : *NOTITLEO =1793 EOJ. TIME = 16:27:31.0
  16:27:31.5 I ODT 1795AX
  16:28:09.5 I ODT 1795AX  S
  16:28:18.7 I ODT 1795AX  SELECT CARDS
  16:28:24.1 I ODT 1795AX
  16:28:29.2 I ODT 1795AX
  16:28:44.7 I ZIP QU SMCS/MCPQ LS SZ 9 RR 1 US SYSTEM/IDA DM
  -- End ODT QUEUE; 861 lines --
```

MCP ANALYSIS

The next two examples show the SMCP activity between the time the DM system command was entered and the time the contents of memory were written to the SYSTEM/DUMPFILE file. The first example shows the flow of control, and the second example shows an important data structure for dumping system memory.

MCP Layout Summary

The MCP layout summary shows the history of MCP procedure calls. In the example that follows, the global procedure called procedure GET_SET_GO, which is where the MCP waits for something to do. GISMO activated the MCP to check a communicate message because a system command was zipped. The control card driver was called to decode the command and discovered the command began with the letter D. The command turned out to be a command to dump the system state. A procedure was then called to build the disk I/O descriptor (MDD) of the MCP and wait either for 30 seconds to elapse or for the operation to complete. The last procedure dispatched the MCP disk descriptor to GISMO.

Example:

```
LAYOUT SUMMARY
Frame Kind     Name                                Next op at:
    0  Global                                        P= 0,S= 0,D=    212
    1  Proc     GET_SET__GO                          P= 1,S= 0,D= 116232
    2  Proc     CHECK_COMMUNICATE_MESSAGE            P= 1,S= 0,D= 103344
    3  Proc     ZIPP                                 P= 4,S=11,D=   4512
    4  Proc     CTRL_CARD_DRIVER                     P=31,S= 8,D=  18012
    5  Proc     D                                    P=31,S= 2,D=  19024
    6  Proc     DUMP_SYSTEM_STATE                    P=11,S= 3,D=   7652
    7  Proc     BIOAW                                P= 1,S= 0,D=  24896
    8  Proc     INITIATE_IO                          P= 1,S= 0,D=   2896
  -- End LAYOUT SUMMARY --
```

MCP Layout Frame and Variable

The LAYOUT 0 VAR MDD command displays the disk descriptor (MDD) of the MCP. This descriptor is in frame 0, the global procedure.

In the example that follows, descriptor bit 1 is OFF and descriptor bit 2 is ON, indicating the descriptor was initiated by GISMO to the disk control and that the operation is in process. When the COMPLETE bit is OFF, the meaning of the following bit changes from EXCEPTION to INITIATED. The OP_CODE and UNIT fields specify a write operation to unit 0. The BEGIN and END_ADDR fields specify that the write operation is to take data from memory address 0 through memory address @FFFFFF@. The DISK_ADDRESS field specifies that the data is to be written to disk sector @047383@. The PORT and CHANNEL fields specify the descriptor is for port 7 channel 9.

Example:

```
LAYOUT 0 VAR MDD
    Frame 0; Next S-op at P= 0,S= 0,D=    212 = @000D4@
    -- Layout of variable MDD
    Global frame
    Locals:
      MDD.ACTUAL_END [FD81E4] = @FD7FA0@ (16613280)
          RESULT [FD81FC] = @520E09@ (5377545)
          BIT_1_2 [FD81FC] = 1
          COMPLETE [FD81FC] = FALSE
          EXCEPTION [FD81FD] = TRUE
          INT_BITS [FD820B] = 0
          INTERRUPT [FD820B] = FALSE
          HI_INT [FD820C] = FALSE
          LINK [FD8214] = @FDBFFF@ (16629759)
          OP [FD822C] = @400000@ (4194304)
          OP_CODE [FD822C] = 2
          UNIT [FD8240] = 0
          BEGIN [FD8244] = 0
          END_ADDR [FD825C] = @FFFFFF@ (16777215)
          DISK_ADDRESS [FD8274] = @047383@ (291715)
          M_EVENTS [FD828C] = 8
          M_EVENTS_IOC [FD828C] = FALSE
          M_EVENTS_SIOC [FD828D] = FALSE
          M_EVENTS_INT_M [FD828F] = FALSE
          M_EVENTS_S_INT_SENT [FD8290] = TRUE
          M_EVENTS_M_INT_SENT [FD8291] = FALSE
          M_EVENTS_INT_S [FD8293] = FALSE
          MCP_IO [FD8294] = @000A@ (10)
          FIB_ADDR [FD82A4] = 0
          FIB_LINK [FD82BC] = @FD80E5@ (16613605)
          BACK_LINK [FD82D4] = @FDA11F@ (16621855)
          PORT_CHAN [FD82EC] = @79@ (121)
          PORT [FD82EC] = 7
          CHANNEL [FD82EF] = 9
          BEEN_THRU_ERROR [FD82F3] = FALSE
    -- End LAYOUT Frame 0; 35 lines --
```

DISK DESCRIPTOR CHAIN

The DISK command displays the disk descriptor chain and shows the disk descriptor of the MCP, which is always in the second entry immediately following the system pause descriptor in the first entry. These descriptors are also found in a layout of MCP frame zero, with the variable names MDD and SPD.

The disk descriptor of the MCP shows the same information that was displayed by the LAYOUT 0 VAR MDD command. It describes a write operation to port 7, channel 9, unit 0, disk sector @047383@, beginning at memory address 0 and ending at memory address @FFFFFF@.

Example:

```
DISK

                        *** DISK DESCRIPTOR CHAIN ***
RESULT                                                          IO
DESCR  ACTUAL RESULT                                            MCP
ADDR   END    DESCR  LINK   IO OP  BEGIN  END    DISK   IO FIB
-------------------------------------------------------------------
FDA11F        000000 FD81FC 840000
FD81FC FD7FA0 520E09 FDBFFF 400000 000000 FFFFFF 047383 10
 .. M_EVENTS = @08@       ... PORT = 7, CHANNEL = 9
FDBFFF FC80EA 800080 FD8C4C 400000 FC7FFE FC80EA 04114E 18 FD2581
 .. M_EVENTS = @C0@       ... PORT = 7, CHANNEL = 9
FD8C4C        120F09 FD2E10 900005                        11
 .. M_EVENTS = @08@       ... PORT = 7, CHANNEL = 9
FD2E10 26C4FB 800080 FBEA57 000000 26B263 26C4FB 0378D3  5 FD2581
 .. M_EVENTS = @C0@       ... PORT = 7, CHANNEL = 9
FBEA57 6BB79D 800080 F32DBF 000000 6BB64D 6BB79D 0000F3 24 FBE88E
 .. M_EVENTS = @C0@       ... PORT = 7, CHANNEL = 9
F32DBF F33457 800080 F93290 400000 F32EB7 F33457 03DD50    F3298F
 .. M_EVENTS = @C0@       ... PORT = 7, CHANNEL = 9
F93290 7D2A64 800080 F8E48F 000000 7D0ECC 7D2A64 01AF43  5 F92A01
 .. M_EVENTS = @C0@       ... PORT = 7, CHANNEL = 9
F8E48F F8EB27 800080 FB9622 400000 F8E587 F8EB27 001B23    F8E05F
 .. M_EVENTS = @C0@       ... PORT = 7, CHANNEL = 9
FB9622 6BB7DD 800080 FDA11F 000000 6BB64D 6BB7DD 0255D8 24 FB9459
```

COLD START VARIABLES

The CSV command displays the first page of the cold start variables. The following example illustrates the cold start variables for a specific system.

Example:

```
CSV

                *** COLD START VARIABLES ***
                             0  :CS_INTERP
                             0  :CS_MCP
                             0  :CS_GISMO
                             0  :CS_INIT
                             0  :CS_MICRO_MCP
                             0  :CS_CONTROLLER
                             0  :CS_MCS
                             0  :CS_ODT
                   @F2000008A@  :NAME_TABLE
                             7  :INTERP_DIC_ENTRIES
                          2258  :CS_SIZE
                   @F20047383@  :DUMP_FILE
          @F00000@:   15728640  :CSV_COLD_START_LEVEL
                          TRUE  :L61_NAME_TABLE
                          TRUE  :L10_0_NEW_SYS_DISK_TABLES
                          TRUE  :L10_0_NAME_TABLE
                          TRUE  :L11_0_NAME_TABLE
                             0  :GISMO_TRACE_FLAGS
                         18650  :DUMP_FILE_SIZE
                             0  :CORRECTABLE_ERROR_TABLE_LEN
                   @000000000@  :MPF_TABLE
                   @F2000FABA@  :LOG_MIX_INFO
```

TERMINATING THE SYSTEM/IDA PROGRAM

Execution of the SYSTEM/IDA program is terminated by entering the BYE command.

```
BYE
    -END OF SESSION-
```

APPENDIX B
HARDWARE ORGANIZATION

An understanding of the paths available for data flow to and from I/O devices, the processors, and memory can greatly facilitate the analysis of system memory dumps.

Figure B-1 shows a B 1955 single processor system. The processor is connected to memory through port 0, the host adapter, and the memory base unit, and is connected to the I/O devices through port 7, the I/O subsystem, and I/O controls. Single-unit channels require I/O controls only; multiple-unit channels include electronics controllers. A single-line control may be attached on channel 13 of the I/O subsystem.

Figure B-2 shows a B 1985 dual (master and slave) processor system. As in figure B-1, the master processor is connected to memory through port 0 and to the I/O devices through port 7, the I/O subsystem, and I/O controls for single unit channels and electronics controllers for multiple unit channels. Note that both NRZ and PE magnetic tape units can be attached to the I/O subsystem on channels 11 and 12.

The slave processor connects to memory through port 1, a host adapter, and the memory base unit. Port 2, from the host adapter, connects the slave processor to a multiline control that services up to 15 datacomm adapters and the associated terminals.

Figure B-3 shows a B 1990 single processor system. The ODT is connected directly to the processor, and there is no host adapter. The disk subsystem controller on channel 9 includes a printer control that services one line printer through channel 8 and up to 8 disk pack units. The magnetic tape control needs no electronics controller; it accepts one tape unit. The multiline control is required and may be connected on either port 1 or port 3.

Figure B-4 shows a B 1990 dual processor system. The slave processor is connected through port 2 to the memory base unit and memory. The disk subsystem controller on channel 9 includes a printer control that services one line printer through channel 8 and up to 8 disk pack units. The magnetic tape control has a master electronics controller with the provision for up to 8 tape units.

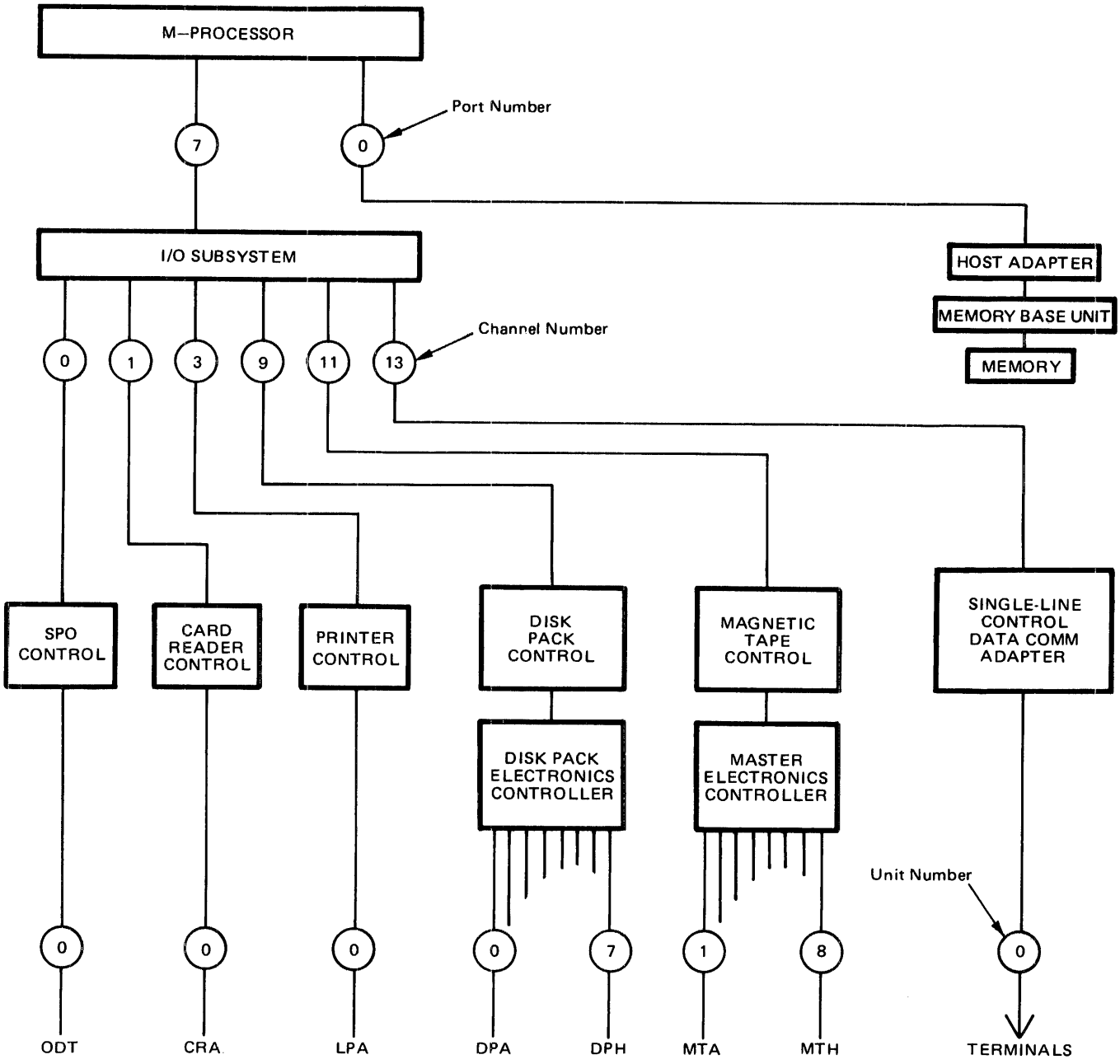


Figure B-1. B 1955 Single Processor System

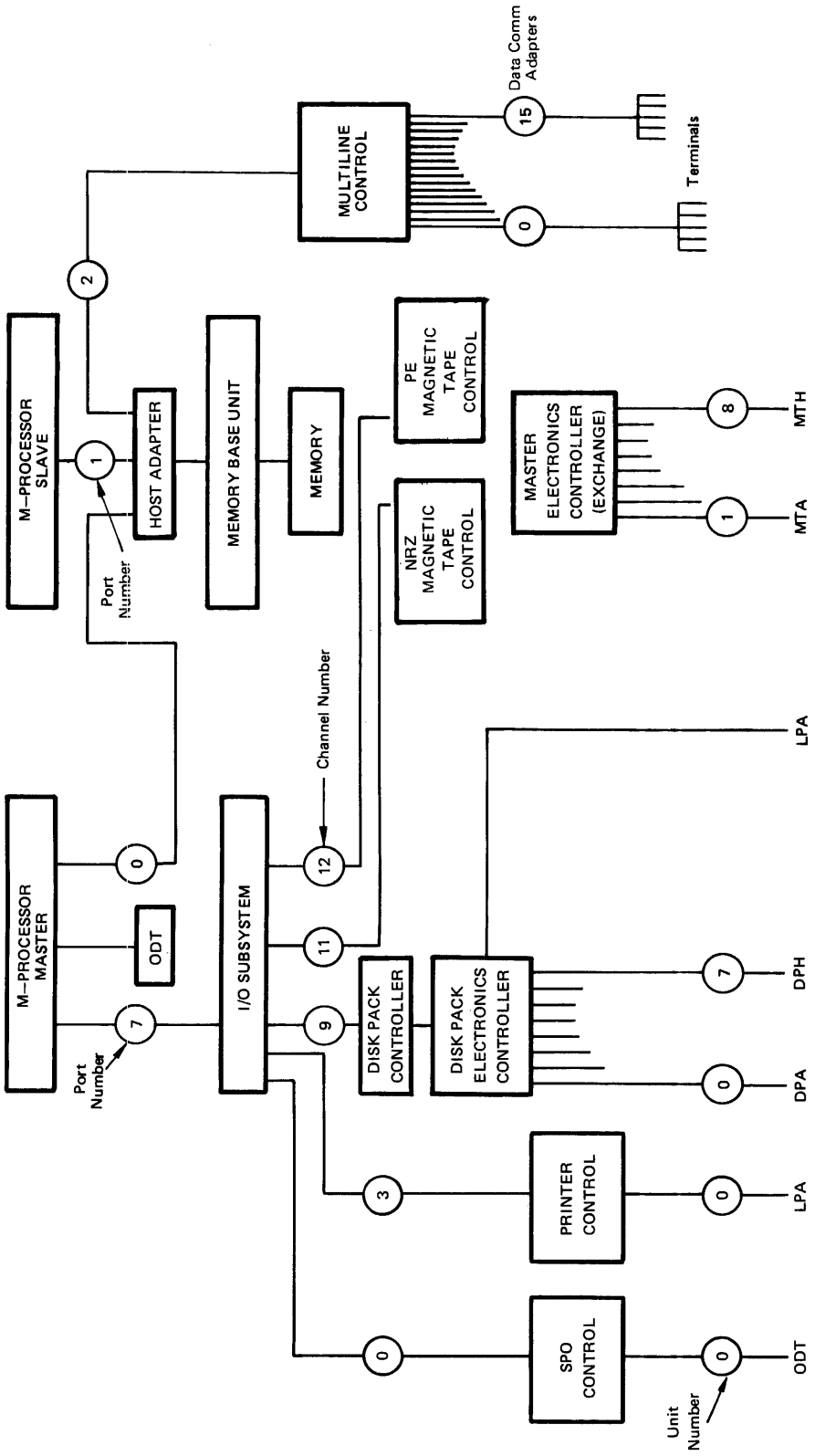


Figure B-2. B 1985 Dual Processor System

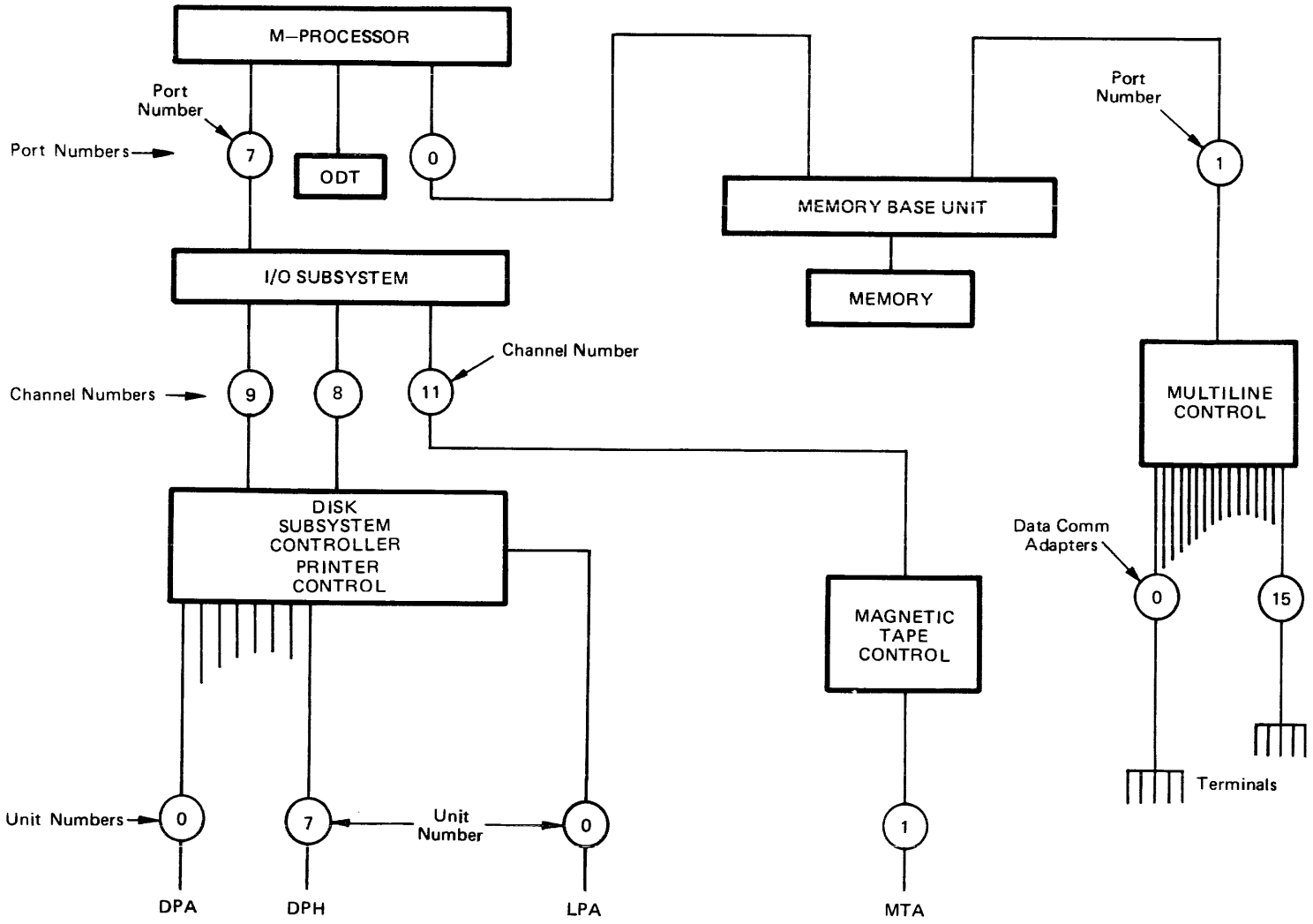


Figure B-3. B 1990 Single Processor System

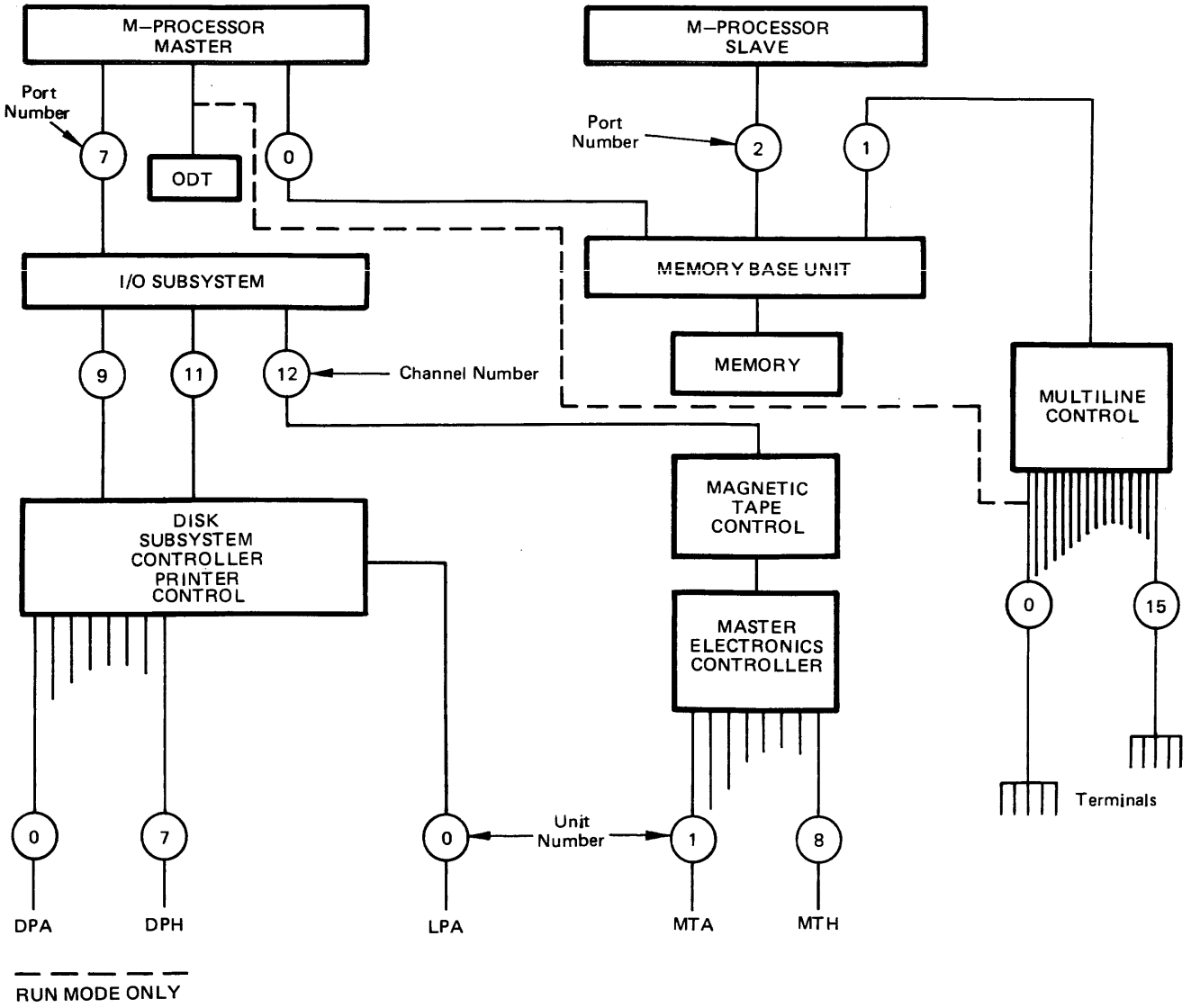


Figure B-4. B 1990 Dual Processor System

APPENDIX C
PROCESSOR ALLOCATION

GISMO controls all processor allocation. It has only one entry point. The address of the entry point is in the HINTS.MASTER_GISMO field. The instruction at the entry point transfers control to the ENTER.GISMO module.

The ENTER.GISMO module first tests for an out-of-bounds memory access or a memory parity error. If neither condition exists, control is transferred, based on a value passed in the X-register, to one of the procedures listed next and described in the paragraphs that follow.

```
HANDLE.COMMUNICATE
HALT.AND.EXPLAIN
DISPATCH.THRU.CHANNEL.TABLE.AND.CLEAR.EXCEPTION.IDLE
DISPATCH.THRU.CHANNEL.TABLE
MCP.FETCH.INTERRUPT
HANDLE.INTERRUPT
MMCP.RETURNING.CPU
MCP.SAVE.IN.IQ
ENABLE.DISABLE.INTERRUPTS
START.SCHEDULER
INTERP.OR.MCP.TRACE
COMMUNICATE.WITH.GISMO
```

HANDLE.COMMUNICATE

A communicate message has been sent from a job. The format of the communicate message is contained in the programmatic description of the environment structure nucleus.

The following is a programmatic description of the communicate routing process. The process is based on the contents of the type bits in the communicate message.

```
Stop accumulating processor time for the job.
If the type indicates a program internal interrupt then
    If the interrupt number is 60 then
        A job is giving up control.
        Mark the job in the Ready queue.
        Run the scheduler.
    If the interrupt number is greater than 55 then
        Mark the job in the SMCP communicate queue.
        Run the scheduler.
    If the interrupt number is 18, 53, or 55 then
        One of the following errors is being reported:
            A write beyond the job's base limit area.
            A cassette data error.
            A read or write out of physical memory.
        If the reporting job is not the SMCP then
            Mark the job in the SMCP communicate queue.
            Run the scheduler.
        Else (the reporting job is the SMCP)
            Move one of the following values to the L-register:
                @0D0012@
                @0D0053@
                @0D0055@
            Halt the processor.
```

When the type does not indicate a program internal interrupt, the run structure for the job is examined before the communicate verb is used to index into the communicate splitter mask.

```
If the job is currently active in a use routine then
    Run the MMCP to interpret a pocket select communicate.
    If the type does not indicate a standard communicate then
        Mark the job in the SMCP communicate queue and then run
        the scheduler.
```

The type indicates a standard communicate. If the verb is @800@ or greater, the communicate is a GISMO communicate. Following is a description of the GISMO communicate in programmatic terms.

```
If the job does not have the ESN control state bit set then
    Discontinue the execution of the job.
If the verb is @801@ then
    Place the job in the Wait queue.
If the verb is @803@ then
    Perform a Adjust MCP Interpreter operation.
    (Obsolete. Used on B1720 systems to transfer control to the
    SMCP interpreter after adjusting control memory management
    fields.)
If the verb is @804@ then
    Perform a Find Window operation.
    (Used to select an area of memory to satisfy a request for
    S-memory)
Else
    Move @0D0039@ to the L-register.
```

If the verb is 47, perform a complex wait operation. Build a list of absolute memory addresses, each pointing to a Boolean. If any of the Booleans is TRUE or becomes TRUE, the job is put into the queue specified by the next Q. If all the Booleans are FALSE, put the job into the WAIT.Q.

If the verb is 40, do a DC INITIATE.IO.

If the verb is 13, perform one of the following operations:

Data Overlay for SDL or SDL2
DMS Read
DMS Lookahead Read
DMS Write
DMS Audit Write
Direct I/O

Use the verb to index into the communicate splitter mask. The mask is a series of 8-bit fields specifying where the communicate operation is to be routed. Based on the value returned, one of the following actions takes place.

Value = 4
Mark the job in the SMCP communicate queue.
Set the SMCP Communicate Queue Event bit.
If the SMCP was in the Wait Queue then
Mark the SMCP in the Ready Queue.
Run the scheduler.

Value = 5
If there is a service request then
Handle the service request.
Run the MMCP.

Value = 2
Switch between the primary and DMS environments.

Value = 3
Switch between the primary and IBASIC environments.

HALT.AND.EXPLAIN

An invalid entry parameter was passed in the X-register. The system halts with L = @0D0042@, and the T-register contains the limit register of the program that passed the invalid parameter.

DISPATCH.THRU.CHANNEL. TABLE.AND.CLEAR.EXCEPTION.IDLE

Clear the EXCEPTION.IDLE bit in the channel table then perform DISPATCH.THRU.CHANNEL-.TABLE.

DISPATCH.THRU.CHANNEL.TABLE

An I/O descriptor has been dispatched. If the I/O channel is not busy, and not idle because of an exception, the descriptor is sent to the I/O control.

NOTE

Only channels that cannot continue after an exception, such as the card reader or the printer, will have the exception idle bit set. If the channel is busy, the descriptor is sent later.

MCP.FETCH.INTERRUPT

The address of an I/O result descriptor is returned to the MCP.

HANDLE.INTERRUPT

Interpreters typically consist of a loop that (1) fetches, (2) decodes, and (3) executes an instruction. A test is made at the beginning of the loop for any interrupt condition. The processor hardware assists here by ORing all interrupt conditions into a single bit, called ANY.INTERRUPT, which can be tested in a single microinstruction. When such a condition exists, control is transferred to the HANDLE.I-NTERRUPT procedure. The following is a programmatic description of the procedure.

If a memory read data error or a write beyond the limit register error occurred then the following action is taken:

```
Read and clear the error log register.
If an uncorrectable memory error occurred then
    Move @0D0054@ to the L-register.
    Halt the processor.
If a single bit error occurred then
    Make an entry in the Correctable Error Table.
    Make an entry in the Interrupt queue.
    Set the SMCP interrupt queue event.
    If the SMCP was in the Wait queue then
       Mark the SMCP in the Ready queue if the SMCP was waiting
       on that event.
If a cassette data error occurred then
    Return with 53 in the L-register.
If a read or write beyond the MAXS register occurred then
    Return with 55 in the L-register.
If a write beyond the limit register occurred then
    Return with 18 in the L-register.
```

The following describes in programmatic terms the beginning of the HANDLE.INT.LOOP procedure.

```
If the console halt bit is set then
    Wait for I/Os to complete
    Move @0D0010@ to the L-register.
    Halt the processor.
Reset the service request bit.
If the port interrupt bit is set then
    Handle one of the following port interrupt conditions:
        A message from the master processor to the slave
        processor to do one of the following:
            Purge the slave processor cache memory.
            Block the scheduler from running a specific job
            or from running any jobs on the slave processor.
            Unblock the scheduler from running a specific job
            or from running any jobs on the slave processor.
            Set the slave processor console interrupt flag.
        A message from the slave processor to the master
        processor to perform a DISPATCH.THRU.CHANNEL.TABLE
        operation.
        A message from the multiline control that an I/O
        operation completed.
    Transfer control to the beginning of the HANDLE.INT.LOOP
    procedure.
If the service request bit is set then
    An I/O control is requesting service by the master processor
    for one of the following reasons:
        To transfer data.
        Because an I/O operation completed.
        Because a seek operation completed.
    Transfer control to the beginning of the HANDLE.INT.LOOP
    procedure.
If the timer interrupt bit is set (occurs every tenth of a second)
then  do the following:
    Test for dual processor timeout.
    Leave message for scheduler that timer interrupt occurred.
    Move the time register to HINTS.TIME_MARK.
    Add 1 to HINTS.SYCOUNTER.
    If the memory management sampling interval has elapsed,
    perform the thrashing detection logic to conditionally
    set the HINTS.MEM_SWEEP_PENDING bit.
    Transfer control to the beginning of the HANDLE.INT.LOOP
    procedure.
End of the HANDLE.INT.LOOP procedure.

If no Force Reschedule bit set or interrupts are disabled then
    Allow the interrupted job to resume processing by returning
    with 0 in the L-register
Else
    Require the interrupted job to save state and give up control
    to the scheduler by returning with 60 in the L-register.
```

MMCP.RETURNING.CPU

The MMCP is returning control of the processor for one of the following reasons:

1. The MMCP completed tanking (saving reader/sorter data in a buffer), and has set up a job for running the use.routine. In this case, reinstate the job to run the use routine.

2. The MMCP completed a communicate operation for a job. If no Force Reschedule has occurred then reinstate the job, otherwise, mark the job in the Ready queue and run the scheduler.

3. The MMCP needs help from the SMCP before it can complete a communicate operation for a job; for example when a logical write operation is in process and a new disk area is required. In such cases, mark the job in the queue specified by the MMCP. Run the scheduler.

4. The MMCP has built a wait list of events. In this case, if any of the events were TRUE, then put the job in the queue specified by the next queue ID; otherwise, put the job in the WAIT_Q. Run the scheduler.

MCP.SAVE.IN.IQ

This entry point is not used.

ENABLE.DISABLE.INTERRUPTS

The DISABLE_INTERRUPTS field in the run structure of the currently running job is decremented or incremented. Interrupts are enabled if the new value is zero; otherwise they are disabled. If interrupts become enabled, the service request bit is set.

START.SCHEDULER

This is the initial entry point from SYSTEM/INIT at CLEAR/START. If DMS is not permitted to run on the slave processor, a bit in HINTS is set to remember this. This procedure then transfers control to the scheduler procedure. Since a CLEAR/START has just been performed, the scheduler will find the SMCP ready to run.

SCHEDULER

The scheduler is the procedure that allocates the processor to the SMCP, the MMCP, and the jobs in the mix. The following is a programmatic description of the scheduler.

```
If any interrupts are outstanding then
    Call the HANDLE.INT.LOOP procedure to handle all outstanding
    interrupts.

If running on the master processor then
    If a timer interrupt has occurred then
        Run the memory link sweeper if the MPRI system option
        is set and the HINTS.MEM_SWEEP_PENDING bit is set.
        Run the timer dispatch procedure (every 16th timer
        interrupt) to initiate test and wait operations
        on idle disk and tape controls to see if any
        disks or tapes have been mounted or dismounted.
        Move the SMCP and any job in the mix whose wait time has
        expired to the queue specified in its RS_NEXT_Q field.
```

```
    If the MMCP high priority interrupt queue event is TRUE (used for
    reader/sorter operations) then
        Run the MMCP.
    If the SMCP is in the Ready queue and the SMCP interrupt queue
    event is TRUE then
        Run the SMCP.
    If the SMCP is in the Ready queue then
        If the scheduler is blocked then
            Run the SMCP.
        If it is time for the SMCP to do its housekeeping then
            Run the SMCP.
        Set a flag to note that the SMCP was in the Ready queue.
    Else
        If the scheduler is blocked then
            Idle and wait for a any interrupt.

    Select the highest priority job in the mix which is in the
    Ready queue, MMCP communicate queue,  SMCP communicate queue
    (if the SMCP was in the Ready queue), or the I/O complete queue.
    If more than one job of the same priority group is available,
    select the job which follows the last job run at that priority.
    (The last job run within a priority group has the RS_LAST_TOP
    bit set.)

    If no job is selected then
        If the SMCP was not in the Ready queue then
            Idle and wait for any interrupt.
        Run the SMCP.
    If the job selected was in the SMCP communicate queue then
        Run the SMCP.
    If the SMCP was in the Ready queue and its CPU priority is
    greater than or equal to that of the selected job then
        Run the SMCP.

    If no job is selected then
    Mark the job as Not queued.
    If the job had been in the MMCP communicate queue or the I/O
    complete queue then
        Run the MMCP.
    Reset the previous RS_LAST_TOP bit; set the current
    RS_LAST_TOP bit.
    Reinstate the selected job.

Else (running on the slave processor)
    If the scheduler is not blocked from running on the
    slave processor then
        Select the highest priority job in the mix which is in the
        Ready queue or the IOC queue and is not blocked.
        If more than one job of the same priority group is available,
        select the job which follows the last job run at that priority.
        If a job is selected then
            Mark the job as Not queued.
            Save the address of its run structure in
            DCPU.SLAVE.LAST.REIN.
            If the job was in the IOC queue then
                Run the MMCP.
        Reset the previous RS_LAST_TOP bit and
        set the current RS_LAST_TOP bit.
        Reinstate the selected job.
    Else
        Go idle and wait for a timer interrupt.
```

INTERP.OR.MCP.TRACE

This procedure is called to make an entry into the GISMO trace table.

COMMUNICATE.WITH.GISMO

The various GISMO functions that are performed at this entry point are listed in the following paragraphs.

Schedule Operations

Included in the GISMO functions are the following operations that affect the scheduler.

INTERRUPT.SLAVE

Dispatches a port interrupt to the slave processor to set the slave processor console interrupt flag.

Q.OUT.TOP

Locates the highest priority program in a specified queue. For example, when the SMCP communicate event is TRUE, the SMCP performs a COMMUNICATE.WITH.GISMO operation to locate the highest priority program in the SMCP communicate queue.

MARK.IN.Q

Moves a job to a specified queue. For example, when one of the events of a job in the Wait queue occurs, a MARK.IN.Q operation is performed to move the job to the queue specified in the RS_NEXT_Q field.

HANG.PROGRAM

Places the SMCP or a job in the mix in the Wait queue. For example, the SMCP can hang on TIME or I/O COMPLETE. The SCMP and jobs in the mix can hang on a list of events specified in a WAIT statement.

CAUSE.PROGRAM

An event has occurred; this operations locates and wakes up all programs waiting on the event. Using the address of the event, CAUSE.PROGRAM looks at the SMCP and all jobs in the mix. For each program waiting on the event, it reports the event and checks intervention as follows.

If the RS_REPORT_EV_INX bit is TRUE, the result of the event in the ES_REINSTATE_MSG_PTR is reported. If the RS_INTERVENTION bit is TRUE, a master processor rescheduling operation is forced and the program is moved to the SMCP communicate queue; otherwise, a master and slave processor rescheduling operation is forced and the program is moved to the queue specified in the RS_NEXT_Q field

BLOCK.SLAVE

Dispatches a port interrupt to the slave processor to block the scheduler from running a specific job or from running any jobs on the slave processor.

UNBLOCK.SLAVE

Dispatches a port interrupt to the slave processor to unblock the scheduler from running a specific job or from running any jobs on the slave processor.

REHANG.PROGRAM

Checks the event list for a program to see if any events on which it was waiting have occurred. If an event has occurred, it is reported and intervention is checked, as specified above in the description of CAUSE.PROGRAM.

PURGE.CACHE.MEMORY

Dispatches a port interrupt to the slave processor to purge the slave cache memory.

This is requested by the SMCP when microcode has been overlaid to prevent the slave from executing code that is no longer present in S-memory.

UPDATE.LAMPS

Updates the information in the 24 main exchange lights on the system console for system performance monitoring.

MEMORY.MANAGEMENT.FUNCTIONS

If the MPRI and THR system options are set, the HINTS.OVERLAY.COUNTER is bumped.

MAKE.TRACE.ENTRY

Makes an entry into the GISMO trace table.

REWIND.CASSETTE

Rewinds the console cassette tape drive.

APPENDIX D
MEMORY ORGANIZATION

The major data structures used by the SMCP program to keep track of its own state and the state of the user tasks in the mix are delineated in this appendix. These structures include the cold start variables, the SMCP global variables (HINTS), the run structure nucleus (RSN), and the environment structure nucleus (ESN).

SOFTWARE CONFIGURATION

Figure D-1 shows how the SMCP locates the run structures and environment structures for the various tasks in the mix. The HINTS.FIRST_QUEUE field points to the run structure nucleus (RSN) for the first task in the mix.

The RSN RS_Q_LINK field points to the RSN of the next task. The RSN RS_ENV_DIC field points to the environment dictionary. The RSN RS_FIB_DIC field points to the FIB dictionary.

The environment dictionary points to the primary and secondary environment for the task and also contains processor times for each environment. The primary environment points to the code and data for the task. The secondary environment points to code and data of a process that performs system functions on behalf of the task.

Following are the environment numbers and the use to which each is put.

Environment 0
The primary environment for the SMCP program and all tasks.

Environment 1
Not used. Reserved for future development.

Environment 2
A secondary environment for DMSII programs.

Environment 3
A secondary environment for IBASIC programs.

A page dictionary exists only for programs that page their code segments. A data dictionary exists only for programs that have paged data. BASIC, COBOL68 and FORTRAN are programs of this type. SDL and SDL2 programs with paged arrays use page tables within the program's dynamic memory. These tables are managed by intrinsics that use the data overlay communicate (type 13).

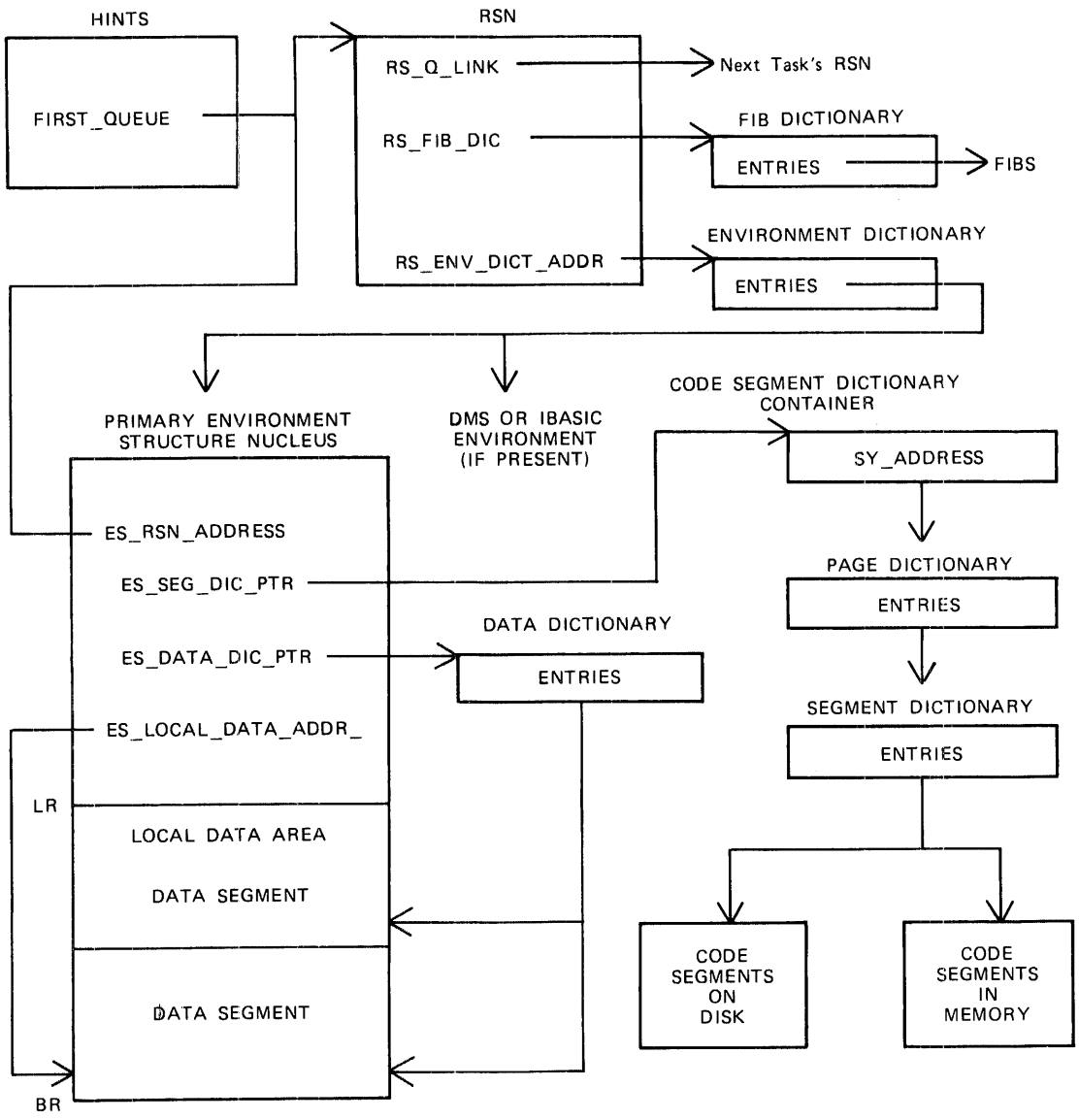


Figure D-1. Software Configuration

COLD START VARIABLES

The cold start variables include information on the date, time, and the system options. A programmatic description follows.

```
RECORD 1 PRO_ARRAY              BIT(16),
    2 PCU                       BIT(12),
    2 LABEL_TYPE                BIT(2),
          % 0 = ANSII
          % 1 = UNLABELED
          % 2 = BURROUGHS
    2 TRANSLATE                 BOOLEAN,% 0=EBCDIC   1=ASCII
    2 CLEAR_START               BOOLEAN;

CONSTANT MAX_SYSTEM_DISKS  = 16;

TYPE  SYS_DISK_INDEX_TYPE  = BIT(4); % for indexes into CSV.SUS(SDI)

RECORD 1 SYSTEM_UNITS_ARRAY BIT(16),
    2 SYS_PCU           PCU_LAYOUT,
    2 AVL_DISP          BIT(4);

%%%%% CONSTANT CSV_SIZE; See the CONSTANT below the CSV record

RECORD COLD_START_VARIABLES
        CLEAR_START_FLAGS               RECORD

        % Add value in the field to the corresponding index. For
        % example, index into name table for the current network
        % controller is  CONTROLLER_INDEX + CSV.CS_CONTROLLER.
        % See CM_GRINDER (mod 27).

          FILLER                            BIT(8),
          CS_INTERP                         BIT(4),
          % Which interpreter. 0=>I, 1=>IX.
          CS_MCP                            BIT(4),
          % Which MCP. 0=>M, 1=>MX.
          CS_GISMO                          BIT(4),
          % Which Gismo. 0=>G, 1=>GX.
          CS_INIT                           BIT(4),
          % Which Initialiser. 0=>N, 1=>NX.
          FILLER                            BIT(4),
          CS_MICRO_MCP                      BIT(4),
          % Which micro mcp. 0=>MM, 1=>MMX.
          CS_CONTROLLER                     BOOLEAN,
          %TRUE=>CX, FALSE=>C is active.
          CS_MCS                            BOOLEAN,
          %TRUE=>MCX, FALSE=>MCS is active.
          CS_ODT                            BOOLEAN,
          %TRUE=>ODX, FALSE=>ODT is active.
          FILLER                            BIT(5)
       END,
```

```
NAME_TABLE                          BIT(36),
% Disk address of the name table.
INTERP_DIC_ENTRIES                  BIT(24),
CS_SIZE                             BIT(24),
DUMP_FILE                           BIT(36),
CSV_COLD_START_LEVEL                RECORD
   L61_NAME_TABLE                      BOOLEAN,
   L10_0_NEW_SYS_DISK_TABLES           BOOLEAN,
   L10_0_NAME_TABLE                    BOOLEAN,
   L11_0_NAME_TABLE                    BOOLEAN,
   FILLER                              BIT(20)
END,
GISMO_TRACE_FLAGS                   RECORD
   [CHANNELS                           BIT(15)
   ICHANNEL                            (15) BOOLEAN],
   GISMO                               BOOLEAN,
   PORT                                BOOLEAN,% 16.
   USER                                BOOLEAN,% 17.
   MMCP                                BOOLEAN,% 18.
   SMCP                                BOOLEAN,% 19.
   GISMO_FREQUENT                      BOOLEAN,% 20.
   TIMESTAMP                           BOOLEAN,% 21.
   DATA_TRANSFER                       BOOLEAN,% 22.
   TABLE_LOCKED                        BOOLEAN % 23.
     %  This is used by the Gismo trace mechanism so that a dual
     %  processor system is correctly traced.
                                         END,
DUMP_FILE_SIZE                      BIT(16),
CORRECTABLE_ERROR_TABLE_LENGTH      BIT(16),% 40 + 32*#ENTRIES
MPF_TABLE                           BIT(36),
LOG_MIX_INFO                        BIT(36),
DISK_AVAIL                          BIT(36),
DISK_DIRECTORY                      BIT(36),
TEMP_TABLE                          BIT(36),
SY_DATE                             RECORD  % FOR SYSTEM/ODT RECORDS
   SY_DAY                              BIT(5),
   SY_MONTH                            BIT(4),
   SY_YEAR                             BIT(7)
END,
SY_JDAY                             BIT(9),
SY_TIME                             RECORD
   SY_HOUR                             BIT(5),
   SY_MIN                              BIT(6),
   SY_SEC                              BIT(6),
   SY_10THSEC                          BIT(4)
END,
SY_12HOUR                           BIT(5),
SY_DAYNAME                          CHARACTER(9),
SY_MERIDIAN                         CHARACTER(2),
SYSTEM_OPTIONS                      RECORD
   LOG_OPTION                          BOOLEAN,
   CHARGE_OPTION                       BOOLEAN,
   LIB_OPTION                          BOOLEAN,
   OPEN_OPTION                         BOOLEAN,
   TERM_OPTION                         BOOLEAN,
   TIME_OPTION                         BOOLEAN,
   DATE_OPTION                         BOOLEAN,
   CLOSE_OPTION                        BOOLEAN,
   PBT_OPTION                          BOOLEAN,
   PBD_OPTION                          BOOLEAN,
   BOJ_OPTION                          BOOLEAN,
   EOJ_OPTION                          BOOLEAN,
   SCHM_OPTION                         BOOLEAN,
   LAB_OPTION                          BOOLEAN,
```

```
    RMOV_OPTION                     BOOLEAN,
    DUMP_OPTION                     BOOLEAN,
    ZIPP_OPTION                     BOOLEAN,
    MEM_OPTION                      BOOLEAN,
    SWO1_OPTION                     BOOLEAN,
    SWO2_OPTION                     BOOLEAN,
    SWO3_OPTION                     BOOLEAN,
    LTB_OPTION                      BOOLEAN,
    AMCS_OPTION                     BOOLEAN,
    FILLER                          BOOLEAN,
    TRMD_OPTION                     BOOLEAN,
    DEBUG_OPTION                    BOOLEAN,
    DISP_OPTION                     BOOLEAN,
    ODTL_OPTION                     BOOLEAN,
    RMSG_OPTION                     BOOLEAN,
    SQRM_OPTION                     BOOLEAN,
    COPY_OPTION                     BOOLEAN,
    BREL_OPTION                     BOOLEAN,
    MPRI_OPTION                     BOOLEAN,
    THRASHING_OPTION                BOOLEAN,
    FLMP_OPTION                     BOOLEAN,
    VLCP_OPTION                     BOOLEAN,
    VLIO_OPTION                     BOOLEAN,
    FILLER                          BOOLEAN,
    WFL_OPTION                      BOOLEAN,
    SWD_option                      BOOLEAN,
    SWE_option                      BOOLEAN,
    SWF_option                      BOOLEAN,
    FILLER                          BIT(38)
END,
FIRST_SCHED_ENTRY                   BIT(36),
FIRST_WAITING_SCHED                 BIT(36),
MIX_LIMIT                           BIT(8),
SYSTEM_UNITS                        RECORD
    SUS (MAX_SYSTEM_DISKS)            SYSTEM_UNITS_ARRAY
END,
MASTER_IOAT                         DSK_ADR,
MASTER_DISK_AVAIL                   DSK_ADR,
NEXT_LOG_REC                        DSK_ADR,
LG_SIZE                             WORD,
NEXT_ELOG                           DSK_ADR,
ELOG_SIZE                           WORD,
JOB_NO                              WORD,
PBD_NO                              WORD,
ODT_Q_SIZE                          WORD,
CTLDCK_NO                           WORD,
LOG_NO                              WORD,
Q_DISK                              DSK_ADR,
TRACE_FPB                           DSK_ADR,
AUTO_MASK_ARRAY                     RECORD  % 4 OF 7 EACH=PRT PC
    AUTO_MASK (4)                     BIT(7)
END,
AB_NUMBER                           BIT(3),% NUM OF SYSTEM/BACKUPS
PBD_BLCKS_AREA                      WORD,
LG_LAST_AREA                        BOOLEAN,
ELOG_LAST_AREA                      BOOLEAN,
dl_backup_designation               NAME,
ODT_Q                               DSK_ADR,
PROTECTED_UNITS                     RECORD
    PRO_ELEMENT (16)                  PRO_ARRAY
END,
```

```
        SYS_LOG_NUMBER                  BIT(24),
        JOB_ACCTING_NUMBER              BIT(24),
        SESSION_NR                      BIT (16),
        FILLER                          BIT(36),% WAS THE XM TABLE ADDR
        CSV_NSEC_DISABL_THRASH_FAULT    BOOLEAN,
        CSV_OVERLAY_RATE                BIT(6),
        CSV_THRASHING_SENSITIVITY       BIT(8),
        CSV_HOSTNAME                    CHARACTER(17),% BNA.
        NEXT_HOSTNAME                   CHARACTER(17),
        SYSTEM_PROGRAM_PACK             NAME,
        ERROR_RATE_ADDR                 DSK_ADR,
        LOG_MIX_INFO_SIZE               BIT(16),
        RIB_NUMBER                      BIT(24),
        dl_dump_designation             NAME;

CONSTANT   CSV_SIZE    =     TYPE_LENGTH(COLD_START_VARIABLES);

DECLARE CSV COLD_START_VARIABLES REFERENCE;
```

HINTS

The MCP global data is kept in a structure named HINTS. The data is used as a means of communication among the SMCP, the MMCP, and GISMO. The HINTS structure also provides global storage for information that those programs need to save between the time periods they have control of the processor.

The HINTS structure starts at memory address zero. The comment field immediately to the right of each variable declaration is the absolute address in hexadecimal of that variable.

The following is a programmatic description of the HINTS structure.

```
RECORD   HINTS_RECORD
 %The order of the following field declarations and their address must
 %not be changed. The first data declaration must consist of this
 %record.
 %The first field is required by port interface hardware. Most of the
 %rest are for the system memory dump analyzer (SYSTEM/IDA), although
 %some fields are needed by gismo and the MMCP.
 %The first three fields are also used by clear start to keep the slave
 %out of harms way whilst initialisation is in process, and there are
 %insufficient structures and code present to control it. The source of
 %clear start should be consulted for final details. However, the
 %memory (starting at address 0) is used like this:-
 %    0005       %  purge cache memory, so that the slave's code can be
 %               % changed by the master. (This does not purge the
 %               % master processor's cache - only the slave's.)
 %    94000000   %  move 24 bit literal to the A register. The six zeros
 %               % are changed (by the master) to the address of the
 %               % scratch pad and A-stack dump routine. When finished
 %               % the slave resets the literal to zero.
 %    000000     %  initialised to zero by the master, but used as a
 %               % parameter for master-slave communication.
   DISPATCH_WORD                  BIT(024)%  HEX  000
 %  Reference address of I/O descriptor for dispatch.
 %  For I/O initiate, the descriptor to be initiated.
 %  Port devices yield address of I/O just completed.
,  ADDR_DP_PROC2                  BIT(024)%  HEX  018
,  FILLER                         BIT(024)%  HEX  030
,  MASTER_GISMO                   BIT(024)%  HEX  048
     %  Entry point to gismo. Used by interpreters and MMCP during
     % transfer of control to gismo.
,  LOCN_MAKE_MCP_BE_HERE          BIT(032)%  HEX  060
     %  Code address of MAKE_MCP_BE_HERE. Set during initialisation and
     % never changes. Used by the SDL2 interpreter when a needed MCP
     % code segment is not in memory. The stacks are faked up so that
     % return from MAKE_MCP_BE_HERE returns to code as if the segment
     % had been present all the time.
,  NO_SLAVE_DMS                   BIT(001)%  HEX  080
     %
     %
,  LOCN_INTERP_DICT               BIT(024)%  HEX  081
     % Address of the interpreter dictionary. Used by Gismo when
     % transfering control to the MMCP or an interpreter.
```

```
,   KI_KO                          BIT(001)%  HEX  099
      %
      %
,   NO_REINSTATES                  BIT(001)%  HEX  09A
      % Not used. Replaced by SCHEDULER_BLOCK_COUNT.
,   FIRE_UP_CONTROLLER             BIT(001)%  HEX  09B
      % Set when the NDL network controller is required. Tested in the
      % outer loop. (To avoid calling program initiation inline and the
      % associated stack space requirements.)
,   N_SECOND_COUNTER               BIT(002)%  HEX  09C
      % Count of number of passes through N_SECOND. Every fourth
      % N-second (when = 0), update the cold start variables on disk.
      % Primarily to ensure that the system date and time are reasonably
      % accurate in the event of a system crash.
, [ MCP_ESN_ADDR                   BIT(024)%  HEX  09E
      % Address of the MCP's environment structure nucleus (ESN).
    ! MCP_LIMIT                    BIT(024)%  HEX  09E
      % Top end of MCP writeable memory.
  ]
,   LAST_OVLY                      BIT(024)%  HEX  0B6
      % Points to the memory link of the most recently overlaid memory.
,   MICR_DEBUG_BIT                 BIT(001)%  HEX  0CE
      %
,   Q_NOT_LOCKED                   BIT(001)%  HEX  0CF
      % Event bit that is caused (set TRUE, and waiting processes
      % checked) when the queue subsystem is available.
      % Checked by SMCP and MMCP.
,   DFH_DIR_AD                     BIT(024)%  HEX  0D0
      % Address of the disk file header (DFH) dictionary. See also
      % DFH_DIR_LTH, MPF_DIR_AD, MPF_DIR_LTH.
,   AUTO_GUARD                     BIT(003)%  HEX  0E8
      % Used to control initiation of SYSTEM/BACKUP for autobackup
      % functions.
,   FIRE_SYSTEM_BACKUP             BIT(003)%  HEX  0EB
      % When a clear start is not required, N_second checks this field
      % and if non zero, initiates a copy of SYSTEM/BACKUP for
      % autobackup, and decrements the count. Thus the autobackups are
      % scheduled one per N-second, and the system is not overloaded.
,   FOUND_BACKUP_DESIGNATION       BIT(001)%  HEX  0EE
      % DL BACKUP pack is alive and well and on the system.
,   INTERRUPT_DISABLE_BIT          BIT(001)%  HEX  0EF
,   TRACE_CONDITIONAL_HALTS        BIT(001)%  HEX  0F0
      % SMCP conditional halts are to be entered into the gismo trace
      % table.
,   TRACE_HALTS_SETTUP             BIT(001)%  HEX  0F1
,   RAM_DISK_DISABLED              BIT(001)%  HEX  0F2
,   SYSTEM_OVRLAY_COUNTS           BIT(024)%  HEX  0F3
,   FIRST_QUEUE                    BIT(024)%  HEX  10B
      % Address of the first user run structure nucleus (RSN). The user
      % RSN's are a linked list ordered by processor priority.
,   ADDR_OF_COLD_START_VAR         BIT(024)%  HEX  123
      % Address of the memory copy of cold start vaiables (CSV).
,   ADDR_OF_INTERRUPT_INFO         BIT(024)%  HEX  13B
      % Prior to 12.0, address of the interrupt queue header, and thus
      % the interrupt queue. 12.0 uses the hardware MAXS register to
      % find the interrupt queue, which is part of gismo's data area.
      % The interrupt queue is maintained by gismo.
      % The SMCP requests information from the interrupt message
      % queue by using the FETCH SDL2 verb. The SDL2 interpreter then
      % calls gismo with the relevant swapper value.
,   FILLER                         BIT(003)%  HEX  153
,   IN_CCD                         BIT(001)%  HEX  156
      %TRUE => The MCP is performin a task in control card driver for
      %         WFL.
```

```
,   ELOG_FULL                       BIT(001)%  HEX  157
      %TRUE => the ELOG is nearly full (or full). At the next N-second
      %         pass, try to transfer the ELOG, and get more space.
,   NON_RELEASE_MCP                 BIT(001)%  HEX  158
      % The MCP sets this field when extra checking and debugging code
      % has been compiled into the MCP. Primarily intended for use by
      % plant personnel.
      %
      %
,   GISMO_LEVEL                     BIT(008)%  HEX  159
      %
,   HI_RESOLUTION_TIMER_SET         BIT(001)%  HEX  161
      %TRUE => special T/O control (channel 15) for high precision
      %         timings is present. Superceded by the TIME register on
      %         M3 (B1860) and later processors.
,   PSR_CHANGE_BIT                  BIT(001)%  HEX  162
      %TRUE => pseudo readers should be examined for work at the next
      %         N-second pass.
, [ RELEASE_LEVEL                   BIT(008)%  HEX  163
      %
    ! MARK                            BIT(004)%  HEX  163
      %
,     LEVEL                           BIT(004)%  HEX  167
      %
  ]
,   FIRST_LINK                      BIT(024)%  HEX  16B
      % Address of the first memory link.
,   MCP_TYPE                        BIT(004)%  HEX  183
      % Not used.
,   SYCOUNTER                       BIT(020)%  HEX  187
      % Time of day in tenths of a second. Maintained every timer
      % interrupt by gismo without requiring the running process to save
      % state.
,   SY_CNTR_MSK                     BIT(020)%  HEX  19B
      % When SYCOUNTER (q.v.) reaches this value (i.e. time) wake up
      % the SMCP. The micro scheuler in gismo, after handling
      % interrupts, running a reader sorter use routine if required,
      % ensures that the SMCP will be selected when SY_CNTR_MSK is less
      % than SYCOUNTER.
,   SY_PRIOR_TIME                   BIT(020)%  HEX  1AF
      % Previous N-second setting of SYCOUNTER.
,   RESERVED_FOR_INTERP_USE         BIT(020)%  HEX  1C3
      %
,   SYSTEM_PACK_INFO                BIT(024)%  HEX  1D7
      % Points to a linked list of pack info tables. One table for
      % every ready pack on the system.
,   RED_ALERT___RED_ALERT           BOOLEAN%   HEX  1EF
      %TRUE => non standard software has been supplied by the plant to
      %         a specific site.
      %
      %
```

```
,   FILLER                          BIT(6)%    HEX  1F0
, [ SYSTEM_UNIT                     BIT(012)%  HEX  1F6
      % Absolute disk address of the primary system disk.
    ! [ SYSTEM_PORT_CHAN                BIT(007)%  HEX  1F6
          %
      ! SYSTEM_PORT                     BIT(003)%  HEX  1F6
          %
,       SYSTEM_CHANNEL                  BIT(004)%  HEX  1F9
          %
      ]
,     DUMMY_BIT_RESERVED              BIT(001)%  HEX  1FD
      % The serial number flag in a disk address must always be reset
      %   here.
,     SYSTEM_UNIT_EU                  BIT(004)%  HEX  1FE
        %
  ]
, [ CONSOUL_SWITCHES                  BIT(028)%  HEX  202
    ! SE_ENABLE_VALUE                 BIT(4)   %  HEX  202
  ]
,   MICRO_TRACE_FLAG                  BIT(001)%  HEX  21E
,   GISMO_TRACE_SPACE                 BIT(024)%  HEX  21F
      % Address of the gismo trace header. The header consists of
      %   TRACE_SIZE                    BIT(24)
      %   TRACE_OFFSET                  BIT(24)
      % The 24 bits below the header is the trace mask.
, [ PORT_CHANNEL_TABLE                BIT(192)%  HEX  237
    ! PCT_ENTRY (8)                   BIT(024)%  HEX  237
      % This array is subsripted by PORT number to yield the address
      % of the channel table for that port. Processor ports do not
      % have channel tables. The soft I/O subsystem, which interfaces
      % to the master processor via the CMND and DATA registers, is
      % deemed to be on port 7.
  ]
,   BYPASS_CLEANUP                    BIT(001)%  HEX  2F7
      %TRUE => system panda can be removed from the name table.
      %
      %
,   CONTRL_CRD_FLG                    BIT(001)%  HEX  2F8
      %TRUE => MCP is in control card driver..
,   EXT_RESULT_DESC_CHAIN             BIT(024)%  HEX  2F9
      % Points to the first extended result descriptor in the chain.
      % See also EXT_RESULT_SAVED.
,   T_FILES                           BIT(008)%  HEX  311
      % Count of number of temporary disk files (i.e. in the directory
      % but DFH_PERMANENT = 1) encountered when cleaning up a disk.
,   MICR_COUNT                        BIT(006)%  HEX  319
      % Count of the number of reader sorter files open. When the
      % first file is openned, hi-priority interrupt handling code is
      % made present in memory and marked save.
      % This code is segments 0 and 6 of the MMCP, and all the user
      % interpreter external segments.
,   CHANGE_BIT                        BIT(001)%  HEX  31F
      %TRUE => check the active schedule upon return to the ouuter
      %         loop.
,   RELEASE_VERSION                   BIT(008)%  HEX  320
,   SQ_IN_PROGRESS                    BIT(001)%  HEX  328
      % A squash of the system disk is in process, and most other
      % activity on the system shouuld be stopped. Only zip input, and
      % AC, AX, DM, DP, DS or LP messages are allowed for the program
      % squashing the system disk (see SQ_JOB_NUMBER).
,   IOAT_POINTER                      BIT(024)%  HEX  329
      % Starting memory address of the input output assignment table.
```

```
,   IOAT_END                        BIT(024)%  HEX  341
      % Ending address + 1 of the IOAT.
,   SYSTEM_PAUSE_DESC               BIT(024)%  HEX  359
      % Reference address of the pause descriptor in the disk chain.
      % The pause descriptor is in SYS_PAUSE_DESC.
,   PSEUDO_TABLE_ADDRESS            BIT(024)%  HEX  371
      % Address of pseudo readers in memory.
,   EMERGENCY_ODT_Q_REGENERATION    BIT(001)%  HEX  389
      %TRUE => rebuild the ODT queue at clear start. This must be set
      %        manually.
,   SQUASH_STARTING                 BIT(001)%  HEX  38A
      % A squash of the system disk has been requested. This is set in
      % the procedure SQ.
, [ GISMO_OPTIONS                   BIT(024)%  HEX  38B
      % Certain portions of gismo code are discarded depending on the
      % system configuration and MCP options. This field records what
      % was kept.
    ! FILLER                          BIT(001)%  HEX  38B
,     COMM_TRACE                      BIT(001)%  HEX  38C
,     GISMO_TRACE                     BIT(001)%  HEX  38D
,     FILLER                          BIT(001)%  HEX  38E
,     PORT_DEVICES                    BIT(001)%  HEX  38F
,     B1720_CODE                      BIT(001)%  HEX  390
,     B1820_CODE                      BIT(001)%  HEX  391
,     MPROC_CODE                      BIT(001)%  HEX  392
,     B1830_CODE                      BIT(001)%  HEX  393
,     PRIORITY_MEMORY_MGMT            BIT(001)%  HEX  394
,     THRASHING_COUNTING              BIT(001)%  HEX  395
,     LAMP_CPU_BASE                   BIT(001)%  HEX  396
,     FIXED_LAMP_DISPLAY              BIT(001)%  HEX  397
,     VAR_LAMP_BASE                   BIT(001)%  HEX  398
,     FILLER                          BIT(001)%  HEX  399
,     VAR_LAMP_IO                     BIT(001)%  HEX  39A
,     FILLER                          BIT(001)%  HEX  39B
,     DCPU_CODE_PRESENT               BIT(001)%  HEX  39C
,     FILLER                          BIT(006)%  HEX  39E
  ]
,   DCH_SCRATCH_MEM_ADDR            BIT(024)%  HEX  3A3
      % Points to DCH_SCRATCH_MEMORY, which contains various variables
      % for NDL, such as the addresses of the network controller's
      % remote FIB, station FIB. For the complete list see
      % DCH_SCRATCH_RECORD.
,   TASK_IN_SCHED                   BIT(001)%  HEX  3BB % FOR WFL TASKS
,   INTERRUPT_SWITCH_SET            BIT(001)%  HEX  3BC
,   DISABLE_INTERRUPT_SW            BIT(001)%  HEX  3BD
,   FIRE_ODT_ROUTINES               BIT(001)%  HEX  3BE
      %TRUE => the next pass through N-second must fire up system/odt.
,   JOBS_RUNNING                    BIT(012)%  HEX  3BF
      % Count of jobs actually running. Used in N-second to calculate
      % the N-second interval. Value is 5 seconds per job, with an
      % upper limit of 1 minute.
      % The number of jobs controlled by the mix limit (ML) is in
      % LIMITED_JOBS, a variable on the SMCPs data stack.
,   BEEN_THRU_MCP_BE_HERE           BIT(001)%  HEX  3CB
      %TRUE => have already tried to bring in an MCP code dictionary
      %        and dictionary. We are not suceeding, therefore, rather
      %        than on a memory request that cannot be satisfied, and
      %        one which cannot wait, we reluctantly halt the system.
      %
,   REMOTE_REROUTE                  BIT(001)%  HEX  3CC
      %TRUE => gismo communicate router should send remote file reads
      %        and writes to the SMCP rather than the MMCP.
```

```
,   QUEUE_REROUTE                   BIT(001)%  HEX  3CD
      %TRUE => gismo communicate router should send queue file reads
      %         and writes to the SMCP rather than the MMCP.
      % Whenever the SMCP uses queues (e.g. autobackup, job spawning
      % replies), the SMCP code is always invoked.
,   DISK_MONITOR_GISMO              BIT(004)%  HEX  3CE
,   SAVE_DUMP                       BIT(001)%  HEX  3D2
      % Save the system memory dump (SYSTEM/DUMPFILE), even if the size
      % of SYSTEM/DUMPFILE is incorrect. The size of the system dumpfile
      % depends on:
      %   - memory size of the system
      %   - the size of the MCP layout tables (these allow symbolic
      %     analysis of the dump)
      %   - the size of the ODT queue
,   FILLER                          BIT(036)%  HEX  3D3
      % Was INTERPRETER_TABLE_ADDRESS prior to 12.0.
      % This is still used by SYSTEM/IDA, to allow 11.0 dumps to be
      % analysed (to a certain extent).
,   ODT_PORT_CHAN                   RECORD%    HEX  3F7
      % For a system using an ODT I/O control, this field contains that
      % port and channel. For a system not using an ODT I/O control,
      % this field is zero. The ODT control can be masked out (see
      % CHANNELS_NOT_PRESENT) on a non gem if required.
        ODT_PORT                      BIT(003)%  HEX  3F7
,       ODT_CHANNEL                   BIT(004)%  HEX  3FA
     END

,   KEYBOARD_ODT_DESC               BIT(024)%  HEX  3FE
      % Reference address of the I/O descriptor used to communicate
      % with an ODT I/O control.
,   LOCN_BIOAW_RECOVER              BIT(024)%  HEX  416
      % Not used.
,   CHANNELS_NOT_PRESENT            BIT(016)%  HEX  42E
      % At clear start, channels can be ignored by use of the FA
      % register (on non gem) or the TEXT IC message (on gem systems).
      % This can be useful when a control has a problem and prevents
      % either a succesful clear start, or trouble free running.
,   LAMP_DATA_PTR                   BIT(048)%  HEX  43E
, [ SEGMENT_HALT                    BIT(004)%  HEX  46E
    ! SPECIFIC_HALT                 BOOLEAN%   HEX  46E
      % See HALT_MASK.
,     DUMP_AT_HALT                  BOOLEAN%   HEX  46F
      % If either a SPECIFIC_HALT or STOP_AT_ALL_HALTS, take a full
      % system memory dump, clear this flag and then halt.
      % (DUMP_AT_HALT is reset to avoid cloberring a previous dump.)
      % See the SH S @hhhhhh@ D message.
,     TRACE_HALTS                   BOOLEAN%   HEX  470
      %  Conditional halts are to be entered into the gismo trace
      % table.
,     STOP_AT_ALL_HALTS             BOOLEAN%   HEX  471
      %TRUE => stop at all conditional halts.
  ]
,   HALT_MASK                       BIT(024)%  HEX  472
      % Leftmost six digits of the SMCP sequence number (in hex) at
      % which to halt when the SPECIFIC_HALT flag is TRUE. See the SH
      % message.
      % NB. this must match a CONDITIONAL_HALT sequence number.
      % The value is in hex so that in an emergency the values can be
      % easily entered from the console.
, [ MMCP_SEGMENT_HALT               BIT(004)%  HEX  48A
    ! SPECIFIC_MMCP_HALT            BOOLEAN%   HEX  48A
      % See MMCP_HALT_MASK.
,     FILLER                        BOOLEAN%   HEX  48B
```

```
,     TRACE_MMCP_HALTS              BOOLEAN%  HEX  48C
      % MMCP conditional halts are to be entered into the gismo trace
      % table.
,     STOP_AT_ALL_MMCP_HALTS        BOOLEAN%  HEX  48D
      %TRUE => stop at all MMCP conditional halts.
  ]
,   MMCP_HALT_MASK                  BIT(024)% HEX  48E
      % Leftmost six digits of the MMCP sequence number (in hex) at
      % which to halt when the SPECIFIC_MMCP_HALT flag is TRUE. See the
      % SH M message.
      % NB. this must match a CONDITIONAL_HALT sequence number.
      % The value is in hex so that in an emergency the values can be
      % easily entered from the console.
, [ COMPILE_TIME_OPTIONS            BIT(008)% HEX  4A6
      % Records compile time options of the SMCP.
    ! RELEASE_VERSION_MCP             BIT(001)% HEX  4A6
        %  The RELEASE option was set.
,     DEBUG_OPTION                    BIT(001)% HEX  4A7
        %  The RELEASE option was reset.
,     FILLER                          BIT(006)% 6 MORE OPTIONS
  ]
, [ ENVIRONMENT_OPTIONS             BIT(004)% HEX  4AE
      %
    ! USE_SLIO_ENV                  BIT(001)% HEX  4AE
      % Use the SDL2 logical I/O environment.
,     USE_DATACOMM_ENV              BIT(001)% HEX  4AF
      % Use the SLTO datacomm and queue environment.
,     USE_ISAM_ENV                  BIT(001)% HEX  4B0
      % Use the SLIO ISAM environment.
,     FILLER                        BIT(001)% HEX  4B1
  ]
,   MCP_VERSION_DATE                BIT(016)% HEX  4B2
      % Compilation date of MOD02. Since this module should be
      % recompiled for every patch to the MCP, this should yield the
      % MCP compile date.
      % Format YY bit 7, MM bit 4, DD bit 5.
,   DMS_MM_EVENT                    BIT(001)% HEX  4C2
      % When the DMCP cannot proceed because memory management is
      % active this event bit is FALSE, and the DMCP is hung on it.
,   DMS_MM_COUNT                    BIT(006)% HEX  4C3
      % When memory management is about to overlay a DMS buffer, this
      % field is bumped. If the new value is one, DMS_MM_EVENT is reset
      % so that the DMCP will not try to use buffers.
      % DELETE_DMS_BUFFER (in ALLOCATE_S_MEMORY) then performs
      % additional checks.
      % When memory management has finished, this field is decremented,
      % and if the new value is zero, DMS_MM_EVENT is caused, to awaken
      % any DMCP which may be waiting.
,   CURR_INTERP_DIC_ENTRIES         BIT(005)% HEX  4C9
      % Size of the interpreter dictionary. Default size is seven (7).
      % The MMCP, GISMO and the SDL2 interpreter occupy entries 0, 1 and
      % 2 respectively. Thus four user interpreter slots are available.
      % The size can be changed with the IC ODT message.  A Clear/Start
      % is required to change the size.
,   DM_GLOBALS                      BIT(024)% HEX  4CE
      % Pointer to a linked list of DMS_GLOBALS.
,   QUEUE_ROOT                      BIT(024)% HEX  4E6
      % Points to the queue subsystem globals.
,   TOP_S_COMM_QUEUE                BIT(024)% HEX  4FE
      %
```

```
,   DC_CHAIN                        BIT(024)%  HEX  516
      % Points to a linked list of DC_POCKET_RECORDs, which remember
      % which job is using which adapter. Unusually, a null list is
      % indicated by 0.
,   TRUTH_TABLE_ADDR                BIT(024)%  HEX  52E
      % Contains the address of the truth table. This table contains a
      % bit for each patch number. Therefore, all the bits upto the SMCP
      % patch level should be on, and all those after should be off.
      % This is built and checked during clear start. If the table fails
      % the check, then we halt with L=@000011@, T=@333333@. The halt
      % itself is push through, but that decision should not be taken
      % lightly - patches are missing, and patches are quite often
      % interdependent. Contains zero when not set up.
,   FILLER                          BIT(012)%  HEX  546
,   S_C_Q_EV                        BIT(001)%  HEX  552
      % Smcp Communicate Queue EVent.
      %TRUE => There is at least one program waiting for the SMCP to
      %        handle the progam's communicate.
,   M_C_Q_EV                        BIT(001)%  HEX  553
      % Mmcp Communicate Queue EVent.
      % On 10.0 and earlier releases, TRUE => there is at least one
      % program waiting for the MMCP to handle the program's
      % communicate.
      % On 11.0 and later, gismo uses a different mechanism to awaken
      % the MMCP, and this field is not used.
      % If a program is running, and executes a communicate for the
      % MMCP, then the gismo communicate router will run the MMCP for
      % that program.
      % If a program is waiting in the MMCP communicate queue or the
      % I/O complete queue, the MMCP will be executed based on the
      % program's priority compared to other programs.
,   S_M_Q_EV                        BIT(001)%  HEX  554
      % Smcp Message Queue Event.
      %TRUE => there is a message for the SMCP in the interrupt
      %        queue. These messages are usually I/O complete
      %        messages (loosely interrupts).
,   S_I_Q_EV                        BIT(001)%  HEX  555
      % Smcp Interrupt Queue Event.
      %TRUE => there is a high priority message in the interrupt
      %          interrupt queue.
      %  This has not been used since 6.1 when high priority handling
      % (for reader sorters) was moved to the MMCP.
      % Kept here in case we ever want to put some high priority
      % processing in the SMCP.
,   M_M_Q_EV                        BIT(001)%  HEX  556
      % Mmcp Message Queue Event.
      %TRUE => there is a message for the MMCP in the interrupt
      %        These messages are always I/O complete messages
      %        (loosely interrupts).
,   M_I_Q_EV                        BIT(001)%  HEX  557
      % Mmcp high priority Interrupt Queue Event.
      %TRUE => there is a high priority interrupt for the MMCP.
      %        Currently only used for reader sorters.
,   M_CAUSE_LOCK                    BIT(001)%  HEX  558
      %
,   M_EV_FILLER                     BIT(005)%  HEX  559
      %
,   M_MCP_LR                        BIT(024)%  HEX  55E
      % A psuedo limit register (processor LR) for the master MMCP. See
      % MASTER_MMCP_DATA_PTR.
,   LOCK_ADDRESS                    BIT(024)%  HEX  576
      %
      %  by gismo.
```

```
,   MCP_RSN_ADDR                    BIT(024)%  HEX  58E
      % Memory adddress of the SMCP's run structure nucleus (RSN).
, [ DISK_TABLE                      BIT(024)%  HEX  5A6
      % Contains the drive transformation vector which is loaded to the
      % DSC to allow drive numbers to reassigned at no cost to the MCP
      % or gismo. The vector is actually loaded by system init. The
      % purpose here is to inform the user that drive transformation
      % has occurred.
    ! DRIVE_NBR (8)                   BIT(003)
  ]
,   TASK_MIX_NO                     BIT(014)%  HEX  5CE
,   JOB_MIX_NO                      BIT(014)%  HEX  5DC
,   WFL_QUEUE_ADDRESS               BIT(024)%  HEX  5EA
      % Address of the WFLQUEUE, which is the queue of WFL commands
      % between the MCP and SYSTEM/WFL. Also used by the RIB mechanism.
,   NC_QUEUE_ADDRESS                BIT(024)%  HEX  602
      % Not used yet.
,   FILLER                          BIT(044)%  HEX  61A
,   M_MCP_Q_IDENT                   BIT(008)%  HEX  636
      %
,   COMM_SPLITTER_ADDR              BIT(024)%  HEX  63E
      % Memory address of the array used by the communicate router in
      % gismo to route communicates to the appropriate part of the MCP
      % (4=>SMCP, 5=>MMCP, 2=>DMCP) or a special system environment
      % (3=>IBASIC). Each entry is 3 bits wide.
,   COMM_SPLITTER_LENGTH            BIT(016)%  HEX  656
      % Length of the comm_splitter_array. See above.
,   FIRST_RUN_UNIT                  BIT(024)%  HEX  666
      % Cobol74 style IPC uses the verb call. Passing parameters
      % between different tasks is permitted. Each such group of tasks
      % is called a run unit. This points to the first in the linked
      % list of run units.
,   INDEX_SEQ_USER_COUNT            BIT(008)%  HEX  67E
      % Count of the number of ISAM (Cobol 74 or RPG $IXSEQ) files
      % open.
, [ MIKES_HALT_SPACE                BIT(096)%  HEX  686
      % When the system comes to a controlled halt, the values of the
      % L, T, X and Y registers are stored here.
    ! FILLER                          BIT(048)
,     SMCP_HALT_NOMEM_SEQ_NO          BIT(032)%  HEX  6B6
,     FILLER                          BIT(016)
  ]
,   RIB_LIST                        BIT(024)%  HEX  6E6
      % Pointer to the linked list of routing information blocks
      % (RIBs).
,   LAST_LINK                       BIT(024)%  HEX  6FE
      % Memory address of the last memory link.
,   SMCP_CPU_PRIORITY               BIT(024)%  HEX  716
      % The current SMCP processor priority. The micro scheduler in
      % gismo uses this field rather than the field in the SMCP's rsn,
      % because since the RSN is above the MCP_LIMIT register, the SMCP
      % cannot write to the RSN field.
, [ LAMP_GLOBALS                    BIT(014)%  HEX  72E
    ! LAMP_SCALE                      BIT(003)%  HEX  72E
,   [ VL_ALLS                         BIT(003)%  HEX  731
      ! VL_AUCPU                        BIT(001)%  HEX  731
,       VL_AUCOLAY                      BIT(001)%  HEX  732
```

```
,       VL_AUDOLAY                          BIT(001)%  HEX  733
    ]
,     FILLERC                             BIT(001)%  HEX  734
,     VL_SSWC                             BIT(001)%  HEX  735
,   [ LAMP_OPTIONS                        BIT(004)%  HEX  736
      ! FLAMPS                              BIT(001)%  HEX  736
,     [ VLAMPS                              BIT(002)%  HEX  737
        ! VLAMPS_CPU_OLAY                     BIT(001)%  HEX  737
,         VLAMPS_IO                           BIT(001)%  HEX  738
      ]
,       VLAMPS_BAR_GRAPH                    BIT(001)%  HEX  739
    ]
,   [ VL_SMCP_OLAYF                       BIT(002)%  HEX  73A
      ! VL_SMCP_OLAY                         BIT(001)%  HEX  73A
,       VL_SMCP_OLAY_USE                    BIT(001)%  HEX  73B
    ]
  ]
,   JOBS_SWEEPS_BEFORE_DECAY          BIT(010)%  HEX  73C
, [ SYSTEM_ID                         BIT(012)%  HEX  746
    ! CPU_ID                             BIT(004)%  HEX  746
%     0=ERROR      1=B1710      2=B1720
%                  3=B1830      4=B1860
%                  5=B1900      6=B1900(GEM)
,     MEMORY_ID                           BIT(004)%  HEX  74A
%     0=DEFAULT    1=CORRECTABLE S-MEMORY PARITY
,     IO_ID                               BIT(004)%  HEX  74E
%     0=DEFAULT

  ]
,   ELOG_HERE                           BIT(024)%  HEX  752
,   QLOCK_COUNT                         BIT(004)%  HEX  76A
,   CHIP_TABLE_ADDRESS                  BIT(024)%  HEX  76E
,   MIX_MEMORY_PRIORITIES               BIT(016)%  HEX  786
      % Treated as MEMORY_PRIORITY(16) BOOLEAN. If there is a job in
      % the mix, with memory priority n, then MEMORY_PRIORITY(n) will be
      %TRUE.
,   STOP_SCHED_INPUT                    BIT(001)%  HEX  796
      % Do not bring any more tasks out of the active schedule.
      % Set when:
      %   - the system is thrashing (i.e. full)
      %   - a squash of the system disk is scheduled
      %   - the number of running jobs would exceeed MAX_TASKS.
,   NSEC_DISABL_THRASH_FAULT            BIT(001)%  HEX  797
,   DISABLE_THRASHING_FAULT             BIT(001)%  HEX  798
,   MCP_VARTABLE_MEM_PRIORITY           BIT(004)%  HEX  799
      % The current memory priority to be used for SMCP memory
      % requests. When performing work on behalf of a program, the SMCP
      % will set this field to that program's memory priority.
,   DOING_FILE_ATTRIBUTE_COMM           BIT(1)   %  HEX  79D
, [ GOTTA_DR_OR_TR                      BIT(2)   %  HEX  79E
      % Checked at N-second to remind the operator.
    ! GOTTA_DR                          BIT(1)   %  HEX  79E
        %  Waiting for an operator DR before releasing the schedule.
```

```
,     GOTTA_TR                        BIT(1)  %  HEX  79F
        %  Waiting for an operator TR before releasing the schedule.
  ]
,   MEM_SWEEP_PENDING                 BIT(001)%  HEX  7A0
      % A gismo memory sweep of memory links is required. The actual
      % sweep can be delayed a few clock due to pressure of other work.
,   SAMPLING_CLOCK                    BIT(006)%  HEX  7A1
,   SAMPLING_INTERVAL                 BIT(006)%  HEX  7A7
,   MEM_SWEEP_INTERVAL                BIT(010)%  HEX  7AD
,   MAX_SWEEP_INTERVAL                BIT(010)%  HEX  7B7
,   MEM_EXTEND_COUNT                  BIT(002)%  HEX  7C1
,   OVERLAY_COUNTER                   BIT(008)%  HEX  7C3
,   OVERLAY_TARGET                    BIT(008)%  HEX  7CB
,   MCP_SWEEPS_BEFORE_DECAY           BIT(010)%  HEX  7D3
,   MEM_DUMP_COMPLETE                 BIT(001)%  HEX  7DD
       % Set by system/init to record the fact that a system memory
       % dump was taken during clear start. During SMCP initialization,
       % this conditions the memory dump completed message.
,   MAX_MEM_PRIORITY_IN_MIX           BIT(004)%  HEX  7DE
       % Highest active memory priority. The SMCP uses this priority
       % when performing general housekeeping. The point is to avoid
       % introducing another active memory priority.
,   COMMAND_LOGGED                    BIT(001)%  HEX  7E2
      % Used by control card driver (CTRL_CARD_DRIVER) to ensure that
      % commands are entered in the WFL job log exactly once. Set when
      % the command has been logged.
,   CONTROLLER_SCHEDULED              BIT(001)%  HEX  7E3
      % Since the NDL newtork controller has been scheduled, please do
      % not fire up another copy if remote file open is received before
      % the network controller has initialized.
,   DCPU_ID                           BIT(002)%  HEX  7E4
      % Set by system/init to record which processor (A or B) in a dual
      % processor environment is master, and which is slave.
      MASTER                          BIT(001)%  HEX  7E4
      SLAVE                           BIT(001)%  HEX  7E5
,   TASK_TABLE_ADDR                   BIT(024)%  HEX  7E6
,   CLEAR_START_REQD                  BIT(001)%  HEX  7FE
,   SCHEDULER_BLOCK_COUNT             BIT(008)%  HEX  7FF
      % When <> 0, the micro scheduler in gismo will only run reader
      % sorter use routines, or the SMCP. Bumped by SMCP procedures that
      % desire to have complete control of the system for awhile.
,   FIRE_MCS                          BIT(001)%  HEX  807
,   DCPU_DATA                         BIT(024)%  HEX  808
,   MASTER_PORT                       BIT(003)%  HEX  820
,   SLAVE_PRESENT                     BIT(001)%  HEX  823
,   SLAVE_PORT                        BIT(003)%  HEX  824
,   FIRE_NDL                          BIT(001)%  HEX  827
,   WFL_JOB_NO                        BIT(016)%  HEX  828
,   [CACHE_BITS                       BIT(6)%    HEX  838
     ! NO_RAM_DISK_1                  BIT(1)%
,      NO_RAM_DISK_2                  BIT(1)%    HEX  839
,      NO_CACHE_1                     BIT(1)%    HEX  83A
,      NO_CACHE_2                     BIT(1)%    HEX  83B
,      CACHE_PRESENT_1                BIT(1)%    HEX  83C
,      CACHE_PRESENT_2                BIT(1)] %  HEX  83D
```

```
,   FILLER                          BIT(002)%  HEX  83E
,   BNA_ADDRESS                     BIT(024)%  HEX  840
,   REAL_MEMORY_SIZE                BIT(024)%  HEX  858
      % Highest memory address (in bits) on the system (i.e. MAXS-1).
      % Note that both processors, and the memory base must have the
      % same settings.
,   PSEUDO_MEMORY_SIZE              BIT(024)%  HEX  870
      % Highest memory address (in bits) on the system. Will be less
      % than REAL_MEMORY_SIZE if the LR register was set at clear start.
      %  NB: It is no good trying to use this to find a L=@0D0055@
      %      problem since the memory is still present. ALL the relevant
      %      MAXS jumpers must be changed.
      %  L=@0D0055@ cannot occur on a 2MByte system. Therefore for
      % "strange" problems try reducing memory ON THE MAXS JUMPERS.
,   TIME_MARK                       BIT(024)%  HEX  888
,   MASTER_MMCP_DATA_PTR            BIT(024)%  HEX  8A0
      % Address of the master MMCP's data. This should match MMCP_LR.
      % Also used as a pseudo limit register in system halts.
,   SLAVE_MMCP_DATA_PTR             BIT(024)%  HEX  8B8
      % Address of the slave MMCP's data.
,   ODT_Q_FILE_ADDR                 BIT(024)%  HEX  8D0
      % Address of the queue descriptor for messages to SYSTEM/ODT. The
      % queue is called "FROM-MCP".
,   SYS_ODT_JOB_NO                  BIT(016)%  HEX  8E8
      % Job number of SYSTEM/ODT.
,   SYS_ODT_DIO_DISK_DESC_ADDR      ADDRESS%   HEX  8F8
      % Address of the I/O descriptor that SYSTEM/ODT uses to access
      % the disk area known as SYSTEM/ODT-QUEUE. This should not be
      % confused with a queue structure used for queue files.
,   SYS_PAUSE_DESC                  BIT(272)
      % The head of the disk chain. Used as a marker by Gismo, to
      % ensure that the full chain is searched at least once, and no
      % more than twice.
,   EXT_RESULT_EXISTS               BOOLEAN
      %TRUE => EXT_RESULT_SAVED and EXT_RESULT_HIT_DESC are valid.
,   EXT_RESULT_SAVED                BIT(96)
      % Saved extended result descriptor.
,   EXT_RESULT_HIT_DESC             BIT(175)
      % Saved left hand part of the I/O descriptor in error.
,   DFH_DIR_LTH                     WORD
      % Length of the disk file header dictionary. See also DFH_DIR_AD.
,   MPF_DIR_LTH                     WORD
      % Length of the Multi-Pack File dictionary.
,   MPF_DIR_AD                      ADDRESS
      % Address of the start of the Multi-Pack File dictionary.
,   LOCN_DESC_BIOAW_RECOVER         ADDRESS
      % Address of the system descriptor containing the code for
      % BIOAW_RECOVERY. Built during initialisation and never changes.
,   TRACE_HALT_DA                   ADDRESS
,   TRACE_HALT_BUF_PTR              ADDRESS
,   SQ_JOB_NUMBER                   BIT(16)
      % Job number of the system/squash program which is squashing the
      % system disk.
,   SQ_MSG_ADDR                     ADDRESS
      % Not used.
,   MASTER_IDLE_TIME                PROCESSOR_TIME
      % Total idle time for the master processor.
,   SLAVE_IDLE_TIME                 PROCESSOR_TIME
      % Total idle time for the slave processor.
,   SMCP_START_TIME                 PROCESSOR_TIME
      % Maintained by gismo. The SMCPs processor time at the start.
,   SMCP_SERVICE_TIME               PROCESSOR_TIME
      % Maintained by gismo. The total SMCP service time.
```

```
,   MEM_STATISTICS_ADDR              ADDRESS
      % Address of the table used to maintain memory usage statistics.
,   PROTECTED_FILE_COUNT             WORD
      % When the system disk is being cleaned up (in DISK_CLEAN_UP),
      % a count is kept of PROTECTION=PROTECTED files encountered.
      % Should this field be non zero, then the program in the PAN
      % slot in the name table (usually SYSTEM/PANDA), will be initiated
      % to handle correcting the end of file pointers for those files.
      %
, [ EXPIRED_PROC_TIMES                   BIT(144)
      % When a job is teminating (i.e. dying or expiring), it's various
      % processor times are added to these totals.
    ! EXP_PRIMARY_INTERP_TIME              BIT(24)
,     EXP_PRIMARY_SMCP_TIME              BIT(24)
,     EXP_PRIMARY_MMCP_TIME              BIT(24)
,     EXP_DMS_INTERP_TIME                BIT(24)
,     EXP_DMS_SMCP_TIME                  BIT(24)
,     EXP_IBASIC_INTERP_TIME             BIT(24)
  ]
;
DECLARE HINTS HINTS_RECORD;
 %       This must remain the first declare, so that the space appears
 %       first in the data stack, and thus at absolute address 0.
RECORD INTERP_DICT_ENTRY
          % One entry for each firmware file used by the running
          % system. Entry 0 is MMCP, 1 is GISMO, and 2 is the SMCP's
          % interpreter (SDL2/INTERP for 11.0, SDL2/INTERP1M for 12.0).
        SEG_DIC                        SYSTEM_DESCR,
          % The system descriptor for the interpreter's segment
          % dictionary (if any), else for the non-segmented code.
          % This descriptor is callled the ED or external descriptor.
        FILLER                         BIT(8),
          % Prior to 12.0 was:-
          %    ENTRY_IN_USE             BOOLEAN,
          %    RSDNT_USERCOUNT          BIT(7),
        DATA_SPACE_SIZE                BIT(24),
          % Size of the interpreter's work space (if any), to be
          % attached to the environment.
        PRE_INIT_DATA_SIZE             BIT(16),
          % Size of the work space to be initialised from the
          % interpreter code file (only valid when DATA_SPACE_SIZE > 0).
        INIT_DATA_OFFSET               BIT(16),
          % Offset of the pre-initialised work space in the interpreter
          % code file (only valid when PRE_INIT_DATA_SIZE > 0).
        GLOBAL_SEG                     SYSTEM_DESCR,
          % System descriptor for the interpreter's global (or main)
          % segment. The version is in hex (4 bytes) before the check
          % sum at the end of the segment.
          % This descriptor is called the GD or global descriptor.
        ENTRY_IN_USE                   BOOLEAN,
          % If TRUE, this entry is in use and all fields are valid.
          % If FALSE, this entry is free (and all fields refer to the
          % most recent occupant) or the entry is being built (and
          % field values relect our progress).
          % Note that this bit is set as the last order of busness.
        RESIDENT_USER_COUNT            BIT(15),
          % Number of environments using this interpreter that are not
          % rolled out.
        TOTAL_USER_COUNT               BIT(15),
          % Total number of environments using this interpreter.
        FILLER                         BOOLEAN,
          % To allow the following fields to stay byte aligned for
          % ease of debugging.
```

```
ARCHITECTURE_NAME                NAME,
  % The name of the architecture for this interpreter,
  % e.g. SDL2, COBOL74, RPG.
COMPILER_LEVEL                   BIT(8),
  % An integer representing the level of the S-machine for this
  % architecture. Checked against a corresponding program PPB
  % field.
ARCHITECTURE_ATTRIBUTES          BIT(80),
  % Used for checking minor S-machine changes (such as adding
  % an S-op) that are not significant enough to warrant a level
  % change.
INTERPRETER_NAME                 NAME_RECORD,
  % The name of the interpreter on disk.
[DFH_ADDR                         DISK_ADDR!
  % The disk address of the interpreter's DFH.
 AREA_ADDRESS_FOR_MMCP_GISMO_SDL2 DISK_ADDR],
  %  For MMCP, GISMO, SDL2 interpreter it is too difficult to
  %  build the DFH address. So we settle for the area address.
VERSION                          BIT(32),
  %  Contains the interpreter version. For use by ISSA.
FILLER                           BIT(36);
  % Available for future use. This filler is used to extend
  % the size of an entry to 768 bits, so that Gismo can
  % compute the offset of individual entries by two shifts and
  % an add (i.e. to require the same amount of work as 224 bits
  % required prior to 12.0.) (which is required every time a
  % task is reinstated).
```

RUN STRUCTURE NUCLEUS

As shown in figure D-1, the run structure nucleus includes variables that point to the next task in the mix, the file dictionary for the task, and the environment dictionary for the task.

A programmatic description follows.

```
%
SET rs_status_types MEMBER(15) =
   executing                  = 0,
   no_file                    = 1,
   no_user_disk               = 2,
   duplicate_library          = 3,
   duplicate_input_file       = 4,
   possible_dup               = 5,
   waiting_for_hardware       = 6,
   program_stopped            = 7,
   waiting_IO_complete        = 8,
   wtg_datacomm_msg           = 9,
   waiting_overlay            = 10,
   waiting_kbd_in             = 11,
   hdwr_not_ready             = 12,
   waiting_operator_action    = 13,
   waiting_close              = 14,
   waiting_DS_or_DP           = 15,
   no_mpf_pack                = 16,
   no_file_on_disk            = 17,
   waiting_for_locked_file    = 18,
   waiting_q_is_full          = 19,
   wait_status                = 20,
   nomem_waiting_comm_q       = 21,
   nomem_waiting_ready_q      = 22,
   wtg_port_open              = 23,
   wtg_pgm_call               = 24,
   waiting_time_comm_q        = 25,
   waiting_time_ready_q       = 26,
   waiting_receive            = 27,
   wtg_datacomm_opn           = 28,
   terminating                = 29,
   in_ready_q                 = 30,
   in_comm_q                  = 31,
   stopped_for_sort           = 32,
   wtg_dc_dsk_cmplt           = 33,
   wtg_datacomm_dsk           = 34,
   no_controller              = 35,
   no_output_pack             = 36,
   vsort_qsort_not_present    = 37,
   no_sort_input_file         = 38,
   waiting_contention         = 39,
   waiting_syncpoint          = 40,
   waiting_recovery           = 41,
   waiting_new_audit          = 42,
   waiting_sorter_IO          = 43,
   terminating_waiting_IO     = 44,
   closing_waiting_IO         = 45,
   waiting_forms              = 46,
   no_translate_file          = 47,
   mf_searching               = 48,
   no_DMS_file                = 49,
   no_DMS_dictionary          = 50,
   wtg_DMS_reorganization     = 51,
   wtg_inactive_data_base     = 52,
```

```
    no_usercode                 = 53,
    waiting_to_be_called        = 54,
    wtg_program_exit            = 55,
    wtg_called_pgm_BOJ          = 56,
    wtg_rel_area_init           = 57,
    wtg_datacomm_result         = 58,
    wtg_beginning_label         = 59,
    no_program                  = 60,
    no_host_services            = 61,
    wtg_host                    = 62,
    waiting_task_completion     = 63,
    waiting_system_lock         = 64,
    no_disk_WFL_log             = 65,
    no_DMS_accessroutines       = 66,
    waiting_server_message      = 67,
    waiting_protected_file      = 68
   ,waiting_schedule_disk       = 71%  Used when program call could not
                                  % schedule the callee, because of
                                  % lack of disk space.
   ,waiting_sort_disk           = 72%  Used when sort initiation failed
                                  % because no Q_DISK was available.
   ;
CONSTANT MAX_REASON             = TYPE_LENGTH(rs_status_types) - 1;
%
```

```
RECORD
   RS_NUCLEUS
      RS_CUR_ENV_ADDR                  BIT(24),
            %ABSOLUTE ADDRESS OF THE CURRENT ENVIRONMENT NUCLEUS
            %INCLUDED FOR EASIER AND FASTER ACCESS BY MMCP AND GISMO
      RS_CUR_ENV_INDEX                 BIT(16),
            %NUMBER OF THE ENTRY IN THE ENVIRONMENT DICTIONARY FOR
            %THE CURRENTLY ACTIVE ENVIRONMENT
      RS_ENV_DICT_SIZE                 BIT(16),
            %NUMBER OF ENTRIES IN THE ENVIRONMENT DICTIONARY FOR THIS
            %JOB. CURRENTLY WE ALWAYS ALLOCATE 'MAX_ENVIRONMENTS'
            %ENTRIES AS FOLLOWS (EXCEPT FOR THE SMCP, WHICH HAS 1):
            %   0  -  THE PRIMARY ENVIRONMENT (THE EXECUTED PROGRAM)
            %   1  -  AN ASYNCHRONOUS MCP ENVIRONMENT (NOT IN
            %                                  INITIAL IMPLEMENTATION)
            %   2  -  A DMS ENVIRONMENT
            %   3  -  A SPECIAL ENVIRONMENT (USED BY IBASIC)
            %
            %THE TABLE IS SIMPLY A LIST OF ABSOLUTE MEMORY ADDRESSES
            %IF AN ENVIRONMENT HAS NOT BEEN ALLOCATED, THE ADDRESS
            %FOR THAT ENTRY WILL BE ZERO.
      RS_ENV_DIC                       POINTER ENVIRONMENT_DICTIONARY,
      RS_FIB_DIC                       BIT(24),
            %MEMORY ADDRESS OF THE FIB DICTIONARY
      RS_SELF_ADDR                     BIT(24),
            %MAKES IT EASIER TO SEARCH FOR AN RSN AT A SPECIFIC
            %ADDRESS, AND ALSO SERVES AS A CHECK THAT A STRUCTURE
            %IS ACTUALLY AN RSN
      RS_MCP_BIT                       BOOLEAN,
            %IF TRUE, THIS IS THE SMCP'S RSN. NOTE THAT THIS BIT
            %DUPLICATES A BIT IN THE ESN. THE ESN BIT IS PRIMARILY
            %FOR THE CONVENIENCE OF THE INTERPRETER, AND THIS BIT
            %MAINLY FOR THE CONVENIENCE OF SMCP
      RS_ODT_INPUT_PRESENT             BOOLEAN,
            %INDICATES THAT AN AX WAS DONE
      RS_TRACE_BUF_ADDR                BIT(24),
            %MEMORY ADDRESS OF THE TRACE BUFFER IF THIS
            %JOB IS TRACING - SHARED BETWEEN ALL ENVIRONMENTS
      RS_TRACE_BUF_OFFSET              BIT(16),
            %CURRENT OFFSET WITHIN THE TRACE BUFFER
            %PREVENTS CONFLICT WHEN TRACE IS ACTIVELY SHARED - FUTURE
      RS_SWITCHES                      BIT(40),
            %10 4-BIT SWITCHES:    SW0-9
      RS_IPC_DICT_ADDR                 BIT(24),
            %ABSOLUTE ADDRESS OF THE IPC_DICTIONARY FOLLOWING THIS
            %RS_NUCLEUS. (FOR IPC)
      RS_IPC_DICT_SIZE                 BIT(16),
            %NUMBER OF ENTRIES IN THE IPC_DICTIONARY
      RS_CALLERS_RSN_ADDR              BIT(24),
            %RSN ADDRESS OF THIS JOBS CALLER
      RS_LAST_LIO_STATUS_SIZE          BIT(16),
            %SIZE OF LAST_LIO_STATUS MASK
      RS_LAST_LIO_STATUS_PTR           BIT(24),
            %ADDRESS OF LAST_LIO_STATUS MASK
      RS_SLAVE_BLOCKED_CNT             RECORD
            %NUMBER OF BLOCKS ON THE SLAVE SCHEDULER FOR THIS JOB
          RS_BLOCK_CNT                   BIT(6)
      RS_Q_LINK                        BIT(24),
            %POINTER TO THE NEXT JOBS RS_NUCLEUS
            %FIRST_QUEUE POINTS TO 1ST JOB; LAST JOB CONTAINS @FFFFFF@
      RS_LAST_TOP                      BOOLEAN,
            %IF SET, INDICATES THAT THIS JOB WAS THE LAST ONE
            %SCHEDULED WITHIN ITS PRIORITY CLASS
```

```
RS_Q_IDENT                          BIT(24),
      %THE QUEUE THAT THIS JOB IS CURRENTLY IN
      %     0 = READY_Q
      %     1 = S_COMM_Q
      %     3 = EXTERMINATE_Q
      %     6 = IOC_Q. Waiting for MCP to handle an I/O complete.
      %    10 = M_COMM_Q
      %    11 = WATE_Q
      %    -2 = NOT QUEUED.
      %          The program may  have a processor (i.e. be truly
      %         executing), Gismo soft I/O may be running (as a
      %         result of an interrupt whilst we were), or the
      %         SMCP may be fiddling with the RSN.
RS_NEXT_Q                           BIT(24),
      %IF THIS JOB IS IN THE WATE_Q, THE QUEUE IT SHOULD BE
      %PLACED IN WHEN IT IS CAUSED
RS_TERMINATING                      BOOLEAN,
      %THIS JOB IS TERMINATING - PREVIOUSLY WE HAD TO TEST

RS_REASON                           BIT(8),
      % USED BY MULTI-THREADING PROCESSES TO INDICATE PROGRESS
      % SO FAR. SHOULD BE CLEARED AT THE END OF THE COMMUNICATE.
      % ALSO USED DURING TERMINATE (CONDITIONED BY RS_TERMINATE)
RS_STATUS                           MEMBER OF rs_status_types,
      %GIVES THE CURRENT STATUS OF THE JOB
      % refer to the set declared immediately preceding.
RS_PRIORITY_INTEGER                 BIT(4),
      %PROCESSOR PRIORITY - 0-15 ALLOWED
RS_JOB_NUMBER_IN_DECIMAL            BIT(16),
      % E_G, JOB NUMBER 1753 WOULD BE @1753@
RS_PAUSE                            BIT(24),
      %TIME TO WAKE THIS JOB IF SLEEPING
RS_WAIT_LEN                         BIT(12),
      %LENGTH OF RS_EVENT_SPACE
RS_WAIT_LOC                         BIT(24),
      %ADDRESS OF RS_EVENT_SPACE
RS_DISABLE_INTERRUPTS               BIT(6),
      %IF THIS FIELD IS GREATER THAN 0 THEN THIS JOB MAY NOT
      %BE INTERRUPTED BY HIGH PRIORITY INTERRUPTS.
RS_USE_FLAG                         BOOLEAN,
      %IF TRUE, JOB IS CURRENTLY ACTIVE IN A
      %USE ROUTINE
RS_REPORT_EV_INX                    BOOLEAN,
      %USED BY PROCESSES THAT WISH TO HANG JOBS AND HAVE THE
      %EVENT WHICH WAKES UP THE JOB REPORTED IN THE RS
      %(USED BY M_WAIT AND COMPLEX_WAIT)
RS_STATE_LIGHT                      RECORD
    RS_VLAMP_DATA                     RECORD
      %USED BY THE LAMP CODE IN GISMO TO DISPLAY ACTIVITIES
      %BY JOB. INITIALLY, INFORMATION WILL BE SUMMARIZED FOR
      %ALL OF THE JOBS ENVIRONMENTS
        RS_VL_2FLAGS                    RECORD
            RS_VARIABLE_LAMP_CPU        RECORD
                %USED TO DISPLAY JOB CPU ACTIVITY
                RS_VL_CPU_GRP             BOOLEAN,
                RS_VL_CPU_USE             BOOLEAN
            END,
```

```
                    RS_VARIABLE_LAMP_CODE_OVLY RECORD
                        %USED TO DISPLAY JOB CODE OVERLAYS
                        RS_VL_COLAY              BOOLEAN,
                        FILLER                   BOOLEAN
                    END,
                    RS_VARIABLE_LAMP_DATA_OVLY RECORD
                        %USED TO DISPLAY JOB DATA OVERLAYS
                        RS_VL_DOLAY              BOOLEAN,
                        FILLER                   BOOLEAN
                    END
                END,
                FILLER                          BIT(10)
            END
        END,
RS_TIME_EVENT                       BOOLEAN,
RS_NULL_EVENT                       BOOLEAN,
RS_JOB_NUMBER                       BIT(16),
            %CONTAINS THE JOB NUMBER ASSIGNED TO THIS JOB. ASSIGNED
            %WHEN THE JOB IS SCHEDULED. JOB NUMBER IS USED ON ANY
            %ODT INPUT MESSAGE THAT REQUIRES JOB IDENTIFICATION.
            %BEGINS WITH 1 AND WRAPS AROUND AT 9999
RS_ABORT                            BIT(2),
            % 0 = RUNNING
            % 1 = DS OR DP-ED
            % 2 = CANCELED
            % 3 = DUE TO DEATH IN FAMILY
RS_DC_IO_COMPLETE                   BOOLEAN,
            %THIS EVENT IS CAUSED WHENEVER A DATA COMM I/O OR
            %AN INITIALIZER I/O COMES COMPLETE
RS_DATA_COMM                        BOOLEAN,
            %IF TRUE, JOB HAS DONE A DC_INITIATE_IO
RS_SORTER_FLOWING                   BOOLEAN,
            %MICR JOB WITH READER/SORTER CURRENTLY IN FLOW MODE
RS_ROLLOUT_BITS                     RECORD
    RS_TO_BE_ROLLED_OUT               BOOLEAN,
            %IF TRUE, JOB IS A CANDIDATE FOR ROLLOUT - DO IT NEXT
            %N_SECOND
    RS_NOT_A_ROLLOUT_CANDIDATE      BOOLEAN
            %IF TRUE, JOB HAS BEEN HUNG BUT CANNOT BE ROLLED OUT_
END,
RS_ROLLOUT_IN_PROCESS             BOOLEAN,
            %IF TRUE, JOB IS IN PROCESS OF BEING ROLLED OUT
RS_ROLLIN_IN_PROCESS              BOOLEAN,
            %IF TRUE, JOB IS IN PROCESS OF BEING ROLLED IN
RS_PREVENT_MOVE                     BOOLEAN,
            %IF TRUE, THIS RSN MAY NOT BE MOVED
            %NO CURRENT USES
RS_DISPLACED                        BIT(24),
            %THE DISTANCE THIS RSN HAS BEEN MOVED
RS_MEDIA                            BOOLEAN,
            %IF RESET, THEN SOME ENVIRONMENTS MAY BE ROLLED OUT
RS_INTERVENTION                     BOOLEAN,
            %SMCP NEEDS TO DO SOMETHING TO THIS JOB BEFORE
            %THE MMCP CAN HAVE IT (USUALLY ROLLIN)
```

```
RS_M_PROBLEM                        RECORD
        %REASON WHY THE MMCP OR GISMO TURNED CONTROL OF THIS
        %JOB OVER TO THE SMCP
    RS_M_PROBLEM_TYPE                RECORD    % FOR SDL
        RS_M_PROB_P1                   BIT(24)   % FOR MIL
        % 1=LIO PROBLEM (SEE PARAMETERS)
        % 3=FIB DICT NOT PRESENT
        % 5=RS_INTERVENTION SET
        % 7=DUMP COMMUNICATE SENT TO MMCP
        %10=RELATIVE FILE PROBLEMS
        %20=MMCP PAGE FAULT (PARAMETER=SEG DESC ADDR)
        %21=GISMO TERMINATE (SEE PARAMETER FOR TYPE 21)
        %22=ENVIRONMENT NOT PRESENT
        %30=INVALID COMPLEX_WAIT COMMUNICATE
        %31=NO ODT_QUEUE
        %110=Port, queue, or remote file problem (see parameter
        %    for type 110).
        %
    END,
    RS_M_PROBLEM_PARAMETER           RECORD    % FOR SDL
        RS_M_PROB_P2                   BIT(24)   % FOR MIL
        %   **** TYPE 1 ****       Logical I/O problem.
        % 1=IRRECOVERABLE EXCEPTION
        % 2=FIB NOT OPEN
        % 3=WRONG POSITION
        % 4=NEED NEW AREA
        % 5=INVALID CHARACTER ON PSEUDO READER FILE
        % 6=EOF
        % 7=AREA OUT OF BOUNDS
        % 8=DISK FILE HEADER INDICATES A MULTI PACK FILE
        % 9=AREA NOT PRESENT
        %10=LOGICAL I/O ALLOWED ONLY FROM SMCP
        %11=DISK FILE HEADER NOT PRESENT
        %12=INVALID FILE ACCESS
        %13=VARIABLE RECD SIZE BELOW BOUNDS
        %14=VARIABLE RECD SIZE ABOVE BOUNDS
        %15=VARIABLE RECD SIZE INVALID ON INPUT
        %16=USER DATA OUTSIDE BASE-LIMIT
        %17=EMULATOR TAPE IRRECOVERABLE EXCEPTION
        %18=EMULATOR TAPE ILLEGAL INITIATE
        %19=EMULATOR TAPE ILLEGAL FETCH
        %20=EMULATOR TAPE OVERLAP
        %21=EMULATOR TAPE ILLEGAL OPCODE
        %22=EMULATOR TAPE ILLEGAL ERROR MASK
        %23=EMULATOR TAPE ILLEGAL ACCESS
        %24=Record size invalid on variable length rewrite.
        %25=Linage page overflow.
        %26=Invalid communicate on printer file. (Linage
        %   communicate error.)
        %28=CONSECUTIVE REWRITE ERROR (Rewrite must be preceeded
        %   by a read.)
        %29=EXCEEDED MAXCARDS LIMIT
        %30=EXCEEDED MAXLINES LIMIT
        %31=Initiate serial protection.
        %
        %   **** Type 10 ****     Relative file problem.
        %   ** and relative communicate **
        %71=Initiate SYSTEM/FILE-INIT to initialise blocks of an
        %   area.
        %72=Next area required.
        %73=Invalid communicate for file.
        %74=End of file/page.
        %75=Boundary violation.
        %76=Invalid key.
```

```
            %77=Duplicate key.
            %78=Boundary violation.
            %else as for type 1.
            %
            %   **** IS communicate ****
            % 1=Memory buffer needed.
            % 2=Split fine table.
            % 3=Next area needed for data file.
            % 4=Update coarse tables.
            % 5=Make structure present.
            % 6=Create current.
            % 7=IS audit writes.
            %
            %50=Invalid key on IS start.
            %51=Duplicate key.
            %52=Invalid key.
            %53=End of file/page.
            %54=Sequence error.
            %55=Duplicate key ok????????????????????????
            %56=Integrity error.
            %
            % Fatal errors.
            %91=Invalid access mode.
            %92=Read on output only file.
            %93=Rewrite on non I/O file.
            %94=Write to non output file.
            %95=Delete on non I/O file.
            %96=Rewrite not preceeded by read.
            %97=Delete not preceeded by read.
            %98=Invalid communicate.
            %99=Irrecoverable write error.
            %100=Sequential read after dynamic invalid key.
            %
            %   **** TYPE 21 ****     Gismo Terminate.
            % 1=CWG FROM NON-MCP
            % 2=NON-MMCP CALLED CHECK.IOD
            % 3=COMMUNICATE OUTSIDE BASE/LIMIT
            % 4=INVALID PORT IN DIRECT IO DESCRIPTOR
            % 5=EXCEEDED DATA OVERLAY DISK SIZE
            %
            %   **** TYPE 22 ****     Environment not present.
            % USE VALUE FROM COMM SPLITTER MASK CORRESPONDING
            % TO ENVIRONMENT TYPE
            %
            %   **** TYPE 110 ****    Port, queue or remote problem.
            % 1=Invalid key.
            % 2=No end of file provision.
            % 3=Number of stations declared in FIB exceeded.
            % 4=No provision for terminate error.
    END
END,
RS_ODT_Q_KEY                    BIT(24),
            %POINTS AT THE QUEUE DESCRIPTOR DESCRIBING THE USERS
            %ACCEPT QUEUE
RS_FILE                         BIT(8),
            %IF THE JOB IS HUNG FOR ANY PROBLEM WITH A FILE, THIS
            %CONTAINS THE INDEX INTO THE FIB DICTIONARY FOR THE
            %FILE IN QUESTION
            %IF THE JOB IS HUNG FOR NO_DMS_FILE, then the RS_DMS_FILE
            % field at the end of rsn (new for 12.0) is used to
            % describe which file is needed. Previously, this field
            % was used to index into the structure dictionary
RS_RUN_UNIT                     BIT(16),
            %JOB NUMBER OF THE PARENT OF THIS RUN UNIT (FOR IPC)
```

```
RS_RUN_UNIT_LINK                BIT(16),
          %JOB NUMBER OF THIS JOBS CALLER (FOR IPC)
RS_IPC_PARAMETER_LIST           BIT(24),
          %ABSOLUTE ADDRESS OF THE IPC_PARAMETER_LIST
RS_EXECUTE_TYPE                 BIT(4),
          % 1 = EXECUTE
          % 2 = COMPILE AND GO
          % 3 = COMPILE FOR SYNTAX
          % 4 = COMPILE TO LIBRARY
          % 5 = COMPILE AND SAVE
          % 6 = GO PART OF COMPILE AND GO
          % 7 = GO PART OF COMPILE AND SAVE
          % 8 = CALLED (IPC)
RS_NAME                         NAME_RECORD,
          %NAME OF THIS JOB
RS_IPC_EVENT                    BOOLEAN,
          %DUMMY EVENT FOR ANY HANG FOR IPC
RS_CANCELED                     BOOLEAN,
          %A CANCEL COMMUNICATE HAS BEEN ISSUED AGAINST THIS JOB
RS_EVENT_SPACE                  BIT(24*17),
          %REPRESENTS THE LIST OF EVENTS ON WHICH A JOB IN THE
          %WAIT_Q IS WAITING
RS_FREEZE_BITS                  RECORD
    RS_BOJ_TO_EOJ_FREEZE          BOOLEAN,
          %IF TRUE, JOB WAS EXECUTED WITH FREEZE AND CAN NEVER BE
          %ROLLED OUT
    RS_SORTER_OPEN                BOOLEAN,
          %TRUE IF THIS PROGRAM HAS A READER-SORTER OPEN. THE
          %READER-SORTER MAY OR MAY NOT BE FLOWING.
    RS_TEMPORARY_FREEZE           BIT(7)
          %COUNTER THAT IS BUMPED EACH TIME A FREEZE IS DONE AND
          %DECREMENTED FOR EACH UNFREEZE. CHANGED BY FREEZE
          %COMMUNICATE, REMOTE FILE OPEN, INITIALIZER I/O, MICR
          %OPEN AND CLOSE
END,
 RS_LOG_PTR                      DISK_ADDR,
          %DISK ADDRESS OF WORKING PPB AND FPB-S
RS_JOB_ACCTING_NO               BIT(24),
          %A UNIQUE ID NUMBER FOR EACH JOB. RESET ONLY BY
          %COLDSTART. INCREMENTED BY 1 EACH TIME A JOB ENTERS THE
          %SCHEDULE. USED BY TABS.
RS_NUMBER_FILES                 BIT(8),
          %MAXIMUM NUMBER OF FPB-S DECLARED BY THIS PROGRAM.
RS_TYPE                         HDWR_TYPE,
          %HARDWARE TYPE REQUIRED TO RESOLVE MISSING HARDWARE
RS_TRACE_FIB                    BIT(8),
          %FILE NUMBER USED FOR TRACE. INDEX INTO THE
          %FIB_DICTIONARY
RS_SER_NO                       BIT(24),
          %SERIAL NUMBER OF A DISK PACK IF THIS JOB IS WAITING
          %FOR A BASE OR CONTINUATION PACK FOR MULTI PACK FILES
RS_UNIT_INDEX                   BIT(24),
          %ADDRESS OF IOAT OF DEVICE INDICATED BY IL,OU,FM,UL
RS_MCP_USE                      BOOLEAN,
          %IF TRUE, MCP IS WAITING FOR AN EVENT FLAGGED BY
          %RS_BOOLEANS TO OCCUR
```

```
RS_BOOLEANS                         RECORD
          %USED BY THE SMCP TO INDICATE ACTIONS AVAILABLE TO
          %SOLVE OPEN AND CLOSE PROBLEMS
    RS_IL                            BOOLEAN,
    RS_UL                            BOOLEAN,
    RS_OF                            BOOLEAN,
    RS_FR                            BOOLEAN,
    RS_FM                            BOOLEAN,
    RS_OU                            BOOLEAN,
    RS_OK                            BOOLEAN,
    RS_RM                            BOOLEAN,
    RS_MR                            BOOLEAN,
    FILLER                           BIT(15)
END,
RS_MEMORY_PRIORITY                  BIT(4),
          %MEMORY PRIORITY - 0-15 ALLOWED
RS_SWEEPS_BEFORE_DECAY              BIT(10),
          %NUMBER OF MEM_SWEEP_INTERVALS BEFORE IMPORTANT CODE
          %SEGMENTS WILL DECAY
RS_FORCED_SUSPENSION                BOOLEAN,
          %IF TRUE, JOB HAS BEEN STOPPED BY AN "ST EOJ". MESSAGE
RS_LENGTH                           BIT(24),
          %LENGTH IN BITS OF THIS RUN STRUCTURE
          %INCLUDES RS_NUCLEUS, FIB_DICT,
          %IPC_DICT, IPC_PARAMETER_TABLE,
          %OVERLAY_DESCRIPTOR
RS_PROTECTED                        BOOLEAN,
          %IF TRUE, JOB IS LOCKED -     NEED LP- TO DS
RS_TO_BE_STOPPED                    BOOLEAN,
          %IF TRUE, AN ST WAS ISSUED ON THIS JOB. IT IS TO BE
          %STOPPED WHEN CONVENIENT
RS_STOPPED                          BOOLEAN,
          %IF TRUE, JOB HAS BEEN STOPPED BY ST
RS_EMULATOR_TAPE                    BIT(8),
          %NUMBER OF EMULATOR TAPE FILES CURRENTLY OPEN
RS_GISMO_PROC_LOCK                  BOOLEAN,
          %FORMERLY RS_PRIVILIGED. NOW IN A RIB.
RS_APPARITION                       BOOLEAN,
          %IF TRUE, THIS JOB HAS CALLED ANOTHER JOB VIA SORT OR
          %PGM_CALLER AND IS WAITING FOR ITS COMPLETION
FILLER                              BIT(67),
          %FORMERLY RJE INFORMATION WHICH NOW RESIDES IN A RIB.
RS_PRIOR_JOB_NO                     BIT(16),
          %JOB NUMBER OF JOB THAT   INVOKED THIS JOB THROUGH
          %PGM_CALLER OR SORT
RS_OVLY_DESC_PTR                    BIT(24),
          %ADDRESS OF RESULT DESCRIPTOR OF OVERLAY DESCRIPTOR
RS_PSEUDO_READER                    BIT(24),
          %ADDRESS OF PSEUDO READER ASSIGNED TO THIS JOB
RS_DUMMY_EV                         BOOLEAN,
          %A GENERAL PURPOSE EVENT USED BY COOPERATING PROCESSES
          %WITHIN THE SMCP TO HANG A JOB AND CAUSE IT TO BE MOVED
          %TO THE SMCP-S COMM_QUEUE
RS_MAX_TIME                         BIT(24),
          % IF <> 0 THEN PROCESSOR TIME IN 10TH OF SECONDS THAT
          % THIS JOB IS ALLOWED TO RUN. ONLY APPLIES TO PRIMARY
          % ENVIRONMENT
RS_IN_TRANSACTION                   BOOLEAN,
          %JOB IS IN DMS TRANSACTION STATE
RS_DM_OPERATION                     BOOLEAN,
          %JOB HAS A DMS OPERATION IN PROCESS -
          %CANNOT BE ROLLED OUT
FILLER                              BIT(24),
```

```
RS_DMS_GLOBALS                  BIT(24),
          %ADDRESS OF DMS GLOBAL SPACE
RS_MFID_CHANGED                 BIT(2),
          %MUST SHIFT NAME LEFT ONE NAME BECAUSE OF USERCODE
RS_PKID_CHANGED                 BOOLEAN,
          %MUST DELETE THE PACK ID IN THE NAME
RS_IIO_IN_PROCESS               BOOLEAN,
          %INDICATES INITIALIZER I/O IS IN PROCESS
RS_MCS_FILE_NUMBER              RECORD
    RS_MCS_FL                     BIT(8)  % JUST A TEMP NAME
          %MCS FILE NUMBER FOR COBOL74 PARTICIPATING OUTPUT.
          %IS A COBOL 74 PROGRAM DOING DATA COMM
END,
RS_TRACE_TO_BE_STOPPED          BOOLEAN,
          %ONE MEANS TRACE FILE WILL BE CLOSED NEXT TIME INTERP
          %DOES A WRITE TO THE TRACE FILE
RS_CHARGE_NUMBER                BIT(24),
          %THIS JOBS CHARGE NUMBER
RS_pseudo_runner                BOOLEAN,
          % job to be made very transparent- no BOJ, EOJ, etc.
FILLER                          BIT(3),
          %FORMERLY RS_RESTRICTIONS. NOW IN A RIB.
RS_BNA_ZIP                      BIT(1),
          %PROGRAM ZIPPED AN "AT" CONTROL CARD COMMAND
RS_JOB_TASK_MIXNUMBER           BIT(16),
          %MIX NUMBER OF THE JOB TASK
RS_PARENT_TASK_NUMBER           BIT(24),
          %TASK NUMBER OF THE PARENT TASK
RS_PRIVATE_TASK                 BOOLEAN,
          %NO INQUIRIES MAY BE MADE OF THE TASK OR FILE ATTRIBUTES
RS_TASK_TYPE                    BIT(4),
          %TYPE OF TASK EXECUTION: 1 = ASYNCHRONOUS
          %                        2 = SYNCHRONOUS
          %                        3 = INDEPENDENT
          %                        4 = JOB
RS_TASK_NUMBER                  BIT(24),
          %TASK VARIABLE NUMBER FOR THIS TASK
RS_OBJECT_NUMBER                BIT(8),
          %OBJECT TASK NUMBER FOR INDEPENDENT TASKING ONLY
RS_TASK_VARIABLE_TABLE_ADDR     BIT(24),
          %MEMORY ADDRESS OF TABLE OF ALLOCATED TASK VARIABLES
RS_WFL_TASK                     BOOLEAN,
          %INDICATES THIS TASK IS WITHIN A WFL JOB
RS_TASK_VARIABLE_DISK_ADDRESS   BIT(36),
          %DISK ADDRESS OF THIS TASK'S TASK VARIABLE
RS_JOBSUMMARY_HDR_OFFSET        RECORD
          %OFFSET INTO HDR DICT FOR JOBSUMMARY FILE - WFL JOB ONLY
    RS_TASK_SCRATCHPAD            BIT(24)
          %MISC USE FOR TASKING
END,
RS_ENFORCE_MAXCARDS             BIT(1),
          %ENFORCE THE LIMIT ON MAXCARDS
RS_ENFORCE_MAXLINES             BIT(1),
          %ENFORCE THE LIMIT ON MAXLINES
RS_CURRENT_CARDS_PUNCHED        BIT(24),
          %CURRENT COUNT OF NUMBER OF CARDS PUNCHED
RS_MAXCARDS                     BIT(24),
          %MAXIMUM NUMBER OF CARDS TO PUNCH
RS_CURRENT_LINES_PRINTED        BIT(24),
          %CURRENT COUNT OF NUMBER OF LINES PRINTED
RS_MAXLINES                     BIT(24),
          %MAXIMUM NUMBER OF LINES TO PRINT
RS_MAX_ELAPSED_TIME             BIT(24),
          %MAXIMUM ELAPSED TIME ALLOWED
```

```
    RS_SCHED_DATE                   BIT(36),
            %YEAR MONTH DAY HOUR MINUTE SECOND WHEN SCHEDULED
    RS_INVISIBLE                    BIT(1),
            %INVISIBLE IN MIX
    RS_GO_EVENT                     BIT(1),
            %SPECIAL EVENT FOR "ST" & "GO"
    RS_ST_EOJ_SPECIFIC              BIT(16),
            % ALLOWS US TO STOP JOB UNTIL EOJ OF A SPECIFIC OTHER JOB.
    RS_DD_SIZE                      BIT(24),
            % NUMBER OF DATA DECKS PRESENT FOR TASK
    RS_DD_ADDR                      BIT(24),
            % DATA ADDRESS OF THE DATA DECK NAMES
    RS_MASTER_BLOCKED_CNT           BIT(6),
            % number of blocks on the master scheduler for this job
    RS_PORT_EVENT                   BOOLEAN,
            % HANG EVENT FOR "BNALIO_OPEN". FORMERLY A BIT IN RMSGP
    RS_DMS_TIME                     BIT(20),
            % ACCUM DMS TIME EXCEPT FOR CURRENT ENV. (FOR DMS LOGS)
    RS_DMS_FILE                     RECORD
            % THIS FIELD REPLACES THE USE OF RS_FILE TO COMMUNICATE
            % WHICH DMS_RELATED FILE IS THE OBJECT OF A
            % HANG NO FILE (WE DID IT WRONG ANYWAY). ONLY ONE OF
            % THE THREE FIELDS SHOULD BE IN USE AT A TIME
        RS_DMS_AUDIT_FILE           BOOLEAN,
            % THE MISSING FILE IS THE AUDIT FILE
            %  (OVERRIDES RS_DMS_STRUCTURE_FILE)
        RS_DMS_DICTIONARY_FILE      BOOLEAN,
            % THE MISSING FILE IS THE DICTIONARY
            %  (OVERRIDES EITHER OF THE OTHER FIELDS)
        RS_DMS_STRUCTURE_FILE       STR_PTR     %bit 10 in 12.0
    END,
            %  (ONLY RELEVANT IF OTHER TWO FIELDS ARE FALSE)
    RS_LOG_MIX_PPB_OFFSET           BIT(16),
            % THE OFFSET OF THE PPB COPY IN THE LOG MIX INFO TABLE.
            % ONLY VALID IF THE LOG OPTION IS SET.

    RS_ORIGINAL_RIB                 ROUTING_INFORMATION_BLOCK,
            % THE ORIGINAL ROUTING INFORMATION FOR THE JOB.
    RS_CURRENT_RIB                  POINTER ROUTING_INFORMATION_BLOCK,
            % THE CURRENT ROUTING INFORMATION OF THE JOB.
    RS_SERVER_QUEUE_EVENT           BOOLEAN;
            % COMPLEX WAIT EVENT TYPE 10 FOR RIB SERVER QUEUES

%
CONSTANT  RS_N_SIZE   =   TYPE_LENGTH(RS_NUCLEUS);
```

ENVIRONMENT STRUCTURE NUCLEUS

The environment structure nucleus (ESN) permits special system code files to be attached to a task.

The Environment Structure Nucleus as well as the run structure nucleus (RSN) of a program are generated by the SMCP at the beginning of a job. The RSN contains pointers to the file dictionary of the job and environment dictionary. The ESN contains pointers to the code dictionary and data dictionary of the job.

Neither the RSN nor the ESN is ever rolled out of memory. The RSN is never moved. The ESN is contiguous with the local data and is moved during rollout.

A programmatic description of the ESN follows.

```
%
CONSTANT ES_N_SIZE = 905;

RECORD
  1 ES_NUCLEUS                          BIT(ES_N_SIZE),
    2 ES_COMMUNICATE_MSG_PTR            BIT(48),
          %CONTAINS EITHER AN SDL DESCRIPTOR THAT POINTS TO A
          %COMMUNICATE MESSAGE OR THE MESSAGE ITSELF
      3 ES_MSG_PARAMETERS               BIT(24),
        4 ES_COMM_MSG_KEY               BIT(8),
          5 ES_ITYPE                    BIT(2),
            %DEFINES THE USE OF COMMUNICATE_MSG_PTR
            %    00 = PROGRAM INTERNAL INTERRUPT
            %    01 = COMMUNICATE
            %    10 = UNDEFINED
            %    11 = TERMINATING
          5 ES_INMBR                    BIT(6),
            %INTERRUPT NUMBER IF ES_ITYPE=00
        4 ES_ILENGTH                      BIT(16),
            %LENGTH OF COMMUNICATE MESSAGE IF ES_ITYPE=01
      3 ES_IADDRESS                     BIT(24),
            %ADDRESS OF COMMUNICATE MESSAGE IF ES_ITYPE = 1
            %IF ES_ITYPE = 3 THEN MAY CONTAIN ERROR COUNT OR TYPE
            %SEE IH FOR DEFINITION WHEN ES_ITYPE = 0
    2 ES_REINSTATE_MSG_PTR              BIT(48),
            %SELF-RELATIVE SDL TYPE DESCRIPTOR USED TO PASS THE RESULT
            %OF A COMMUNICATE FROM AN MCP TO AN ENVIRONMENT
            %SEE EACH COMMUNICATE FOR DEFINITION OF VALUES
      3 ES_RMSG_P1                      BIT(24),
      3 ES_RMSG_P2                      BIT(24),
    2 ES_RSN_ADDRESS                    BIT(24),
            %ADDRESS OF THE RSN FOR THIS ENVIRONMENT
    2 ES_MY_BASE                        BIT(24),
            %BASE REGISTER FOR THIS ENVIRONMENT
    2 ES_MY_LIMIT                       BIT(24),
            %LIMIT REGISTER FOR THIS ENVIRONMENT
    2 ES_LOCAL_DATA_ADDR                BIT(24),
            %ABSOLUTE ADDRESS OF THE LOCAL DATA SPACE FOR THIS
            %ENVIRONMENT.
            %NOTE: THE LOCAL DATA SPACE MAY NOT COINCIDE WITH THE
            %BASE-LIMIT AREA
```

```
2 ES_NIP                          BIT(32),
        %PAGE, SEGMENT AND DISPLACEMENT OF THE NEXT EXECUTABLE
        %INSTRUCTION FOR THIS ENVIRONMENT
  3 ES_NIP_SG_PG                  BIT(12),
        % SDL - (ES_PAGED_DICT TRUE)
    4 ES_NIP_SG                   BIT(6),
    4 ES_NIP_PG                   BIT(6),
  3 ES_NIP_SEG REMAPS ES_NIP_SG_PG BIT(10),
        % NON_SDL
  3 ES_NIP_DISP                   BIT(20),
2 ES_SEG_DIC_PTR                  BIT(24),
        %ADDRESS OF THE MASTER CODE SEGMENT DICTIONARY FOR THIS
        %ENVIRONMENT
2 ES_DATA_DIC_ADDR                BIT(24),
        %ADDRESS OF THE DATA SEGMENT DICTIONARY
2 ES_SPAD_PTR                     BIT(24),
        %ADDRESS OF SCRATCH PAD IN S-MEMORY
2 ES_INTERP_ID                    BIT(5),
        %INTERPRETER NUMBER FOR THIS ENVIRONMENT
        %INDEX INTO THE INTERPRETER DICTIONARY
2 ES_BOOLEANS                     BIT(23),
  3 ES_CONTROL_STATE_BITS         BIT(2),
        % THESE TWO MUST REMAIN IN THIS ORDER FOR GISMO'S BENEFIT
    4 ES_MCP_BIT                  BOOLEAN,
        %INDICATES TO THE INTERPRETER THAT THIS IS THE SMCP
    4 ES_CONTROL_STATE            BOOLEAN,
        %INDICATES TO THE INTERPRETER THAT THIS ENVIRONMENT
        %BELONGS TO A CONTROL STATE JOB
  3 ES_MEDIA                      BOOLEAN,
        % IF SET, THEN THE LOCAL DATA AREA IS PRESENT,
        % OTHERWISE IT IS ROLLED OUT TO DISK
  3 ES_LINKS                      BOOLEAN,
        %IF TRUE, DYNAMIC SPACE CONTAINS MEMORY LINKS
  3 ES_SIZECHANGE                 BOOLEAN,
        %IF TRUE, THE SCRATCHPAD FOR THIS ENVIRONMENT IS
        %BEING CHANGED
  3 ES_SD_PTR_FLAG                BOOLEAN,
        % 0 = ES_SEG_DIC_PTR CONTAINS ADDRESS OF DICTIONARY
        %     CONTAINER
        % 1 = ES_SEG_DIC_PTR CONTAINS ADDRESS OF SEGMENT
        %     DICTIONARY ITSELF
  3 ES_INTRIN_AGGR_USED           BIT(2),
        % SAME AS PROG_INTRIN_AGGREGATE IN THE PPB
  3 ES_DONT_REENTER               BOOLEAN,
        %IF TRUE, THIS ENVIRONMENT CANNOT SHARE ITS SEGMENT
        %DICTIONARY
  3 ES_PAGED_DICT                 BOOLEAN,
        %INDICATES THAT CODE SEGMENT DICTIONARY IS PAGED
  3 FILLER                        BIT(12),
        % FOR EASY ADDITION OF FLAGS
2 ES_PAGED_ARRAY_OVERLAY          BIT(6),
        %SEGMENT NUMBER OF THE SDL PAGED ARRAY HANDLER OVERLAY
        %IF REQUIRED FOR THIS ENVIRONMENT - ALWAYS PAGE 0
2 ES_LAST_ENVIRONMENT             BIT(16),
        %NUMBER OF THE ENVIRONMENT THAT CALLED THIS ONE
        %USED BY THE EXIT_ENVIRONMENT COMMUNICATE
2 ES_SPAD_SIZE                    BIT(16),
        %SIZE IN BITS OF SCRATCH PAD FOR THE M-MACHINE.
        %FOR B1700/B1800 IT WILL BE 768
2 ES_INTERP_DATA_SIZE             BIT(24),
        %LENGTH IN BITS OF INTERPRETER DATA SPACE
```

```
2 ES_INTERP_DATA_ADDR              BIT(24),
        %ABSOLUTE ADDRESS OF INTERPRETER DATA SPACE
2 ES_LOCAL_DATA_SIZE               BIT(24),
        %LENGTH IN BITS OF THE LOCAL DATA SPACE FOR THIS
        %ENVIRONMENT
2 ES_TRACE_BITS                    BIT(8),
        %FLAGS INDICATING WHAT TYPE OF TRACE IS TO BE PERFORMED
        %THE TRACE BUFFER IS GLOBAL TO THE PROCESS, BUT THE TRACE
        %BITS ARE LOCAL TO EACH ENVIRONMENT.
        %INITIALLY IT WILL ONLY BE POSSIBLE TO SPECIFY TRACE
        %BITS FOR THE PRIMARY ENVIRONMENT, EXCEPT VIA MODIFY
        % ON THE CODE FILE, OR PROGRAMMATICALLY
2 FILLER                           BIT(44).
2 ES_DATA_DIC_SIZE                 BIT(16),
        %NUMBER OF DATA DICTIONARY ENTRIES
2 ES_PROG_PTR                      DISK_ADDR,
        %DISK ADDRESS OF THE CODE FILE FOR THIS ENVIRONMENT
        %MAY BE ZERO FOR SPECIAL ENVIRONMENTS SUCH AS IBASIC
2 ES_DISK                          DISK_ADDR,
        %ROLLOUT DISK ADDRESS FOR THE LOCAL DATA
2 ES_LENGTH                        BIT(24),
        %LENGTH IN BITS OF THIS ENVIRONMENT
        %INCLUDES LOCAL DATA, ES_NUCLEUS, DATA_DICT,
        %SCRATCH_PAD
2 FILLER                           BIT(40),
2 ES_DATA_OVERLAYS                 BIT(24),
        %ADDRESS OF DYNAMIC SPACE WITHIN LOCAL DATA SPACE
        %FIRST LINK FOR MEMORY MGMT OF DATA OVERLAYS
2 ES_LAST_LINK                     BIT(24),
        %ADDRESS OF LAST MEMORY LINK WITHIN DYNAMIC SPACE
        %USED FOR MEMORY MGMT
2 ES_LAST_OVLY                     BIT(24),
        %LEFT OFF POINTER FOR MEMORY MGMT - USED ONLY FOR
        %ENVIRONMENTS WITH DATA OVERLAYS AND LINKS
2 ES_OVLY_DISK_BASE                DISK_ADDR,
        %DISK ADDRESS OF BEGINNING OF DATA OVERLAY SPACE
2 ES_OVLY_DISK_PTR                 BIT(24),
        %INDEX INTO THE DATA OVERLAY AREA ON DISK
2 ES_OVLY_DISK_SIZE                BIT(24),
        %NUMBER OF DISK SEGMENTS RESERVED FOR DATA OVERLAYS
2 ES_PREVENT_MOVE                  BOOLEAN,
        % IF TRUE, THIS ESN MAY NOT BE MOVED.
        % THERE ARE CURRENTLY NO CONDITIONS UNDER WHICH THIS
        % WILL BE SET. IT IS PURELY A 'FUTURES' FIELD.
2 ES_DISPLACED                     BIT(24),
        %DISTANCE THE ESN HAS BEEN MOVED AS A RESULT OF ROLLOUT
2 ES_EMULATOR_BITS                 BIT(4),
        %USED BY THE B1700 EMULATOR
2 ES_ENVIRONMENT_TYPE              BIT(4),
        % 0 = PRIMARY ENVIRONMENT (THE USER PROGRAM) - 1 ONLY
        % 1 = MCP ENVIRONMENT - FUTURE?
        % 2 = DMS ENVIRONMENT - CURRENTLY 1 ONLY
        % 3 = SPECIAL ENVIRONMENT - CURRENTLY 1 ONLY, FOR IBASIC
2 ES_TEMP_PPB_PTR                  DISK_ADDR,
        % DISK ADDRESS OF THE WORKING PPB FOR THIS ENVIRONMENT.
        % WILL BE THE SAME AS RSN.RS_LOG_PTR FOR THE PRIMARY
        % ENVIRONMENT.
```

```
2 ES_INTERFACE_ADDR                BIT(24),
        % REL ADDRESS OF DMS INTERFACE AREA WITHIN DMCP'S LOCAL
        % DATA. ALSO USED AS AN ABSOLUTE ADDRESS OF TEMP INTERFACE
        % DURING OPEN, BEFORE DMS ENV IS ALLOCATED OR BEFORE
        % ES_CONTROL_STATE IS SET -- WAS RS_DMS_INTERFACE_ADDR.
2 ES_FATALERROR                    BOOLEAN,
        % INDICATES DMCP HAS A FATAL OR DEBUG ERROR --
        %                              WAS DI_FATALERROR.
2 ES_NON_FATALERROR                BOOLEAN,
        % DMCP HAS A NON-FATAL ERROR
2 ES_RESOURCE_COUNT                BIT(16);
        % NUMBER OF RESOURCES DMCP HAS LOCKED. IF 0, PROGRAM CAN
        % BE DS'ED IMMEDIATELY, OTHERWISE RS_ABORT MUST BE SET --
        %                              WAS DI_DONT_DS_ME.
```

APPENDIX E
MEMORY MANAGEMENT

Figure E-1 shows global memory allocation and figure E-2 shows linked memory allocation. The paragraphs that follow the figures provide (1) a description of the fence within linked memory and (2) a discussion of the manner in which the SMCP program attempts to minimize memory checkerboarding.

Programmatic descriptions of a system descriptor (used in code and data dictionaries) and of a memory link (describes a memory segment) complete the appendix.

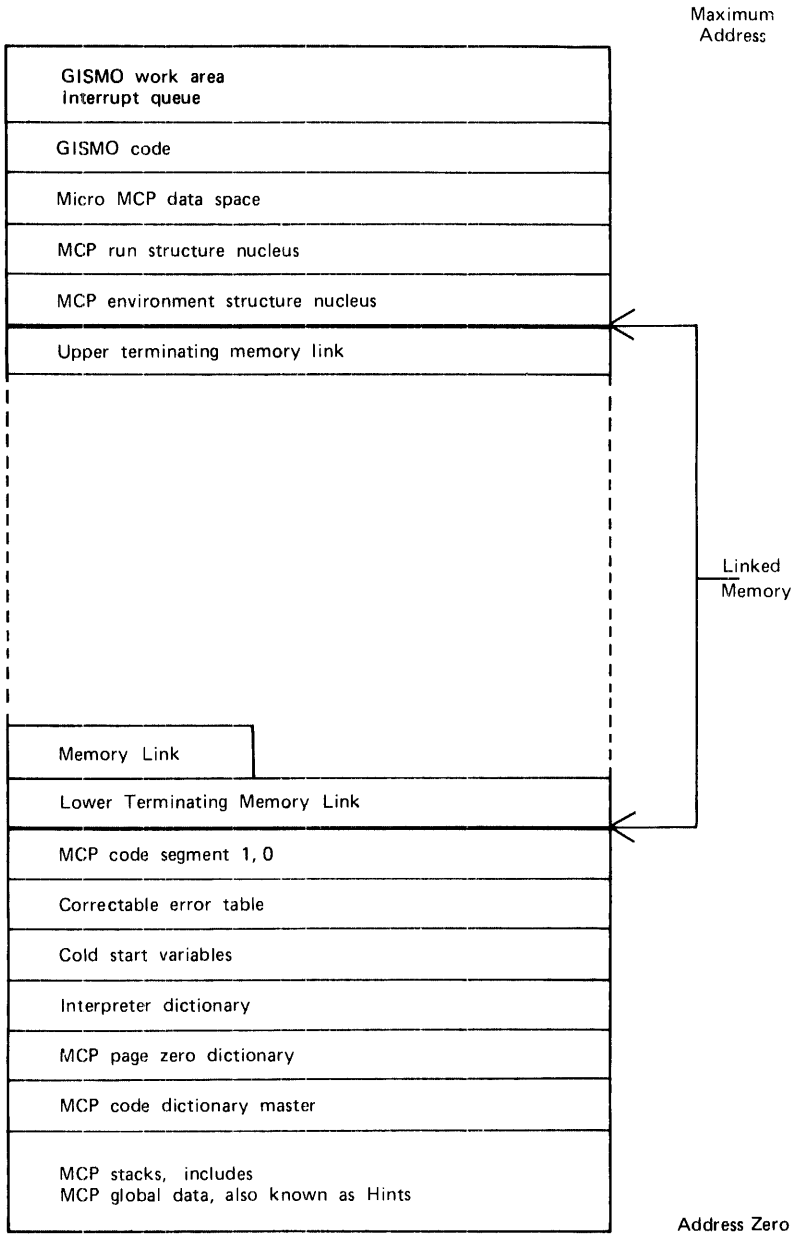


Figure E-1. Global Memory Allocation

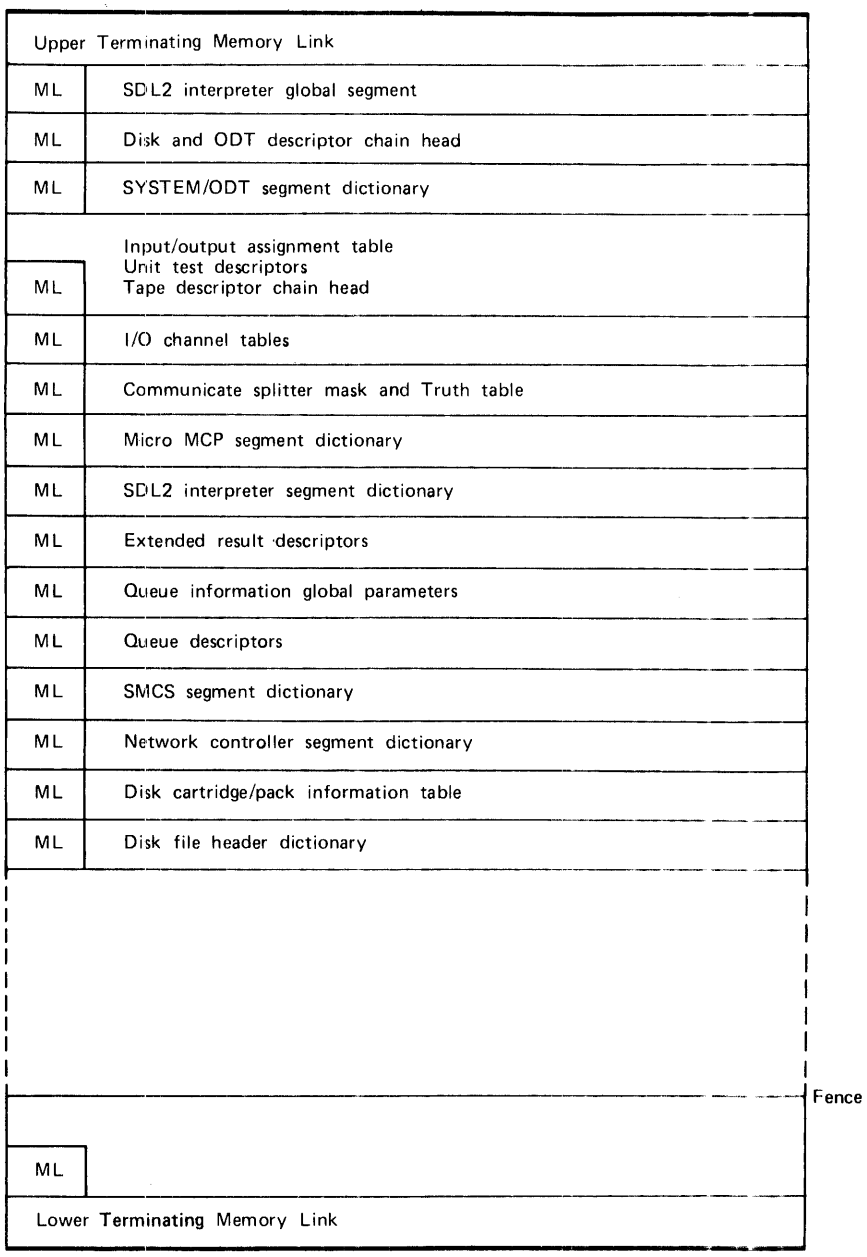


Figure E-2. Typical Linked Memory Allocation

THE FENCE

Within linked memory, the location called the fence is used to guarantee that there is always room in memory for the largest SMCP code segment and segment dictionary for that page. If the SMCP program were performing an operation and did not have room to bring in a required code segment and the segment dictionary, the system would have to halt.

Simply using the largest SMCP code segment is not good enough because a shorter SMCP code segment could have a longer dictionary; thus the combination of the two could be longer.

To calculate the location of the fence, add the bit length of the largest MCP code segment for its page, the bit length of the segment dictionary for its page, and the bit length of three memory links to the memory address of the lower terminating memory link.

MINIMIZATION OF CHECKERBOARDING

Checkerboarding, also known as "external fragmentation," is a condition in which the disk contains many permanently allocated save areas separated by small overlayable areas. In such a situation, there may be no contiguous overlayable area large enough to service a given request, even though the small areas aggregate to more total area than is needed. This situation has a serious impact upon system performance.

To minimize checkerboarding, the SMCP program allocates non-overlayable, or "save," memory segments at the high end of linked memory. Examples of such segments are program run structures, user files, and disk file headers.

SEGMENT DICTIONARIES AND SYSTEM DESCRIPTORS

Virtual memory is supported by allowing process segmentation. By segmenting code, data, and interpreters and dynamically moving a segment into or out of memory as required, the system functions as though it has almost infinite memory capacity. The MCP manages this facility through three structures: Code segment dictionaries, data segment dictionaries, and interpreter segment dictionaries. Each dictionary consists of a string of system descriptors, each of which describes one segment, including its length, location, and status. As a segment is moved in or out of memory, its dictionary entry is updated accordingly.

At run time the MCP creates the code and data segment dictionaries from information in the program's code file. The interpreter segment dictionary is created from the interpreter code file in the same manner and is referenced by an entry in the interpreter dictionary, a structure fixed in memory at CLEAR/START time. The run structure of the program contains pointers to the code and data segment dictionaries and an index into the interpreter dictionary.

A programmatic description follows.

```
RECORD
    01 SYSTEM_DESCR          BIT(SY_SIZE),
       02 SY_IN_USE          BOOLEAN,    % TO HELP MEMORY MANAGEMENT
       02 SY_STATUS          BIT(3),
          03 SY_MEDIA           BOOLEAN,% 0=DISK, 1=S-MEMORY
          03 SY_LOCK            BOOLEAN,%
          03 SY_IN_PROCESS      BOOLEAN,% TRUE IF THERE IS AN I/O IN
                                        % PROCESS FOR THE INFORMATION
                                        % REPRESENTED BY THIS DESCRIPTO
                                        % IF TRUE, "SY.CORE" CONTAINS A
                                        % POINTER TO THE I/O DESCRIPTOR
       02 SY_INITIAL         BOOLEAN,    % "ADDRESS" IS READ-ONLY MOTHER
                                        % COPY, HENCE IF "WRITE" THEN
                                        % GET NEW DISK AND REPLACE AD-
       02 SY_FILE            BOOLEAN,    % DRESS. THE OBJECT OF THIS
                                        % DESCRIPTOR IS A FILE WHOSE
                                        % USERCOUNT MUST BE DECRENENTED
                                        % WHEN THIS DESCRIPTOR IS
                                        % RETIRED.
       02 SY_DK_FACTOR       BIT(3),     % MEMORY DECAY FACTOR
       02 SY_SEG_PG          BIT(7),     % MCP MEMORY ACTIVITY AUDITING
       02 SY_TYPE            BIT(4),     % UNITS FOR SY.LENGTH.
                                        % 0 = BITS
                                        % 1 = DIGITS (4 BIT)
                                        % 2 = CHARACTERS (8 BIT)
                                        % 3 = NORMAL DESCRIPTORS
                                        % 4 = DISK SEGMENTS
                                        % 5 = SYSTEM DESCRIPTORS
                                        % 6 = SYSTEM INTRINSIC
                                        % 7 = INDIRECT REFERENCE
                                        %     ADDRESS GIVES RELATIVE
                                        %     DISPLACEMENT IN BITS
                                        %     (SIGNED NUMBER).
                                        % 8 = MICROS
       02 SY_ADDRESS         BIT(36),    %
          03 FILLER          BIT(12),    % PORT,CHANNEL AND UNIT.
          03 SY_CORE          BIT(24),    % CORE, OR ADDRESS WITHIN UNIT.
       02 SY_LENGTH          BIT(24);    % NUMBER OF UNITS, AS
                                        % DETERMINED BY SY.TYPE.
```

MEMORY LINKS

The SMCP program organizes and allocates space in memory through the use of fields known as memory links. Each link immediately precedes the block of memory it describes. The link contains such information as the disk address from which the memory was loaded, a pointer to the dictionary entry for the memory segment, the number of the job using the segment, the type of use of the segment, whether or not the segment can be overlaid, and pointers to the preceding and following memory links.

The following describes some of the different things that can happen to a memory segment beginning with the time before it is allocated and ending with the time it is deallocated:

Initially, the memory segment is available.

It is allocated.

It is swept by the GISMO firmware if the MPRI system option is set.

It is passed over for overlay.

It is overlaid (reallocated).

It is deallocated and becomes available again.

A programmatic description of a memory link follows.

```
CONSTANT MEMORY_LINK_SIZE = 185;

 RECORD
  1 MEMORY_LINK  BIT(MEMORY_LINK_SIZE),
      2 ML_DISK                  DISK_ADDR, % FROM SY_ADDRESS FIELD
      2 ML_GROUP                 BIT(47),
        3 ML_POINTER               ADDRESS, % OF DICTIONARY ENTRY
        3 ML_JOB_NUMBER            BIT(16), % OF JOB USING SEGMENT
        3 ML_TYPE                  BIT(6),  % OF MEMORY SEGMENT
        3 ML_SAVE                  BOOLEAN, % TRUE IF NOT OVERLAYABLE
      2 ML_SIZE                  BIT(24),   % SEGMENT SIZE IN BITS
      2 ML_PRIORITY_FIELD        BIT(28),
        3 ML_DK_INTERVAL           BIT(10),
        3 ML_CURRENT_DK_INT        BIT(10),
        3 ML_INCOMING_PRIORITY     BIT(4),
        3 ML_RESIDENCE_PRIORITY    BIT(4),
      2 ML_FRONT                 ADDRESS,   % OF FOLLOWING LINK
      2 ML_BACK                  ADDRESS,   % OF PRECEDING LINK
      2 ML_USAGE_BITS            BIT(2),
        3 ML_PREVIOUS_SCAN_TOUCH   BOOLEAN,
        3 ML_CURRENT_SCAN_TOUCH    BOOLEAN;
```

MEMORY LINK TYPES

```
SET ML_TYPE_SET   MEMBER (6) =
    CODE                          = 0,
    DATA                          = 1,
    AVAILABLE                     = 2,
    RN_S                          = 3,
    MCP_TEMP                      = 4,
    USER_FILE                     = 5,
    SEG_DICTV                     = 6,
    MICROCODE                     = 7,
    DICT_MASTER                   = 8,
    QUEUE_DIRECTORY_TYPE          = 9,
    MSG_BUFFERV                   = 10,
    MESSAGE_LIST_TYPE             = 11,
    I_S_BUFFER                    = 12,
    DATA_SEG                      = 13,
    DMS_BUFFER                    = 14,
    TERMINATING_LINK              = 15,
    MCP_PERM                      = 16,
    PSR_MEM                       = 17,
    MCP_IOAT                      = 18,
    DISK_HEADER                   = 19,
    PACK_MEM                      = 20,
    SD_CNTNR                      = 21,
    SCHED_MEM                     = 22,
    SORT_MEM                      = 23,
    DCH_MEM                       = 24,
    MICROCODE_NON_OVERLAYABLE     = 25,
    QUEUE_AVL_BUF_V               = 26,
    DMS_DTSK_HDR                  = 27,
    DMS_STRUCTURE                 = 28,
    DMS_TEMP                      = 29,
    DMS_GLOBALS                   = 30,
    DMS_LOCK_DESCR                = 31,
    ROUTING_INFO_BLOCK            = 32,
    PERM_ODT_BUFF                 = 33,
    DMS_WORKAREA                  = 34,
    I_S_CURRENT                   = 35,
    INTERP_DATA                   = 36,
    LOG_MIX_TABLE                 = 37,
    I_S_STRUCTURE                 = 38,
    RUN_UNIT                      = 39,
    TASK_VARIABLE_TABLE           = 40,   % 40
    SYSTEM_ERROR_TABLE            = 41;   % 41
%
CONSTANT  MAX_ML_TYPE    = 41,
          CAN_IT_GO_BELOW_FENCE = [CODE, DATA, AVAILABLE, SEG_DICTV,
```

APPENDIX F
INPUT/OUTPUT OPERATIONS

The data structures the MCP uses to keep track of input/output operations are described in the first four parts of this appendix. The ways in which various input/output functions are performed are described in the last part.

INPUT/OUTPUT ASSIGNMENT TABLE

The MCP monitors the status of all peripheral devices that are attached to the system. To do this, it maintains information about the status of each device. The major portion of this information is kept in the input/output assignment table (IOAT).

The IOAT allows the MCP to keep track of all peripheral units except the ODT and the various data communication devices. Each unit is identified by port, channel, and unit number as well as by a symbolic name. Various fields reflect the status of the unit; for example, AVAILABLE, SAVED, REWINDING, LOCKED.

Following is a programmatic description of the IOAT.

```
DEFINE IOAT_SIZE AS #528#;
 %
RECORD  1 IOAT_REC BIT(IOAT_SIZE),
    02   UNIT_INITIAL                    BIT (66), %
         03   UNIT_HDWR                  BIT (6),
         03   UNIT_PCD                   BIT (12), %
           04 UNIT_PORT_CHANNEL          BIT (7), %
             05 UNIT_PORT                BIT (3), %
             05 UNIT_CHANNEL             BIT (4), %
           04 FILLER                     BOOLEAN, %
           04 UNIT_UNIT                  BIT (4), %
         03   UNIT_NAME                  CHAR (6),
    02   UNIT_LABEL_ADDRESS              DSK_ADR,
         03   FILLER                     BIT (12),
         03   UNIT_PACK_INFO             ADDRESS,
    02   UNIT_RS                         ADDRESS,% USER LIMIT REGISTER
    02   UNIT_FLAGS                      BIT(36),
         03   UNIT_AVAILABLE             BOOLEAN,
         03   UNIT_AVAILABLE_INPUT       BOOLEAN,
         03   UNIT_AVAILABLE_OUTPUT      BOOLEAN,
         03   UNIT_WAIT_FOR_NOT_READY    BOOLEAN,
         03   UNIT_TEST_AND_WAIT         BOOLEAN,
         03   UNIT_SAVED                 BOOLEAN,
         03   UNIT_REWINDING             BOOLEAN,
         03   UNIT_EOF_SENSED            BOOLEAN,
         03   UNIT_LOCKED                BOOLEAN,
         03   UNIT_LABEL_SENSED          BOOLEAN,
         03   UNIT_LABEL_SENSED          BOOLEAN,
         03   UNIT_PRINT_BACKUP          BOOLEAN,
         03   UNIT_PURGE                 BOOLEAN,
         03   UNIT_LOCK_AT_TERM          BOOLEAN,
         03   UNIT_TO_BE_SAVED           BOOLEAN,
         03   UNIT_FLUSH                 BOOLEAN,%  FLUSH TO EOF
         03   UNIT_TAPEF                 BOOLEAN,
         03   UNIT_DISKF                 BOOLEAN,
         03   UNIT_STOPPED               BOOLEAN,
```

```
      03   UNIT_TRANSLATE              BOOLEAN,
      03   UNIT_CTRL_CARD_USING        BOOLEAN, %
      03   UNIT_REMOTE_JOB             BOOLEAN,
      03   UNIT_CLOSED                 BOOLEAN, %
      03   UNIT_CLEARED                BOOLEAN,
      03   UNIT_MULTI_FILE             BOOLEAN, %
      03   UNIT_EOT                    BOOLEAN,
      03   UNIT_TAPE_FILE_STATUS       BIT(3),% 0 = NOT RELEVANT(_ANSI)
                                               % 1 = BOV(BEG OF VOLUME
                                               % 2 = BOF(BEG OF FILE)
                                               % 3 = EOV(END OF VOLUME)
                                               % 4 = EOF(END OF FILE)
                                               % 5 = PFB(PROCESS FILE
                                               %     BLOCK)
                                               % 7 = UNDEFINED
      03   UNIT_TAPE_XCH               BOOLEAN,% FOR MIS-MATCHED UNITS
      03   UNIT_NO_TRANS_TBLE          BOOLEAN,%PC-5
      03   UNIT_OFFLINE_YET_IN_USE     BOOLEAN,%FOR ASSIGNED UNITS
      03   UNIT_AUDIT                  BOOLEAN,  % DMS AUDIT TAPE
      03   UNIT_RESERVED_BY_AB         BOOLEAN,%  AUTO BACKUP 6.1
      03   UNIT_LABEL_OP               BIT(3),% 0=@00E00X@ ODD TRANS
                                               % 1=@00C00X@ ODD NO TRANS
                                               % 2=@00600X@ EVEN TRANS
                                              % 3=@00400X@ EVEN NO TRANS
02    UNIT_DRIVE_TYPE                  BIT(4),% DISK ONLY
           %  VALUE     DCC1/2/3  DPC1/2  DFC1     DFC3
           %    0       32X203     N/A     N/A      N/A
           %    1       32X406     215    SYS.MEM   5N
           %    2       64X203     225     N/A      N/A
           %    3       64X406     N/A     1C-3     N/A
           %    4         N/A      207     1C-4     N/A
           %    5         N/A      205     1A-3     N/A
           %    6         N/A      206     1A-4     N/A
           %    7         N/A      N/A      N/A     N/A
      03 UNIT_PRINTER_TYPE            BIT(4),
           %    0       450-700 LINE PER MINUTE PRINTER.
           %    1       1100    LINE PER MINUTE PRINTER.
           %    2       NOT USED.
           %    3       1500    LINE PER MINUTE PRINTER.
           %    4-15    NOT USED.
02    UNIT_STATUS                     BIT (15),
02    UNIT_TO_BE_POWERED_OFF          BOOLEAN,
02    UNIT_PC2_TRAIN                  BOOLEAN,
02    FILLER                          BIT(6),
02    UNIT_JOB_NUMBER                 BIT(16),
02    UNIT_FIB_ADDRESS                ADDRESS,
02    UNIT_LABEL_TYPE                 BIT (2),
                %    0 = OMITTED
                %    1 = BURROUGHS
                %    2 = USASI
                %    3 = INSTALLATION
02    UNIT_TRANS_TBLE_ID              BIT(8),  %PC-5 TRAIN ID
02    UNIT_SAVED_FOR_JOB              BIT (16),
                %  IDENTIFIES WHO THIS DEVICE IS SAVED FOR
02    FILLER                          WORD,% PLEASE DO NOT DISTURB
02    UNIT_TEST_DESC                  BIT (DESCRIPTOR_SIZE);
```

CHANNEL TABLE

Another structure associated with peripheral management is the channel table, a structure for passing information between GISMO and the MCP. There is one channel table for each port. Each element of a channel table describes one channel of the port.

The channel table reflects the status of a particular channel. Certain information passed to GISMO during a dispatch operation is used by soft I/O to manage the execution of that operation. Before GISMO passes control back to the MCP, certain fields that direct the course of action the MCP will take are updated.

Following is a programmatic description of the channel table.

```
RECORD
01 CHANNEL_TABLE          BIT(48),
  02 BUSY                 BIT( 1),% SET WHEN CONTROL IS BUSY.
  02 PENDING              BIT( 1),% SET ON RECEIPT OF DISPATCH OPERATION.
  02 EXCEPTION_IDLE       BIT( 1),% SET IF EXCEPTION OCCURS AND
                                  % LINK_ON_EXCEPTION FALSE.
                                  % INHIBITS DISPATCHES.
                                  % RESET BY DISPATCH WITH OVERRIDE SET.
  02 TIMER_DISPATCH       BIT( 1),% SET BY MCP DURING CLEAR/START.
                                  % IMPLEMENTS TEST AND WAIT OPERATIONS
                                  % ON TAPE AND DISK CONTROLS.
                                  % CAUSES GISMO TO INITIATE THE CHANNEL
                                  % AT EACH TIMER INTERRUPT IF THE
                                  % CHANNEL IS NOT BUSY.
  02 EXCEPTION_OVERRIDE BIT( 1),% SET BY MCP TO CLEAR TEST & WAIT ON
                                  % ODT OR DATACOMM, TO CLEAR A HUNG CON-
                                  % TROL OR TO PROCEED AFTER AN EXCEP-
                                  % TION. IF SET AT DISPATCH,
                                  % CAUSES GISMO TO RESET BUSY, PENDING,
                                  % EXCEPTION_IDLE AND EXCEPTION_OVER-
                                  % RIDE, AND THEN TO PROCEED WITH NORMAL
                                  % DISPATCH.
  02 EXCHANGE             BIT( 1),% SET BY MCP DURING CLEAR/START.
                                  % IMPLIES CHANNEL IS ON AN EXCHANGE.
  02 OLD_MODE             BIT( 1),% NOT USED
  02 INTEGRITY            BIT( 1),% SET TO INDICATE CHANNEL TABLE ENTRY
                                  % INITIALIZED CORRECTLY.
  02 LINK_ON_EXCEPTION    BIT( 1),% SET BY MCP DURING CLEAR/START
                                  % FOR TAPE AND DISK ONLY.
                                  % CAUSES GISMO TO KEEP LINKING THRU
                                  % THE DESCRIPTOR CHAIN IF AN EXCEPTION
                                  % OCCURS RATHER THAN SETTING THE
                                  % EXCEPTION_IDLE BIT TO INHIBIT FURTHER
                                  % DISPATCHES.
  02 ODT_DISPATCH_OVERRIDE BIT(1),% IF SET AND EXCEPTION HAD OCCURRED
                                  % OVERRIDE EXCEPTION AND PROCEED WITH
                                  % DISPATCH.
                                  % IF RESET AND EXCEPTION HAD OCCURRED
                                  % INHIBIT DISPATCH AND RETURN ERROR
                                  % TO CALLER.
  02 DISK_DEVICE          BIT( 1),% NOT USED PRESENTLY
  02 FILLER               BIT( 1),%
```

```
02 TYPE                 BIT( 4),% DEVICE TYPE, ONLY FOR DUMP ANALYSIS
                                % 0 = SERIAL DEVICE
                                % 1 = DISK
                                % 2 = TAPE
                                % 3 = CASSETTE
                                % 4 = FLEXIDISK
                                % 5 = DATACOMM
02 LAST                 BIT( 1),% DELIMITS CHANNEL TABLE
02 EXCHANGE_PC          BIT( 7),% EXCHANGE PORT AND CHANNEL
  03 EXCHANGE_P         BIT( 3),% EXCHANGE PORT
  03 EXCHANGE_C         BIT( 4),% EXCHANGE CHANNEL
02 REF_ADDR             BIT(24);% ADDRESS OF DESCRIPTOR IN PROCESS
                                % EXCEPT FOR TAPE AND DISK CHANNELS
                                % WHERE IT IS ADDRESS OF HEAD OF CHAIN.
```

INPUT/OUTPUT DESCRIPTOR

The following is a programmatic description of an input/output (I/O) descriptor as seen in the disk descriptor chain and the tape descriptor chain.

```
RECORD
01 IO_DESCRIPTOR BIT(272),
  02 ACTUAL_END  M_ADDR, % POINTS TO END MEMORY ADDRESS WHEN OP
                         % COMPLETE.
  02 RESULT      BIT(24),
    03 BIT_0_1   BIT( 2),% 00 = READY TO DISPATCH FROM MCP TO GISMO
                         % 01 = INITIATED FROM GISMO TO I/O CONTROL
                         % 10 = OP COMPLETE, NO EXCEPTION
                         % 11 = OP COMPLETE, EXCEPTION
                         % DEFINITION OF THE REMAINING 22 BITS VARIES
                         % DEPENDING ON WHETHER THE OP IS COMPLETE.
                         % THE FOLLOWING DEFINITIONS APPLY WHEN
                         % THE OP IS NOT COMPLETE
    03 FILLER    BIT( 1),
    03 BIT_3     BIT( 1),% RSF.INIT
                         % IF BIT_0_1 = 0 AND BIT_3 = 1 AND BIT_6 = 1
                             THE IO DESCRIPTOR HAS BEEN INITIATED
                             THE UNIT IS SEEKING
    03 FILLER    BIT( 2),
    03 BIT_6     BIT( 1),% RSF.DISK.DEVICE
  02 LINK        M_ADDR, % POINTS TO FOLLOWING DESCRIPTOR
  02 IO_OP       BIT(24),
    03 OP        BIT( 3),% 000x = READ
                         % 010x = WRITE
                         % ETC
    03 FILLER    BIT(17),
    03 UNIT      BIT( 4),% UNIT NUMBER
  02 BEGIN       M_ADDR, % BEGINNING MEMORY ADDRESS FOR DATA TRANSFER
  02 END         M_ADDR, % ENDING    MEMORY ADDRESS FOR DATA TRANSFER
  02 DISK        BIT(24),% BEGINNING  DISK   ADDRESS FOR DATA TRANSFER
  02 M_EVENTS     BIT( 8),
    03 IOC        BIT( 1), % HARD I/O COMPLETE, PHYSICALLY
    03 SIOC       BIT( 1), % SOFT I/O COMPLETE, LOGICALLY
    03 FILLER     BIT( 1),
    03 INT_P_OR_S BIT( 1), % IF ON, CAUSE WAITING PROGRAM WHEN IOC
                           % IF OFF, QUEUE SMCP INTERRUPT WHEN IOC
    03 S_INT_SENT BIT( 1), % SMCP INTERRUPT QUEUED
    03 M_INT_SENT BIT( 1), % NOT USED
    03 FILLER     BIT( 1),
    03 INT_S      BIT( 1), % QUEUE SMCP INTERRUPT WHEN IOC
  02 MCP_IO      BIT(16), % CALLED I/O TYPE IN DUMP
  02 FIB         M_ADDR,  % ADDRESS OF FIB ASSOCIATED WITH THIS I/O
  02 FIB_LINK    M_ADDR,  % USED TO LINK MULTIPLE BUFFER FIBS
  02 BACK_LINK   M_ADDR,  % POINTS TO PREVIOUS DESCRIPTOR
  02 PORT_CHANNEL BIT( 7),
    03 PORT      BIT( 3), % PORT    TO WHICH OPERATION IS SENT
    03 CHANNEL   BIT( 4), % CHANNEL TO WHICH OPERATION IS SENT
  02 BEEN_THRU_ERROR BIT( 1); % SMCP HAS ALREADY HANDLED EXCEPTION
```

RESULT DESCRIPTORS

The following are programmatic descriptions of non-test operation result descriptors for tape and disk devices.

```
RECORD
01 TAPE_RESULT   BIT(15), %          NON-TEST OPERATIONS
  02 COMPLETE    BIT( 1), %  0
  02 EXCEPTION   BIT( 1), %  1
  02 NOT_READY   BIT( 1), %  2
  02 PARITY      BIT( 1), %  3
  02 MEM_ACCESS  BIT( 1), %  4
  02 MEM_PARITY  BIT( 1), %  5
  02 EOT         BIT( 1), %  6
  02 BOT         BIT( 1), %  7
  02 WRITE_LOCK  BIT( 1), %  8
  02 E_O_F       BIT( 1), %  9
  02 REWIND      BIT( 1), % 10
  02 BLANK_TAPE  BIT( 1), % 11
  02 CRC         BIT( 1), % 12
  02 FILLER      BIT( 2); % 13 & 14

RECORD
01 DISK_RESULT       BIT(17), %      NON TEST OPERATIONS
  02 COMPLETE        BIT( 1), %  0
  02 EXCEPTION       BIT( 1), %  1
  02 NOT_READY       BIT( 1), %  2
  02 DATA_PARITY     BIT( 1), %  3
  02 FILLER          BIT( 1), %  4
  02 MEMORY_PARITY   BIT( 1), %  5
  02 WRITE_LOCKOUT   BIT( 1), %  6
  02 FILLER          BIT( 2), %  7 & 8
  02 ADDRESS_PARITY  BIT( 1), %  9
  02 SECTOR_ADDRESS  BIT( 1), % 10
  02 SEEK_TIMEOUT    BIT( 1), % 11
  02 FILLER          BIT( 3), % 12, 13, 14
  02 TRANSMISSION    BIT( 1); % 15
```

INPUT/OUTPUT FUNCTIONS

Normal state programs request I/O functions in a symbolic fashion; for example, Write a Record. The MCP must transform these expressions into explicit I/O operators called I/O descriptors. An I/O descriptor allows the MCP to communicate directly with a peripheral device by means of the soft I/O routines of GISMO. GISMO manages the execution, by the I/O subsystem, of these operators.

ALL I/O descriptors include fields providing such information as the type of I/O operation requested, source or destination memory addresses, and the device that is to execute the operators, as well as a field for result information used when control is returned to the MCP. Certain types of I/O descriptors also contain fields for information unique to their specific functions.

Any number of I/O descriptors may be linked together to form a single chain, dispatched in one MCP operation to minimize the interaction of the MCP with the I/O subsystem.

The multiline control is the only B 1000 device control that has a direct connection with main memory. For all other controls, all data transfers between the control and memory must go through the processor. GISMO contains a set of microcoded routines with the primary function of interfacing between the MCPs (MMCP, SMCP) and the hardware. This allows the MCPs to view the I/O subsystem as an I/O processor. The MCPs can initiate I/O descriptors; GISMO handles initiation of the control, data transfer, and termination. The MCPs can queue several descriptors for execution by a control by properly setting the link fields in the descriptors; GISMO initiates each one in turn.

User programs make requests to the MMCP. Sometimes, the MMCP must ask that the request be handled by the SMCP. In either case, the request is passed to GISMO, which, in turn, passes it to the I/O control.

The I/O subsystem has the capacity for handling up to 15 controls (channels). GISMO initiates an I/O operation on a channel but does not wait for the operation to complete. It returns control to the requesting channel. Consequently, more than one I/O operation may be in process at any given time. However, GISMO addresses only one channel at a time.

The primary communication between the MCPs and GISMO is through the I/O descriptors. The SMCP dispatches I/O operations to GISMO using the DISPATCH S-operator. (The MMCP contains microcode to perform a similar function.) This S-operator requires two parameters, the port and channel of the device being addressed and the memory address of the descriptor. The I/O descriptor contains all of the information needed by GISMO for the operation.

An I/O descriptor is usually located by its Reference Address, which is the memory address of the result descriptor field of the I/O descriptor. (The result descriptor field is often referred to as the Result Status (RS) field.) All the descriptors associated with a given control are linked together in memory by setting the LINK field in one descriptor to the memory address of the RS field of the next descriptor. The descriptors are also linked in the reverse direction through the BACK_LINK field. This facilitates the adding and deleting of descriptors. A LINK field may not be zero but it may hold the memory address of the descriptor it is in.

Each RS field is 24 bits in length, and the bits have different meanings at different times. When the descriptor is ready for initiation, the RS field is formatted as shown in table F-1.

Table F-1. Result Status Field

RS Field	
Bits 0-1	RS Status Bits 00 – Ready to be executed 01 – I/O currently in process 10 – I/O complete with no exception 11 – I/O complete with exception
Bits 2-11	Gismo Toggles MCPs may not alter any bits in this field if RS Status = 01.
Bits 12-14	Port to which this I/O is directed. (Not used)
Bit 15	Interrupt requested on I/O Completion.
Bit 16	High-Priority interrupt requested on I/O Completion.
Bits 17-19	Port to which interrupts are to be sent upon I/O Completion. (Usually processor 0.)
Bits 20-23	Channel on which I/O is to be performed.

The leftmost bit (bit 0) of an RS field is always set when the operation is complete. Consequently, storing a result descriptor locks the descriptor to GISMO. The MCP may lock a descriptor as well, if the status field does not contain 01. Gismo initiates only ready descriptors, descriptors with status bits equal to 00. When the operation is initiated, GISMO sets the status bits to 01.

During an I/O operation, bits 2-11, designated GISMO Toggles, are used by GISMO to store information that it needs concerning the operation.

GISMO/Hardware Interface

The I/O descriptor contains most of the information GISMO needs to accomplish an I/O operation. In the actual hardware interface, the OP, BEGIN, END, DISK address and ACTUAL_END fields are used. The ACTUAL_END field is 24 bits in length and immediately precedes the RS field in each descriptor. The field is used by GISMO while the operation is in process to store the memory address of the data that is to be transferred to or from the memory buffer. When the operation is complete, the ACTUAL_END field contains the address of the next bit at which data would have been accessed.

Each control is able to buffer (store) a certain amount of data to be transferred. The size of the buffer depends on the device. The amount of data that may be contained in the controls and the procedures that GISMO must follow in the execution of an operation are specified when the control is designed and do not change afterward.

I/O Chaining

The I/O subsystem of the B 1000 system does not use queues for I/O operations. Using the facilities described in the preceding paragraphs, it connects all I/O descriptors that are directed to the same control or group of controls connected by an exchange into a circular chain. This eliminates the need to direct an I/O COMPLETE interrupt to the MCP as long as the requestor, usually a user program, does not produce requests faster than they can be satisfied. In other words, if the I/O subsystem is completing operations before they are actually required by the user, the user never needs to wait for the completion of an I/O request, and the MCP never needs to suspend the program waiting for such a completion.

Even when the user program is forced to wait for the completion of I/O requests, the amount of processing needed to suspend and then reinstate a program is minimized by the use of chaining. Processing is limited to that required for program execution; none is needed to tell the I/O subsystem what it should do next because that information is already contained in the I/O descriptor.

For all devices except tape and disk, the MCP constructs a circular chain of descriptors in memory. GISMO executes the requested operations in turn as each descriptor is unlocked by the MCP. Upon encountering a locked descriptor, GISMO simply stops going through that chain until the descriptor is unlocked. This occurs when the user program requires a physical I/O operation or when the file is closed for any reason. If the program must wait on an operation, an I/O COMPLETE interrupt is requested through the use of the appropriate bit in the RS field, and the program is then suspended pending the occurrence of the interrupt.

Disk I/O Chaining

The disk I/O subsystem operates somewhat differently from the operation just described. Since each disk I/O descriptor contains a disk address field, it is not necessary for the operations to execute in any particular order. Various means are provided in the software to avoid contention problems. (Similar techniques are needed in I/O subsystems that utilize queueing instead of chaining.)

All I/O descriptors for disk controls are connected in the same chain. If the system has more than one disk control, then each Channel Table entry points to the head of the chain. If GISMO encounters a descriptor that is not ready for execution or is already in process (RS field, bits 0 and 1 not equal to 00), it does not stop but continues to the next descriptor in the chain. Also, if an exception condition occurs, GISMO does not stop as it does on other controls. Both of these actions are specified by the LINK_ON_EXCEPTION bit in the Channel Table.

Since GISMO continues linking in both of the cases mentioned above, it must know when it has examined all the descriptors in the chain. At that time it must stop to free the processor for other execution. To accomplish this, a special descriptor with the IO OP field set to @840000@ is used to mark the top of the chain.

The PENDING bit in the Channel Table is set by GISMO when it receives a dispatch operation from the MCP. When GISMO links to the special descriptor denoting the top of the chain, and the PENDING bit is set, it does not stop but resets the PENDING bit and continues linking. If the PENDING bit is reset when GISMO links to the top of the chain, GISMO stops linking.

This method assures proper functioning of dispatch operations that occur in a sequence different from that of the descriptor link fields. For example, if descriptors A, B, and C are present in the chain, and B is dispatched, GISMO links to and initiates B. If, during the time that B is in process, A is dispatched, GISMO links past C to find and initiate A.

Since all descriptors for all disk controls are maintained in the same chain, GISMO must be able to recognize descriptors that are addressed to controls different from the one it is handling. This is accomplished using bits 20-23 of the RS field of the I/O descriptor. Upon encountering an unlocked I/O descriptor, GISMO compares this field to the channel it is executing upon. If the two are not equal, GISMO does not mark the descriptor in process but continues linking.

Disk I/O Overlapped Seeks

When an I/O operation is initiated on a moveable arm disk device with the arm positioned to a cylinder different from the one specified in the descriptor, the arm must be moved to the proper cylinder. This operation is called a "seek." On the B 1000 system, all seek operations are implicit; there is no explicit seek operation in the hardware. The MCPs initiate disk I/O operations without regard for the current arm position. If arm movement is required, it is accomplished by GISMO, the control, and the device without the participation of the MCP, which never knows when a seek is required or is being performed.

All seek operations are overlapped. This means that the arm of any given drive may be in motion simultaneously with the arms of other drives. Also, the control may be performing data transfer or any other operation while the arms are in motion. This is accomplished by the control returning a result descriptor with RS bit 16 = 0 to inform GISMO that some special action is necessary and that the result descriptor should not be stored. In this particular case, the control also informs GISMO that the selected drive is now seeking (RS bit 3 = 1). No further operations are initiated upon that drive until GISMO is informed by the hardware that the seek operation has completed.

All disk pack controls except the DSC on the B 1990 notify GISMO that a seek operation has completed by raising Service Request while in Status Count 1. GISMO again sends the descriptor to the control and this time, after any required latency period, data transfer occurs.

Because the DSC has an I/O descriptor buffer for each disk unit, it does not interrupt GISMO when the seek operation completes. Instead, it retains control of the I/O descriptor until it is ready for data transfer to occur.

Tape I/O Chaining

The chaining of I/O descriptors for magnetic tape controls is perhaps the most complex of the three chaining operations. The complexity is caused by the fact that tape I/O descriptors directed to each separate tape unit must be executed in logical sequence, and there may be several such units attached to the same controls. It does not matter which unit GISMO addresses next, but the descriptor that is used to address the unit must be the next logical descriptor in the "subchain" for that unit. Therefore, it is necessary to break the channel chain into subchains (one subchain for each physical unit) and to implement a means of remembering the next logical descriptor that must be used within each subchain.

Both of these requirements are satisfied by the LOCK descriptor. LOCK, a pseudo I/O operation, is handled completely by GISMO and causes no physical I/O operations. It also serves as a means of resolving contention problems between the MCPs and GISMO and between two or more tape controls that are attached to the same units by an exchange. LOCK operates as follows:

When the system is Clear/Started, the MCP constructs a tape chain with one LOCK descriptor for each unit connected to the system. The ACTUAL_END field of a Lock descriptor is not used, and the LINK field contains the memory address of the next Lock descriptor. The BEGIN and END address fields of the Lock descriptor contain the address of the TEST.AND.WAIT I/O descriptor that the MCP uses to monitor the status of each unit. This is discussed in the following paragraph.

When a file is opened on a tape unit, the MCP locks the Lock descriptor by swapping @01@ into the first two bits of the result status field. The MCP next constructs a subchain for the unit. The subchain consists of one I/O descriptor for each buffer requested by the user. The BEGIN and END addresses of the Lock descriptor are set to the memory address of the first I/O descriptor in the subchain and the TEST.AND.WAIT descriptor is removed from the subchain. The BEGIN address field is not altered until the file is closed. The END address is modified by GISMO each time it executes an operation in the subchain so that the next operation to be performed on the unit is remembered.

The LINK fields in each I/O descriptor in the subchain will address the next physical descriptor in the subchain, as they do for all other controls. An exception to this is the last physical descriptor in the subchain. The LINK field of this descriptor contains the address of the Lock descriptor for that unit. This prevents one unit from monopolizing the entire control and assures that GISMO will periodically determine if there is anything to be done on the other units.

The REF__ADDR field of the Channel Table entry for a tape chain contains the address of the special descriptor with the IO OP field set to @840000@, which marks the top of the chain. GISMO, upon receiving a dispatch for a tape control, discards the Reference Address passed and starts at the address provided by the REF__ADDR field. GISMO first attempts to lock the Lock descriptor by swapping @01@ into the first two bits of the RS field. If successful, it fetches the address in the END field of the Lock descriptor and proceeds to that address. If this descriptor is unlocked, it begins the operation specified. If not, it returns to the Lock descriptor and stores the address, which it previously fetched from the END address field back into the END address field.

Assume now that the descriptor at the address fetched from the END field of the Lock descriptor was unlocked. GISMO begins this operation and, assuming that the operation cannot be completed without some intermediate Service Requests, returns to the Lock descriptor and continues linking through the chain. Eventually, the control will raise a Service Request and reference the initiated descriptor. Upon completion of that descriptor, GISMO stores a result and fetches the LINK field of the descriptor. It then proceeds to the new descriptor and again checks to see if it is locked. If it is, GISMO returns to the Lock descriptor for the unit and stores the new address in the END address field. The new descriptor now becomes the next logical descriptor to be executed on that unit. In this manner, GISMO effectively maintains a logical sequence of operations that are to be performed on any tape unit.

There is no possibility of conflict for a unit between two or more controls connected by an exchange, since GISMO first attempts to lock the Lock descriptor before proceeding down a subchain. Similarly, the MCP must lock the subchain before altering any descriptor in the subchain.

APPENDIX G
DISK ORGANIZATION

This appendix, in eight parts, describes the following disk formats:

1. Records at the beginning of system disks.
2. Records at the beginning of user disks.
3. Pack labels.
4. Master and working available tables and temporary tables.
5. Directories.
6. File headers.
7. File dictionaries.
8. File Information Blocks.

SYSTEM DISK FORMAT

Table G-1 shows the formats of the records at the beginning of the system disk.

Table G-1. System Disk Beginning Record Formats

Address	Pointed to by	Description
0	– –	Pack label (HPT disks have no labels.)
1	– –	Disk sector relocation table
2-4	CSV	Master available table
5-6	– –	CSV (Cold start variables)
7-19	CSV	Filler (Was Log mix information (if LOG is set) prior to 12.0)
20-31	CSV	Trace FPB
32-47	CSV	Disk directory
48	– –	SYSTEM.PCU.AND.SERIAL.NUMBERS:
49-57	– –	Filler (Was the XM table.)
58-63	CSV	Temporary table
64-73	CSV	Working available table

USER DISK FORMAT

Table G-2 shows the formats of the records at the beginning of a user user disk.

Table G-2. User Disk Beginning Record Formats

Address	Pointed to by	Description
0	– –	Pack label
1	– –	Disk sector relocation table
2-31	Pack label	Master available table
32-47	Pack label	Disk directory
48-57	Pack label	Working available table
58-62	Pack label	Temporary table

PACK LABEL

All disks except head-per-track subsystems are identified by a standard American National Standard Institute (ANSI) pack label. A pack label occupies sector 0, and is non-expandable. Sector 0 contains pack identification information. sector 1 is the start of the sector relocation table.

The following is a programmatic description of a pack label.

```
CONSTANT PACK_LABEL_SIZE = 180;% BYTES.
RECORD
 1 PACK_LABEL_DECLARATION CHARACTER (PACK_LABEL_SIZE)
,   02 PL_VOL1                CHARACTER(4)   % "VOL1"
,   02 PL_SERIAL_NO           CHARACTER(6)   % SERIAL (CAN) NUMBER
,   02 PL_ACCESS_CODE         CHARACTER(1)   % ACCESS CODE
,   02 PL_ID                  CHARACTER(17)  % PACK ID
,        03 PL_NAME           CHARACTER(10)
,        03 FILLER            CHARACTER(7)
,   02 PL_SYSTEM_INTERCHANGE  CHARACTER(2)   % SYSTEM INTERCHANGE/CODE
                                             %  00 = INTERCHANGE
                                             %  17 = B1700 INTERNAL
                                             %  35 = B3500 INTERNAL
                                             %  ETC, ETC, ETC
,   02 PL_CODE                CHARACTER(1)   % PACK CODE 00 = SCRATCH
,   02 FILLER                 CHARACTER(6)
,   02 PL_OWNER_ID            CHARACTER(14)
,   02 PL_TYPE                CHARACTER(1)   % "U" = USER PACK
                                             % "S" = SYSTEM.PACK
,   02 PL_CONTINUE            CHARACTER(1)   % "C" = CONTINUATION FLAG
,   02 FILLER                 CHARACTER(26)
,   02 PL_INT                 CHARACTER(1)
,   02 PL_VOL2                CHARACTER(4)   % "VOL2"
,   02 PL_DATE_INITIALIZED    CHARACTER(5)
,   02 PL_INIT_SYSTEM         CHARACTER(6)   % INITIALIZING SYSTEM
,   02 PL_DISK_DIRECTORY      CHARACTER(8)   % DIRECTORY ADDRESS
,   02 PL_MASTER_AVAIL        CHARACTER(8)   % MASTER AVAILABLE TABLE
,   02 PL_DISK_AVAILABLE      CHARACTER(8)   % WORKING AVAILABLE TABL
,   02 PL_INTEGRITY           CHARACTER(1)   %     0 = NORMAL
                                             %     1 = RECOVERY REQUIRED
,   02 PL_ERROR_COUNT         CHARACTER(6)
,   02 PL_SECTORS_XD          CHARACTER(6)   % REMOVED SECTORS
,   02 PL_TEMP_TABLE          CHARACTER(8)   % TEMP TABLE LINK
,   02 PL_PCD                 CHARACTER(3)   % LAST PORT, CHAN, DRIVE
,   02 PL_ASSIGNED_TO_BPS     CHARACTER(6)   % BASE PACK SERIAL NUMBER
,   02 PL_SP_SEC_FLAGS        CHARACTER(8)   % SPARE.SECTOR.TABLE FOR 225
,   02 FILLER                 CHARACTER(23)
```

DISK AVAILABLE TABLES

In order to allocate disk storage, available disk space is described in the three tables described next.

Master Available Table

- Begins at disk sectors 2, 3, and 4.
- Is expandable as needed.
- Includes a list of disk segments not removed by the XD command or by disk initialization or by extensions to the relocate table (RLT).
- Consists of one table for each user drive and one for each system drive. Each system drive table is physically located on the drive it describes.

Working Available Table

- Begins at disk sectors 64 through 73 (@40@ through @49@) on the system disk.
- Begins at disk sectors 48 through 57 (@30@ through @39@) on the user disk.
- Is expandable as needed.
- Contains a list of available disk segments.
- When a user disk is purged, the contents of the working available table are replaced by the contents of the master available table.
- One table for each user drive and one for all system drives. The system drive table is on the first drive.

Temporary Table

- Begins at sectors 58 through 62 (@3A@ through @3E@).
- Is expandable as needed.
- Provides a list of temporarily-in-use disk segments.
- At CLEAR/START time, all segments are returned to the working available table.
- One table for each user drive and one for all system drives. The system drive table is on the first drive.

General Information

The master available table and the working available table are always maintained in sorted (ascending AVL__ADDRESS) order. When entries are inserted or deleted, the table is compressed or expanded.

The temporary table is not maintained in sorted order and is not compressed and expanded.

All three types of table are extended as necessary.

All three tables have the following programmatic description.

```
RECORD
  01 SLOT_REC          BIT(60),          % SLOT RECORD
     02 DADR                BIT(36),     %   DISK ADDRESS
        03 PCU              BIT(12),     %     PORT, CHANNEL, & UNIT
           04 PC            BIT(7),      %       PORT & CHANNEL
           04 FILLER        BOOLEAN,     %
           04 EU            BIT(4),      %       UNIT
        03 DA               WORD,        %     ADDRESS
     02 LTH                 WORD;        %   LENGTH
 %
RECORD
  01 AVL_REC           BIT(SEG_SIZE),   % AVAILABLE RECORD
     02 PTRS                BIT(108),    %   POINTERS
        03 SUCC             DISK_ADDR,   %     SUCCEEDING RECORD
        03 PRED             DISK_ADDR,   %     PRECEDING RECORD
        03 SELF             DISK_ADDR,   %     THIS       RECORD
     02 TEMP_AVL_TYPE       BIT(4),      %
         % Only used in temporary tables. If the temporary table
         % requires extension segments (i.e. above the preallocated
         % 5 sectors), space for the extension segment is recorded in
         % the available table. During disk cleanup (at clear start
         % for the system disk, at pack ready time for user disks)
         % the temporary table extension segments are returned to
         % available table. Extension segments are flagged by @F@,
         % base segments are flagged by @O@.
     02 SLOTS                BIT(1320), %   SLOT RECORDS
        03 SLOT(22)         SLOT_REC,    %     22 PER AVL_REC
     02 FILLER              BIT(8);      %
```

DISK DIRECTORY

The disk directory catalogs all files on disk. The directory is a two-level (master directory and secondary directory) structure. Each master directory entry contains a file name, a type, and either the address of the disk file header (DFH) for the file or the address of the secondary directory for all double-name files with that first name. The characteristics of the directories are described next.

Master Directory

- Begins at sectors 32 through 47 (@20@ through @2F@).
- Is expandable as needed.
- Each sector contains entries for 11 files.
- For single-name files, name, DFH address, and type are listed.
- For double-name file, the first name and the secondary directory address for all files with that first name are listed.
- There is one directory for each user drive and one for all system drives. The system drive directory is on the first drive.

Secondary Directory

- Allocated as needed.
- Is expandable as needed.
- Each sector contains entries for 11 files.
- For double-name files, the second name, DFH address, and type are listed. The file name look-up algorithm hashes the first name into one of the 16 master dictionary disk sectors and then performs a sequential search.

Both directories have the following programmatic description.

```
01 DIRECTORY              BIT(1440),
   02 DISK_SUCCESSOR      DISK_ADDRESS,  % FORWARD  LINK
   02 DISK_PREDECESSOR    DISK_ADDRESS,  % BACKWARD LINK
   02 DISK_SELF           DISK_ADDRESS,  % THIS     LINK
   02 FILLER              BIT(12),
   02 DISK_NAME           CHARACTER(10), % 1ST ENTRY
   02 DISK_ADDRESS        DISK_ADDRESS   % .
   02 DISK_FILE_TYPE      BIT(4),        % .  (VALUE IS 0 OR 2)
   02 FILLER              BIT(1200);     % 10 MORE ENTRIES
```

DISK FILE HEADER

The disk file header describes the physical attributes and contains pointers to each area of a disk file. The length of a DFH varies between one and three disk sectors, depending on the number of areas declared. A disk file header for a file with fixed record length has the following characteristics:

- Allocated as needed.
- Expandable from one to three sectors.
- First sector contains physical attributes and addresses for areas 1 through 25 for fixed record length files.
- First sector contains physical attributes and addresses for areas 1 through 23 for variable record length files.
- Second sector, if any, contains addresses for areas 26 through 65 for fixed record length files.
- Second sector, if any, contains addresses for areas 24 through 63 for variable record length files.
- Third sector, if any, contains addresses for areas 66 through 105 for fixed record length files.

A programmatic description follows.

```
RECORD
  01 DFH_RECORD                          BIT(580),
    02 DFH_AREA_ADDR_OFFSET               BIT(16),
       %OFFSET INTO THE DFH (IN BITS) FOR THE FIRST AREA ADDRESS
    02 DFH_FILE_TYPE                     BIT(8),
       %TYPE OF FILE DESCRIBED BY THIS HEADER
    02 DFH_SELF                          BIT(36),
       %DISK ADDRESS OF THIS HEADER
    02 DFH_NO_USERS                      BIT(8),
       %NUMBER OF USERS WHO HAVE THIS FILE OPENED
    02 DFH_USERS_OPEN_OUT                BIT(4),
       %NUMBER OF USERS WHO HAVE THIS FILE OPENED I/O OR OUTPUT
    02 DFH_OPEN_TYPE                     BIT(4),
       %HOW THIS FILE WAS OPENED
      3 DFH_OPEN_LOCKOUT                 BOOLEAN,
      3 DFH_OPEN_LOCK                    BOOLEAN,
      3 DFH_OPEN_OUTPUT                  BOOLEAN,
      3 DFH_OPEN_INPUT                   BOOLEAN,
    02 DFH_FILE_TYPE_8_0                 BIT(4),
       %PRE-9.0 FILE TYPES
    02 DFH_PERMANENT                     BIT(4),
       %HOW PERMANENT THIS FILE IS. THE VALUES ARE ---
       %  0 = TEMPORARY - WILL BE REMOVED NEXT CLEAR/START
       %  1 = PERMANENT - NORMAL FILES CONTAIN THIS VALUE
       %  2-D          NOT    USED
       %  E = IAD FILE - CANNOT BE MOVED BY SQUASH
       %  F = SYSTEM FILE - CANNOT REMOVE, CHANGE OR SQUASH
    02 DFH_JOB_WAITING_ON_CLOSE          BOOLEAN,
       %SOMEONE ATTEMPTED TO OPEN THIS FILE BUT COULDN'T BECAUSE
       %IT IS CURRENTLY OPENED LOCK OR THE REQUESTOR WANTS TO OPEN
       %IT LOCK AND ITS IN USE. TELLS CLOSE TO CAUSE ANY JOBS
       %WAITING NO FILE WHEN THIS FILE IS CLOSED.
    02 DFH_NEWFILE                       BOOLEAN,
       %THIS FILE IS NOT IN THE DIRECTORY YET
```

```
02 FILLER                              BIT(6),
02 DFH_HDR_SIZE                        BIT(16),
     %TOTAL SIZE OF THIS HEADER (IN BITS)
02 DFH_NO_USERS_LOCK                   BIT(4),
     %NUMBER OF USERS WHO HAVE THIS FILE OPENED WITH LOCK
02 DFH_RECORD_SIZE                     BIT(20),
     %SIZE OF THE RECORDS (IN BITS)
02 DFH_FILE_LEVEL                      BIT(4),
     % 0 = 8.0 AND EARLIER
     % 1 = 9.0
02 DFH_RCDS_BLOCK                      BIT(20),
     %NUMBER OF RECORDS PER BLOCK
02 DFH_BLOCKS_AREA                     BIT(24),
     %NUMBER OF BLOCKS PER AREA
02 DFH_SEGS_AREA                       BIT(24),
     %NUMBER OF SEGMENTS OR SECTORS PER AREA
02 DFH_AREAS_RQST                      BIT(12),
     %MAXIMUM NUMBER OF AREAS ALLOWED IN THIS FILE
02 DFH_AREA_CTR                        BIT(12),
     %CURRENT HIGH AREA NUMBER ALLOCATED
02 DFH_EOF_POINTER                     BIT(24),
     %HIGHEST RECORD NUMBER WRITTEN IN THIS FILE
02 DFH_AUDITED                         BOOLEAN,
      % DO NOT REINSTATE USER UNTIL I/O IS COMPLETE
02 DFH_PROTECTION_ATTR                 BIT(2),
      % HOW BADLY DOES THE USER WISH TO SAVE THIS FILE IN THE
      % OFF CHANCE OF A CLEAR/START WHILE OPEN ?
      % 0 = TEMPORARY
      % 1 = ABNORMALSAVE
      % 2 = SAVE
      % 3 = PROTECTED
02 FILLER                              BOOLEAN,
02 DFH_BPS_NO                          BIT(20),
     %SERIAL NUMBER OF THE BASE PACK TO WHICH THIS MULTI-PACK
     %FILE BELONGS
02 FILLER                              BIT(27),
02 DFH_MPF                             BOOLEAN,
     %THIS IS A MULTI-PACK FILE
02 DFH_UPDATE_DATE                     BIT(16),
     %JULIAN DATE OF THE LAST TIME THIS FILE WAS CLOSED AFTER
     %HAVING BEEN WRITTEN ON. ALSO DATE OF LAST NAME CHANGE.
     %FOR CODE FILES, ITS THE DATE OF THE LAST MODIFY.
02 FILLER                              BIT(4),
02 DFH_CREATE_TIME                     BIT(20),
     %TIME THE FILE WAS OPENED OUTPUT NEW.
02 FILLER                              BIT(32),
02 DFH_SAVE_FACTOR                     BIT(12),
     %NUMBER OF DAYS TO SAVE THIS FILE. NO SIGNIFICANCE.
02 DFH_CREATION_DATE                   BIT(16),
     %JULIAN DATE OF WHEN THIS FILE WAS OPENED OUTPUT NEW.
02 DFH_ACCESS_DATE                     BIT(16),
     %JULIAN DATE OF WHEN THIS FILE WAS LAST OPENED. FOR CODEFILES,
     %DATE LAST EXECUTED OR MODIFIED.
02 FILLER                              BIT(61),
02 DFH_UPDATE_VERSION                  BOOLEAN,
     %DMS USE ONLY
02 FILLER                              BIT(2),
02 DFH_VERSION                         BIT(36),
     %TIME AND DATE OF THE LAST CLOSE. I/S AND DMS ONLY.
02 DFH_PROTECTION                      BIT(2),
     % 0 = PUBLIC FILE
     % 1 = PRIVATE FILE
```

```
    02 DFH_PROTECTION_IO                    BIT(2),
         % 0 = ACCESS MAY BE I/O
         % 1 = ACCESS MAY BE INPUT ONLY
         % 2 = ACCESS MAY BE OUTPUT ONLY
    02 DFH_USERS_FROZEN                     BIT(8),
         % NUMBER OF USERS FOR WHO THIS HEADER CANNOT BE ROLLED OUT
    02 FILLER                               BIT(8),
    02 DFH_MINRECSIZE                       BIT(20),
         %MINIMUM NUMBER OF BITS IN EACH LOGICAL RECORD
    02 DFH_MAXRECSIZE                       BIT(20);
         %MAXIMUM NUMBER OF BITS IN EACH LOGICAL RECORD


RECORD
   01 AN_AREA_ADDRESS                       BIT(36),
      02 DFH_UNIT                           BIT(12),
         03 DFH_PC                          BIT(7),
            04 DFH_PORT                     BIT(3),
            04 DFH_CHAN                     BIT(4),
         03 DFH_SER_NO_FLAG                 BOOLEAN,
         03 DFH_EU                          BIT(4),
      02 DFH_ADDR                           BIT(24);


RECORD
   01 AREA_ADDRS                            BIT(3780),
      02 AREA_ADDR (105)                    AN_AREA_ADDRESS,
      02 FIRST_AREA REMAPS AREA_ADDR AN_AREA_ADDRESS;
```

B 1000 FILE TYPES

```
DEFINE
     DATA_TYPE_FILE(X)                     AS                                %
      #((X = 0 OR X = 9 OR X = 11)   OR   (X >= 13 AND X <=  15)
         OR  (X >= 17 AND X <= 22)  OR   (X >= 60 AND X <= 109))#,

     CODE_TYPE_FILE(X)                     AS                                %
         #(X = 8 OR (X >= 110 AND X <= 139))#,

     MICROCODE_TYPE_FILE(X)                AS                                %
         #(X = 7 OR (X >= 140 AND X <= 169))#;

CONSTANT

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %                                                                       %
 %          VALID FILE TYPES : DFH_FILE_TYPE, FPB_FILE_TYPE              %
 %          -----------------------------------------------              %
 %                                                                       %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %                                                                       %
 %                                     SBP                         CSG   %
 %   DESCRIPTION                       CODE            ABVR        CODE  %
 %-----------------------------------------------------------------------%
 %                                                                       %
 % S Y S T E M    F I L E S                                              %
 %                                                                       %
     LOG_FILE                     =   001,  %      "LOG "          000   %
     DIRECTORY_FILE               =   002,  %      "DIR "          301   %
     PSR_DECK                     =   003,  %      "DECK"          306   %
     PRT_FILE                     =   004,  %      "PRT "          304   %
     PCH_FILE                     =   005,  %      "PCH "          305   %
     DUMP_TYPE_FILE               =   006,  %      "DUMP"          000   %
     INACTIVE_LOG_FILE            =   010,  %      "TLOG"          000   %
     INTRIN_FILE                  =   012,  %      "INSC"          119   %
     USERCODE_FILE                =   016,  %      "USER"          000   %
     MCP_TEMPORARY_FILE           =   023,  %      "TEMP"          000   %
     JOBLOG_FILE                  =   024,  %      "JLOG"          000   %
 %                                                                       %
 %   025 <-----> 059       RESERVED FOR ADDITIONAL SYSTEM FILES          %
 %                                                                       %
 %-----------------------------------------------------------------------%
 %                                                                       %
 % D A T A    F I L E S                                                  %
 %                                                                       %
     UNSPECIFIED_DATA_FILE        =   000,  %      "DATA"          000   %
     DATA_FILE                    =   009,  %      "DATA"          000   %
     VARIABLE_LENGTH_FILE         =   011,  %      "VAR "          000   %
     DMS_DATA_FILE                =   013,  %      "DMS "          000   %
     DMS_DICTIONARY               =   014,  %      "DMSD"          000   %
```

```
        DMS_AUDIT_FILE                  =   015,  %      "AUDT"      000  %
        RELATIVE                        =   017,  %      "REL "      000  %
        INDEX_SEQ_GLOBAL_FILE           =   018,  %      "IS.G"      000  %
        INDEX_SEQ_DATA_SET_FILE         =   019,  %      "IS.D"      000  %
        INDEX_SEQ_INDEX_FILE            =   020,  %      "IS.I"      000  %
        INDEXED_TAG_FILE                =   021,  %      "TAGS"      000  %
        INDEXED_DATA_FILE               =   022,  %      "INXD"      000  %
        SDLSYMBOL                       =   060,  %      "SDL "      000  %
        COBOL68SYMBOL                   =   061,  %      "C068"      004  %
        RPGSYMBOL                       =   062,  %      "RPG "      031  %
        NDLSYMBOL                       =   063,  %      "NDL "      012  %
        FORTRANSYMBOL                   =   064,  %      "FOR "      010  %
        MILSYMBOL                       =   065,  %      "MIL "      000  %
        BASICSYMBOL                     =   066,  %      "BAS "      002  %
        UPLSYMBOL                       =   067,  %      "UPL "      000  %
        COBOL74SYMBOL                   =   068,  %      "C074"      035  %
        FORTRAN77SYMBOL                 =   069,  %      "F77 "      033  %
        IBASICSYMBOL                    =   070,  %      "IBAS"      000  %
        DASDLSYMBOL                     =   071,  %      "DASD"      005  %
        PASCALSYMBOL                    =   072,  %      "PASC"      030  %
        SDL2SYMBOL                      =   073,  %      "SDL2"      000  %
        NON_NATIVEDATA                  =   074,  %      "IDTA"      000  %
        IBASIC_INTERNAL                 =   075,  %      "IBSU"      000  %
        FORTRAN77_UNFORMATTED           =   076,  %      "F77U"      000  %
        COBDMS_LIB                      =   077,  %      "CDMS"      000  %
        RPGDMS_LIB                      =   078,  %      "RDMS"      000  %
        NETWORK_INFORMATION             =   079,  %      "NIF "      000  %
        NDL_LIBRARY                     =   080,  %      "NDLB"      000  %
        PASCAL_INTERCHANGE_DATA         =   081,  %      "PIDF"      000  %
        PASCAL_MODULE_DATA              =   082,  %      "PSMD"      000  %
        SORTSYMBOL                      =   083,  %      "SRTS"      034  %
        SEQDATA                         =   084,  %      "SEQD"      020  %
        JOBSYMBOL                       =   085,  %      "JOBS"      026  %
        UPL2SYMBOL                      =   086,  %      "UPL2"      000
        WFL_DATA_DECK                   =   087,  %      "DATA"      UNK
%                                                                        %
%       086 <-----> 109     RESERVED FOR ADDITIONAL DATA FILES           %
%                                                                        %
%------------------------------------------------------------------------%
%                                                                        %
%   C O D E   F I L E S                                                  %
%                                                                        %
        CODE_FILE                       =   008,  %      "CODE"      100  %
        SDLCODE                         =   110,  %      "SDLO"      100  %
        COBOL68CODE                     =   111,  %      "C680"      103  %
        RPGCODE                         =   112,  %      "RPGO"      162  %
        NDLCODE                         =   113,  %      "NDLO"      109  %
        FORTRANCODE                     =   114,  %      "FORO"      107  %
        MILCODE                         =   115,  %      "MILO"      100  %
        BASICCODE                       =   116,  %      "BASO"      102  %
        UPLCODE                         =   117,  %      "UPLO"      100  %
        COBOL74CODE                     =   118,  %      "C740"      165  %
        FORTRAN77CODE                   =   119,  %      "F770"      163  %
        IBASICCODE                      =   120,  %      "IBSO"      100  %
        PASCALCODE                      =   121,  %      "PSCO"      161  %
        SDL2CODE                        =   122,  %      "SD20"      100  %
        SMCPCODE                        =   123,  %      "SMCP"      120  %
        MCSCODE                         =   124,  %      "MCSO"      100  %
        NON_NATIVECODE                  =   125,  %      "ICDE"      100  %
        B500CODE                        =   126,  %      "B50 "      100  %
        IBM1400CODE                     =   127,  %      "IBMO"      100  %
```

```
    SORTCODE                        =   128,  %       "SRTO"        164  %
    SDL2UNBOUNDCODE                 =   129,  %       "SD2U"        100  %
    JOBCODE                         =   130,  %       "JOBO"        115  %
    UPL2CODE                        =   131,  %       "UP2C"        100

    131 <-----> 139     RESERVED FOR ADDITIONAL CODE FILES

M I C R O C O D E   F I L E S

    INTERPRETER_FILE                =   007,  %       "INTP"        130
    SDLINTERPRETER                  =   140,  %       "SDLI"        130
    COBOL68INTERPRETER              =   141,  %       "C68I"        122
    RPGINTERPRETER                  =   142,  %       "RPGI"        125
    NDLMICROCODE                    =   143,  %       "NDLI"        130
    FORTRANINTERPRETER              =   144,  %       "FORI"        123
    BASICINTERPRETER                =   145,  %       "BASI"        130
    UPLINTERPRETER                  =   146,  %       "UPLI"        130
    COBOL74INTERPRETER              =   147,  %       "C74I"        130
    FORTRAN77INTERPRETER            =   148,  %       "F77I"        130
    IBASICINTERPRETER               =   149,  %       "IBSI"        130
    PASCALINTERPRETER               =   150,  %       "PSCI"        130
    SDL2INTERPRETER                 =   151,  %       "SD2I"        130
    MICROMCP                        =   152,  %       "MMCP"        130
    GISMO                           =   153,  %       "GSMO"        130
    SYSTEM_INITIALIZER              =   154,  %       "INIT"        130
    B500INTERPRETER                 =   155,  %       "B5I "        130
    IBM1400INTERPRETER              =   156;  %       "IBMI"        130
        %   END OF FILE TYPE DEFINES ***********************************

    157 <-----> 169     RESERVED FOR ADDITIONAL MICRO CODE FILES

R O O M   T O   G R O W

    170 <-----> 199       FUTURES
    200 <-----> 219       RESERVED FOR DISKMAP
    220 <-----> 255       FUTURES
```

FILE DICTIONARY

Every program that has any files declared for input of output has a file dictionary. The dictionary is a set of system descriptors. There is one system descriptor for each file declared, plus one for the trace file. (Appendix E, Memory Management, includes a programmatic description of a system descriptor.) The trace file is used by debug interpreters to write trace information. The RAID system file equates the trace file to a queue file that has been opened by RAID.

At the time a system memory dump is executed, each file in the dictionary is either open or closed. If the MEDIA field FIB (file information block) dictionary contains MEDIA, the File Information Block (FIB) is in memory at the address in the ADDRESS field, and the file is open if FIB.OPEN_FLAG is TRUE. If the MEDIA field is blank and the address field contains zeros, the file is closed. If the MEDIA field is blank and the address field is non-zero, the file information block and any file buffers are rolled out to disk, and the file is open if FIB.OPEN_FLAG is TRUE.

FILE INFORMATION BLOCK (FIB)

As each file is opened by the user program, a File Information Block (FIB) is created in memory by the MCP. The FIB contains all information necessary for the MCP to perform I/O operations on the file. Much of the information in the FIB is taken directly from the file declaration in the user program. Other information is inserted by the MCP, based upon the characteristics of the peripheral device assigned to the file.

I/O descriptors and buffer memory areas are allocated and initialized when the file is opened. There is only one memory link for each file that is open. Buffer areas and descriptors are not normally shared between files, although exceptions to this rule include DMS, Data Comm, and Indexed files.

FIB size depends upon the type of device assigned to the file. Due to the amount of information that must be maintained, a disk file FIB is much larger than a card punch file FIB.

A complete programmatic description of a FIB follows.

```
CONSTANT
        FIB_SIZE              =  1048,
        FIB_COMMON_SIZE       =  268,
        FIB_SIZE_EXTRA_BNALIO = 402,
        FIB_SIZE_QUEUE        =  433, % INCLUDES 1 ELEMENT
        FIB_SIZE_PORT         =  451, % INCLUDES 1 SUBFILE
        SUBPORT_ARRAY_SIZE    =  77,
        FIB_SIZE_NDL          =  933,
        FIB_SIZE_UFW          =  1048,
        FIB_SIZE_DIAGNOSTIC   =  442,
        FIB_UNIQUE_SIZE       =  24717,  % QUEUE PART
        FIB_SIZE_DISK         =  FIB_SIZE,
        FIB_SIZE_BASIC        =  684,
        FIB_SIZE_TAPE         =  796,
        FIB_SIZE_PRINTER      =  FIB_SIZE_TAPE,
        FIB_SIZE_MICR         =  1048,
        MAX_MAX_SUBPORTS      =  255,
        FIB_ORGANIZATION_RELATIVE = 1,
        FIB_ORGANIZATION_INDEX_SEQ = 2;
```

```
 %
RECORD FIB_SUBPORT_ATTR_RECORD
     CHANGEEVENT    BOOLEAN,  %
     STATE          BIT (4),  %
     QIN_PTR        ADDRESS,  % USERS INPUT  SUBPORT ADDRESS.
     QOUT_PTR       ADDRESS,  % USERS OUTPUT SUBPORT ADDRESS.
     ERROR          BIT (8),
     MAX_REC_SIZE   BIT (16);
 %
RECORD 1 FIB_RECORD BIT(FIB_COMMON_SIZE + FIB_UNIQUE_SIZE),
 2 COMMON   BIT(FIB_COMMON_SIZE), % COMMON TO ALL DEVICES
   3 BOOLEANS BIT(58), % COMMON TO ALL DEVICES
     4 OPEN_FLAG    BOOLEAN,
                    % 1 = FILE HAS BEEN OPENED, NOT YET CLOSED,
     4 CLOSING      BOOLEAN,%
                    %1= FILE IS BEING CLOSED, BUT HAS TO WAIT
                    %  FOR I/O TO COMPLETE.
     4 MYUSE        BIT(2),
                    %  PECULIAR 2 BIT FIELD USED BY NDL
       5 OUTPUT     BOOLEAN,%
                    %1 = FILE IS OPENED FOR OUTPUT,
                    %  MAY BE NEW OR OLD FILE.
       5 INPUT      BOOLEAN,%DONT CHANGE POSN FIB.OUTPUT.
                    %1 = FILE IS AN OLD FILE OPENED FOR
                    %    INPUT OR OUTPUT OR BOTH(RANDOM OR SEQ)
     4 STOP_IO    BOOLEAN,%
                    %1 = I/O STOPPED, DUE TO I/O ERROR OR EOF.
     4 ENHANCED_IO_PERMITTED BOOLEAN,%
                    %0 = SUPPRESS ENHANCED I/O. NOT USED YET.
     4 VARIABLE   BOOLEAN,%
                    %1 = FILE HAS VARIABLE LENGTH RECORDS,
                    %  HENCE FIB.RECORDS.BLOCK WILL BE SET TO 1
                    %  FIB.ACCESS MUST NOT BE 1(RANDOM I/O),
                    %  FILE POSITIONING IS NOT ALLOWED YET.
     4 COBOL_FILEF BOOLEAN,
                    %1 = FILE IS TO BE HANDLED AS COBOL FILE.
                    %  HENCE HANG PROGRAM IF OPEN IS INVOKED
                    %  WHILE FILE IS ALREADY OPEN,
                    %  CLOSE IS INVOKED WHILE FILE IS NOT YET
                    %  OPEN OR ALREADY CLOSED, OR EOF HIT
                    %  TWICE.
     4 LABELED    BOOLEAN,%
                    %1 = FILE IS LABELED(MEANINGLESS FOR DISKS)
     4 PSEUDO     BOOLEAN,%
                    %1 = FILE IS A PSEUDO READER ON DISK.
     4 BACKUP     BOOLEAN,%
                    %1 = FILE IS BACKUP PRINTER OR PUNCH ON
                    %  TAPE OR DISK.
     4 DMS        BOOLEAN,%
                    %1 = FILE IS DATA MANAGEMENT.
     4 DEVICE_FLAGS  BIT(14),%
                    %ACTUAL DEVICE TYPE. VERY USEFUL.
       5 REVERSE BOOLEAN,%
       5 CRD96        BOOLEAN,%
       5 DATA_RCDR BOOLEAN,
       5 DISK_DEVICE BOOLEAN,
                    %1 = FILE IS ACTUALLY ON DISK,
                    %  PROBABLY PSEUDO.READER OR BACKUP OR
                    %  "FILE EQUATED" BEFORE OPEN TIME AS DISK.
       5 DISK_PACK_DEVICE BOOLEAN,
       5 TAPE_DEVICE BOOLEAN,%
       5 REM_BACKUP BOOLEAN,%
       5 PUNCH      BOOLEAN,%
```

```
  5 MCP_CLOSE_REEL BOOLEAN,%
  5 EOF_REPORTED BOOLEAN,%
              %1 = EOF HIT ONCE.  IF FIB.COBOL, THEN
              %  NEXT HIT ON EOF WILL HANG PROGRAM.
  5 PRINTER_DEVICE BOOLEAN,
  5 TRACE        BOOLEAN,% THIS IS A TRACE.FIB
  5 LINAGE_CODE   BOOLEAN,  % INVOKE LINE COUNTER CODE
  5 PRINTER_CHECK BOOLEAN,%
4 IO_SEQ_WRITE_REQ BOOLEAN,%
4 IO_ERROR_INFO BIT(7),
  5 IO_ERROR_SEEN_SMCP BOOLEAN,%
  5 IO_ERROR_THIS_COMQ BOOLEAN,%
  5 IO_ERROR_ON_READ BOOLEAN,
  5 IO_ERROR_TYPE BIT(4),%
              % 1 = EOF OR EOP
              % 2 = PARITY
              % 3 = INCOMPLETE I/O
              % 4 = EOF ON PAPER TAPE READERS
              % 5 = EOV ON LDDMP MULTIFILE SEARCH
4 TEMPORARY_FREEZE BOOLEAN, % LOCK FIB IN MEMORY
4 DUMMY_FILE BOOLEAN,
              % NO LOGICAL I/O TO BE DONE
4 CANNOT_CLOSE_LOCK BOOLEAN,%
              % OPENED ON-BEHALF-OF BUT WITH ERROR;
4 AUDITED       BOOLEAN,%
              % DO NOT REINSTATE USER UNTIL I/O IS COMPLETE
4 MPF                BOOLEAN,% MULTIPACK DISK FILES.
4 CLOSED_HERE        BOOLEAN, % MPF WAITING DISK.
4 EOT                BOOLEAN,% END OF TAPE REEL.
              %THIS BIT ALSO MEANS END OF BACKUP IF FIB DISK
4 EMULATOR_TAPE      BOOLEAN,%
4 CYL_ALLOC          BOOLEAN,% USER WANTS CYLNDR BOUNDS
4 WAITNEWAREA        BOOLEAN,%
              % I/O STOPPED WAITING FOR NEW AREA ON MULTI
              % PACK FILE(INPUT OR OUTPUT) OR ELSE NON
              % MULTIPACK  OUTPUT FILE NEEDS DISK SPACE.
4 NEWAREA            BOOLEAN,%
              % NEW AREA HAS BEEN CREATED ON OLD OR NEW
              % DISK FILE, HENCE AT CLOSE OR DS, AREAS IN
              % TEMPORARY TABLE MUST BE REMOVED, OR ELSE
              % NEXT CLEAR/START WILL CLOBBER DISK.
4 SPECIAL_EU         BOOLEAN,%
              % EU NO SPECIFIED BY USER FOR HPT ONLY.
4 OPEN_LOCK          BOOLEAN,%
4 MCPINTERNAL        BOOLEAN,% "PARAMETERS" FILE FOR LO
              % LOAD.DUMP USE ONLY.
4 NEWFILE            BOOLEAN,% OUTPUT NEW, FIB.INPUT=0.
4 LABEL_IN_PROCESS    BOOLEAN,%USED IN PARTICULAR FOR
              %D.RECORDER OPEN LABEL.
4 PRINTER_FILE        BOOLEAN, %FILE IS PRINTER HARDWARE
4 FILE_WRITTEN_ON     BOOLEAN, %DISK FILE UPDATED
4 PROTECTION_ATTR     BIT(2),
              %  0 = TEMPORARY
              %  1 = ABNORMALSAVE
              %  2 = SAVE
              %  3 = PROTECTED
4 OPTIONAL_FILE           BOOLEAN,
4 OPEN_LOCKOUT            BOOLEAN,
4 DIAGNOSTIC_FILE         BOOLEAN,
              % INDICATES THIS IS A DIAGNOSTIC FILE
```

```
    4 BNA_FILE                  BOOLEAN,
                      % HAVE ONLY LOGICAL, NO PHYSICAL, FILE.
                      % ALSO MEANS THAT PORT_STATE_VARIABLES (RECORD AT
                      %   END OF FIB) LIVE AT END OF FIB EXCEPT FOR
                      %   DEVICE = PORT FILE. PORT_STATE_VARIABLES ARE
                      %   USED ONLY IN BNA LOGICAL I/O ACROSS NETWORK,
                      %   DEVICES SUCH AS DISK, TAPE, CARD, PRINTER.
  3 ORGANIZATION          BIT(4),
                      % IDENTIFIES THE TYPE OF FIB BEING USED
                      %    0 - ALL FILES SUPPORTED PRIOR TO 9.0
                      %    1 - RELATIVE FILE
                      %    2 - INDEXED/SEQUENTIAL FILE
  3 HDWR                  BIT(6),%
                      %  FOR MPF, IT PERTAINS TO
                      %  THE HARDWARE TYPE OF THE BASE PACK.
                      %  NOTHING PREVENTS US MIXING DEVICES FOR
                      %  MULTIPACK FILES.
  3 VERSION       BIT(8),%
                    %FOR FUTURE USE, IN CASE OF INCOMPATIBLE
                    %  FIB STRUCTURES.  RIGHT NOW IT IS 1.
  3 REAL_SIZE     BIT(16),%
                    %REAL LENGTH OF FIB, IN BITS.
                    %  SET UP BY OPEN IN "NO.FILE.SPACE".
  3 ENHANCED_IO_STATUS BIT(8),%
                    %FOR ENHANCED I/O TO SAY:
                    %  I'VE DONE SO MUCH SO FAR, YOU S.MCP
                    %  CONTINUE WITH  IT."
  3 FILE_NUMBER     BIT(8),%
                    %= CT.OBJECT, = ENTRY NUMBER IN FIB.DICT.
  3 RS            BIT(24),%
                    %ADDRESS OF LIMIT.REGISTER OF RUN.STRUCTURE
                    %  ONLY ONE RUN STRUCTURE ALLOWED PER FIB.
  3 ERR_REPORT_FLAGS BIT(24),
    4 ERR_PRESENT BOOLEAN,
                    %ERROR OCCURRED DURING POCKET SELECT
    4 FILLER BIT(10),
    4 WHO_FOUND_ERR BOOLEAN,
                    % 0=MMCP FOUND THE ERROR
                    % 1=INTERPRETER FOUND THE ERROR
    4 INTERP_ERROR BIT(8),
                    % SAME AS RS_ITYPE CAT RS_INMBR
    4 MMCP_ERROR BIT(4),
                    % ERROR FOUND BY MMCP AS FOLLOWS:
                    %  0=INVALID COMMUNICATE FROM USE ROUTINE
                    %         (FATAL)
                    %  1=ILLEGAL POCKET SELECT (FATAL)
                    %  2=JAM (NON-FATAL)
                    %  3=MISSORT (NON-FATAL)
                    %  4=NOT READY (NON-FATAL)
  3 TRANSLATE_TABLE BIT(24),
    4 DMSGLOBALS BIT(24),
                    %ADDR OF SOFT TRANSLATE TABLE IF PRESENT
  3 ERRORS BIT(16),
                    % NO OF RETRIES FOR THIS FILE
  3 LIO_FILE_STATUS BIT(24),
                    % CHANGED ON EVERY LIO.
  3 NEXT_PORT           ADDRESS,       % NEXT PORT FIB ADDRESS.
  3 BACK_PORT           ADDRESS,       % BACK PORT FIB ADDRESS.
                    %%%%% END OF COMMON PORTION %%%%%%%%%%%%%%%%%%%%%%%%%%
2 UNIQUE BIT(FIB_UNIQUE_SIZE),
  3 FILLER BIT(3),
  3 RETRY_COUNT BIT(5),%
                    %USED BY IO.ERROR TO COUNT RETRIES.
```

```
3 UNIT          BIT(12),%
                  %DEVICE'S PCU.  FOR MPF WE NEED ALSO
                  %  FIB.DISK.UNIT FOR CURRENT PACK.
  4 CHANNEL       BIT(7),%
                  %BOTH PORT AND CHANNEL.  POOR NOMENCLATURE.
  4 FILLER        BOOLEAN,%
  4 UNIT_NO       BIT(4),%
3 IOAT_ADDR       BIT(24),%  ADDRESS OF IOAT ("UNIT" TABLE)
                  %  QUEUE-FILES USE DIFFERENT FIB HENCEFORTH
3 MINRECSIZE    BIT(16),
                  %MINIMUM RECORD SIZE IN BITS
3 MAXRECSIZE    BIT(16),
                  %MAXIMUM RECORD SIZE IN BITS
3 RECORD_DESC   BIT(48),%
                  %A DESCRIPTOR OF  THE CURRENT RECORD
                  %  (THE ONE TO BE PROCESSED, NOT THE ONE
                  %  JUST PROCESSED). LOGICAL I/O ENTITY.
  4 FILLER        BIT(8),%
  4 RECORD_SIZE   BIT(16),%
                  %  CONSTANT UNLESS
                  %  VARIABLE LENGTH RECORDS.
    5 ALPHA_SIZE BIT(13),%
                  %TO BE USED IF RECORDS ARE CHARACTER. TYPE.
  4 RECORD_ADDR   BIT(24),%
                  %
3 CURRENT       BIT(24),%
  4 DESCRIPTOR_A  BIT(24),
                  %ADDRESS OF CURRENT I/O DESCRIPTOR, HENCE
                  %  INDIRECTLYT THE CURRENT BUFFER.
                  %  LOGICAL I/O ENTITY.
3 RECORDS_BLOCK BIT(20),%
                  %  =1 IF VARIABLE LENGTH
                  %  RECORDS.  CONSTANT VALUE.
3 BUFFER        BIT(20),%
                  %REMAINING SPACE IN BUFFER,
                  %  INCLUDING THE CURRENT RECORD DESCRIBED
                  %  BY FIB.CURRENT, IN BITS.
                  %  LOGICAL I/O ENTITY.
  4 BITS_LEFT_IN_BUFFER   BIT(20),
                  % FOR DMS AUDIT, SIMILAR TO REGULAR MEANING
3 BUFFER_EMPTY  BIT(24),%
                  %FOR INPUT FILES ONLY. A PHYSICAL I/O
                  %  ENTITY.  IF PHYSICAL I/O STOPPED (E.G.,
                  %  WAITING FOR INPUT MPF PACK, OR EOF) THIS
                  %  WILL POINT TO THE OLDEST IO DESCRIPTOR
                  %  WHOSE BUFFER IS UNFILLED.  LOGICAL I/O
                  %  IS ALLOWED TO CATCH UP TILL FIB.BUFFER.
                  %  EMPTY = FIB.CURRENT.
  4 BUFFER_NEXT_AVAILABLE BIT(24),
                  % FOR DMS AUDIT, CONTAINS THE ABSOLUTE ADDRESS OF
                  % THE NEXT FREE POSITION IN THE BUFFER.
3 BLOCK_COUNT   BIT(24),%
                  %  PREVIOUSLY A PHYSICAL
                  %  ENTITY FOR DISK FILES, BUT SINCE 5.1
                  %  A LOGICAL I/O ENTITY. FOR INPUT FILES IT
                  %  IS BUMPED ONLY WHEN THE FIRST RECORD HAS
                  %  BEEN TRANSFERRED TO THE USER, HENCE
                  %  PROVING THE BLOCK TO BE NON-EMPTY
3 BLOCK_SIZE    BIT(20),%
                  %  CONSTANT.
3 RECORD_COUNT  BIT(24),%
                  %  NUMBER OF RECORDS PROCESSED SO FAR
                  %  EXCLUDING THE CURRENT RECORD.
```

```
3 SEGS_AREA BIT(24),%NO. OF SEGMENTS PER AREA
3 EOF_PTR     BIT(24),
                %FOR OLD FILES, IT IS THE NUMBER OF ACTUAL
                %  DATA RECORDS TO BE COMPARED WITH FIB.
                %  RECORD.COUNT.  IT IS THE ONLY WAY TO
                %  CATCH EOF FOR VAR.LENGTH RECORDS CASE.
3 CHANNEL_INFO BIT(24),%
                %ADDRESS OF CHANNEL TABLE.
3 CLOSE_TYPE  BIT(36),%
                %36 BITS WITH SAME STRUCTURE AS CT_ADVERB AND
                %  CT_1 IN CLOSE.  BITS ARE SET ON BY
                %  CLOSE, IF SOME OTHER BITS ARE ON.
3 ACCESS      BIT(4),%
                %TOO TROUBLESOME TO SAY "IF FIB.DISK THEN "
                %  EVERYTIME WE TEST IT, SO WE FORCE IT ON
                %  ALL PERIPHERALS
                %     0  SERIAL
                %     1  RANDOM
                %     2  SEQUENTIAL.IO
                %     5  EMULATOR.TYPE
                %     6  DELAYED.RANDOM
                %     7  EXTENDED.SEQUENTIAL.IO
                %     8  DYNAMIC
                %  FIB.INPUT AND FIB.OUTPUT MUST BE SET
                %  TO ONE IF FIB.ACCESS <> 0
3 KEY               BIT(24),%
                % USED BY DISK AND TAPE FILES FOR LOGICAL
                % RECORD NUMBER, REPLACING FIB.RECORD.COUNT
                % IN THE OLD DAYS, LOGICAL I/O ENTITY.
3 USE_AREA    BIT(48),%
                %USED BY DISK FILES.
  4 SPACE_CTR       BIT(8),% COUNTER FOR SKIPPING
                % BLANK LINES ON PRINT.
  4 PAGE_SIZE       BIT(8),%LOGICAL PAGE SIZE
                %(TEMP.PAGE.SIZE + TEMP.TOP.MARGIN +
                %TEMP.BOTTOM.MRGN)
  4 UPPER_MARGIN    BIT(8),%ABSOLUTE LINE # A LINE
                %OF PRINT MAY START.
  4 FOOTING         BIT(8),%ABSOLUTE LINE # WHERE
                %EOP IS TO BE REPORTED.
  4 LOWER_MARGIN    BIT(8),%ABSOLUTE LINE # TO STOP
                %PRINTING ON CURRENT PAGE.
  4 LINAGE_COUNTER  BIT(8),%KEEPS TRACK OF THE LINE
                %# FOR EACH PAGE.
3 USE_ROUTINE       BIT(32),% USED BY MICR
                %
  4 TEMP_PAGE_SIZE  BIT(8),%TOTAL # OF LINES WRITING
                %IS PERMITTED. (BODY).
  4 TEMP_UPPER_MARGIN BIT(8),%TOTAL LINES IN THE TOP
                %MARGIN.
  4 TEMP_FOOTING    BIT(8),%LINE # WITHIN THE BODY
                %WHERE EOP IS REPORTED
  4 TEMP_LOWER_MARGIN BIT(8),%PRINTER FIB ENDS HERE.
                %MICR FIB ENDS HERE.
                %TOTAL # OF LINES IN THE
                %BOTTOM MARGIN.
3 LAST_OP           BIT(4),%LIO PREVIOUS OP CODE.
3 LAST_SPACING      BIT(4),%LIO PREVIOUS SPACING
3 CURRENT_OP        BIT(4),%LIO CURRENT OP CODE.
3 FIRST_WRITE_BACKUP BOOLEAN,%FOR MMCP
3 FILLER                BIT(1),
3 COUNT_CARDS           BIT(1),
3 COUNT_LINES           BIT(1),
```

```
3 LIO_CURRENT_STATUS BIT(3),
  4 LTO_OVERFLOW      BOOLEAN,%SPACE PAST BOTTOM MARGN
  4 LIO_EOP           BOOLEAN,%SPACE INTO FOOTING.
  4 LIO_WRITE_PENDING BOOLEAN,%POSITION BEFORE WRITE.
3 FILLER                BOOLEAN,
3 PRINTER_TYPE        BIT(2),
                % 00  450-700 LINE PER MINUTE PRINTER.
                % 01  1100 LINE PER MINUTE PRINTER.
                % 10  NOT USED.
                % 11  1500 LINE PER MINUTE PRINTER.
3 DISK_ADDRESS        BIT(36),%
                % DISK ADDRESS OF BLOCK LAST RELEASED
                % (FOR OUTPUT FILES) OR LATELY READ-INT
  4 DISK_PCU            BIT(12),%
    5 DISK_PC             BIT(7),
    5 SER_NO_FLAG         BOOLEAN,
    5 DISK_EU             BIT(4),
  4 DISK_SG             BIT(24),
3 HEADER                BIT(24),%
                % OFFSET IN DFH.DER OF DISK FILE HEADER
                %MAY BE SHRUNK TO 12 BITS IN FUTURE.
3 AREAS                 BIT(8),%
                % IF FIB.OUTPUT THEN MAXIMUM POSSIBLE AREAS
                % REQUESTED BY USER, ULSE ACTUAL NUMBER
                % OF AREAS CONTAINING DATA.
3 AREA_NUMBER BIT(8),%
                %PHYSICAL I/O ENTITY. AREA NUMBER (COUNTING
                %  FROM ZERO) WHERE LATEST I/O OCCURRED.
                %  WARNING: VALUE SHOULD BE <105
                %  EXCEPT 255, WHICH MEANS -1.
3 BLOCKS_AREA BIT(24),%
                %  SINCE RELEASE 5.1 WILL BE
                %  ABLE TO HANDLE PARTIAL BLOCK AT THE END
                %  OF EACH AREA.  PREVIOUS RELEASES SIMPLY
                %  IGNORE THEM.
3 BPA_COUNT   BIT(24),%
                %PHYSICAL I/O ENTITY.
                %  VALUE=1+"FIB.BLOCKS.AREA" - NUMBER OF
                %  DATA BLOCKS IN CURRENT AREA PRIOR TO AND
                %  EXCLUDING THE CURRENT BLOCK.  IN SERIAL
                %  I/O, WHEN THE SECOND LAST BLOCK IS
                %  RELEASED BY RELEASE.BUFFER, FIB.BPA.CT
                %  FOR OUTPUT FILES WILL BE REDUCED FROM 3
                %  TO 2 AND FOR INPUT FILES FROM 2 TO 1.
                %  AWKWARD, BUT PRECISE.  A CALL FOR
                %  NEW.AREA IS TRIGGERED WHEN RELEASE.BUFFR
                %  FINDS THAT FIB.BPA.CT IS TO BE REDUCED
                %FROM 1 TO 0.  NEW.AREA WILL RESET IT TO
                %  TO (FIB.S.A+FIB.S.B-1)/FIB.S.B=FIB.B.A
3 SEGS_BLOCK BIT(12),%
                %  A CONSTANT. PREVIOUSLY
                %  CALLED FIB.SEGS.
3 RECORDS_AREA BIT(24),%
                %  MAY NOT NECESSARILY BE
                %  A MULTIPLE OF FIB.RECORDS.BLOCK.
                %  MEANINGLESS FOR VARIABLE LENGTH RECORDS.
3 EU_DRIVE    BIT(4),%
                %NOT USED YET.
3 PSEUDO_RDR  BIT(24),%
                %ADDRESS OF PSEUDO READER TABLE.
3 MAX_RECORDS BIT (24),%
3 PARTIAL_BLOCK_SIZE BIT(24),%=FIB.S.A MOD.FIB.S.B
3 FILLER      BIT(2),%
```

```
  3 RELATIVE_RECORD_INDEX BIT(20),
                  %%% END OF STANDARD FIB %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 NDL_PART REMAPS UNIQUE  BIT(FIB_SIZE_NDL-FIB_COMMON_SIZE),
  3 NXT_FIB         BIT (24),
  3 BCK_FIB         BIT (24),
  3 QUE_NR          BIT (24),
  3 MCS_PRTCP       BIT (2),
    4 MCS_PRTCP_OUTPUT  BOOLEAN,
    4 MCS_PRTCP_INPUT   BOOLEAN,
  3 RMT_KEY         BOOLEAN,
  3 HDR             BOOLEAN,
  3 SIMPLE_HEADERS BOOLEAN,
  3 RSDNT           BIT (2),
  3 MCS_FL          BIT (8),
  3 LGL_STN_NR      BIT (10),
  3 RND_RBN         BIT (10),
  3 CUR_MAX_STN     BIT (10),
  3 REAL_MAX_STN    BIT (10),
  3 INPUT_COUNT     BIT (24),
  3 OUTPUT_COUNT    BIT (24),
  3 AUD_REC         BOOLEAN,
  3 MSG_ID_VALID    BOOLEAN,
  3 MSG_ID          BIT(20),% MSG.TIME, E.G.
  3 CREATED         BOOLEAN,
  3 JN_TBL_ADDR     WORD,
  3 LSN_LIST        ADDRESS,% POINTS TO C74.LSN.LIST.
  3 QUE_NAME     CHARACTER(48),
    4 QUE_NAME_1ST_3 CHARACTER(3),
    4 FILLER          CHARACTER(45),
  3 C74_ESI         CHARACTER (1),
  3 C74_EMI         CHARACTER (1),
 3 C74_EGI         CHARACTER (1),                  % EGI NOT IMPLEMENTED.
  3 LGL_STN_TBL BIT(11),
    4 LGL_STN    BIT (10),
    4 STN_DTCHD BOOLEAN,
              %%%% END NDL FIB %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 UFW_PART REMAPS UNIQUE  BIT(FIB_SIZE_UFW-FIB_COMMON_SIZE),
  3 UFW_FIRST_TIME_THRU    BOOLEAN,
  3 UFW_LAST_OP_READ       BOOLEAN,
  3 UFW_DUPLICATE          BOOLEAN,
  3 UFW_MATCH_FOUND        BOOLEAN,
  3 UFW_UPDATE_FLAG        BOOLEAN,
  3 UFW_FIRST_PASS         BOOLEAN,
  3 FILLER                     BOOLEAN,
  3 UFW_WRITE_ERR_REPORTED BOOLEAN,
  3 UFW_ACCESS_MODE        BIT(4),
  3 UFW_JOB_NUMBER         BIT(24),
  3 UFW_RECORD_ADDRESS     BIT(24),
  3 UFW_KEY_POINTER        ADDRESS,
```

```
 3 UFW_COMMUNICATE_WORKSPACE BIT(616),
   4 UFW_BINARY_SEARCH_ARGUEMENTS BIT(208),
   4   UFW_INTERFACE_PADS          BIT(96),
     5 UFW_FIRST_24                BIT(24),
     5 UFW_DESCR                   BIT(48),
      6 UFW_TYPE                   BIT(08),
      6 UFW_LENGTH                 BIT(16),
      6 UFW_ADDRESS                BIT(24),
     5 UFW_R REMAPS UFW_DESCR      BIT(48),
      6 UFW_SECOND_24              BIT(24),
      6 FILLER                     BIT(24),
     5 FILLER                      BIT(24),
   4 UFW_SAVE_STATE_AREA           BIT(312),
 3 UFW_GLOBAL_POINTER        ADDRESS,
 3 UFW_CURRENT_STRUCTURE     BIT(8),
 3 UFW_HEADER                BIT(24),
               %%% END UFW FIB %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 QUEUE_PART REMAPS UNIQUE  BIT(
        FIB_SIZE_QUEUE + 1023 * 24 - FIB_COMMON_SIZE),
 3 FILLER              BIT(44),% Q-DISK GOODIES
 3 Q_FAMILY_SIZE       BIT(24),% QFF # MEMBERS
 3 Q_READ_IO_DESC      BIT(24),%  REF-ADDR OR ZERO
 3 Q_WRITE_IO_DESC     BIT(24),%  REF-ADDR OR ZERO
 3 Q_FAMILY            BIT( 1),%   1 => QFF FIB
 3 Q_RECORD_COUNT      BIT(24),% # OF LOGICAL I/O-S
 3 Q_PTRS1             BIT(1024 * 24),%FIB.Q.FAMILY.SIZE
   4 QD_PTR(1024)      BIT(24),
 3 Q_PTRS2 REMAPS Q_PTRS1 BIT(1024 * 24),
   4 Q_PTR             BIT(24),
   4 FILLER            BIT(1023 * 24),
               %%% END NDL FIB %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 DIO_PART REMAPS UNIQUE  BIT(FIB_SIZE_DIAGNOSTIC - FIB_COMMON_SIZE),
 3 DTO_PCU             BIT (12),
                  % PORT CHANNEL AND UNIT OF PRIMARY UNIT
 3 DIO_IOAT_ADDR       ADDRESS,
                  % IOAT ENTRY FOR PRIMARY UNIT
 3 DIO_CONTROLS        BIT (3),
                  % NUMBER OF CONTROLS ASSIGNED TO THIS FILE
                  % (MAXIMUM OF 4)
 3 DIO_PC (4)          BIT (7),
                  % PORT AND CHANNEL OF EACH CONTROL
 3 DIO_CONTROL_OPENED BOOLEAN,
                  % ALL UNITS SHARING THE CONTROL HAVE BEEN OPENED
 3 DIO_SHARED_ACCESS   BOOLEAN,
                  % OTHER USERS MAY ACCESS THE UNIT(S)
 3 DIO_NAMED_FILE      BOOLEAN,
                  % CORRESPONDS TO PHYSICAL DISK FILE
 3 DIO_CHAIN           ADDRESS,
                  % ADDRESS OF FIRST DESCRIPTOR FOR THIS FILE
 3 DIO_HEADER          BIT (24),
                  % OFFSET IN DFH.DIR OF DISK FILE HEADER
 3 DIO_STARTING_ADDR   BIT (24),
                  % MINIMUM SECTOR THAT MAY BE ACCESSED
 3 DIO_STOPPING_ADDR   BIT (24),
                  % MAXIMUM SECTOR THAT MAY BE ACCESSED
 3 DIO_OPS_WAITING     BIT (4),
                  % NUMBER OF OPS MARKED AS WAITING THAT MUST COME
                  % COMPLETE BEFORE JOB CAN BE REINSTATED
 3 DIO_CHANNELS        BIT (4),
                  % NUMBER OF CHANNELS ASSIGNED FOR A DATACOM DEVICE
             %%%% END DIO FIB %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
 2 PORT_PART REMAPS UNIQUE    BIT (FIB_SIZE_PORT +
       (MAX_MAX_SUBPORTS - 1) * SUBPORT_ARRAY_SIZE - FIB_COMMON_SIZE),
   3 WAIT_SUBPORT          BIT (8),        % 0-REL. INDEX TO NON-ZERO SUBP.
   3 BROADCAST_SP          BIT (8),        % SUBPORT# FOR BROADCAST_WRITE.
   3 MAX_SUBPORTS          BIT (8),        % HOW MANY USER WANTED.
   3 LASTSUBPORT           BIT (8),        % LAST ONE READ FROM.
        % FOR AUTOMATIC WRITE BACK TO THE SAME ONE.
   3 ROUNDROBIN            BIT (8),        % USED FOR "READ ANY/NEXT".
        % TO SERVICE PORTS FAIRLY, WITHOUT STARVATION OF SOME BECAUSE
        % OF OTHERS THAT ALWAYS HAVE DATA IN THEM.
   3 INPUTCOUNT            WORD,           % FOR ALL SUBPORTS.
   3 OUTPUTCOUNT           WORD,           % FOR ALL SUBPORTS.
   3 PORT_KEYS             BOOLEAN,        % ALL I/O VERBS HAVE KEYS.
   3 PORT_CHANGEEVENT      BOOLEAN,
  3 EVENT_COUNT          BIT (8),        % NUM SUBPORTS WITH CAUSED CHG EV
   3 EVENT_SUBP_INDEX     BIT (8),        % USED BY CAUSE_SUBP_STATE_CHG,
        % GET_ATTR(CHANGEDSUBFILE), GET_ATTR(FILESTATE)
   3 SUBP(MAX_MAX_SUBPORTS) FIB_SUBPORT_ATTR_RECORD;
                  %%% END PORT_FIB %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 %%
 %%
CONSTANT                        % VALUES FOR SUBPORT_STATES:
    SPS_CLOSED                  =   @00@  % REFER TO HOST SERVICES USER
,   SPS_AWAITINGHOST            =   @01@  % INTERFACE SPEC 2373-2571
,   SPS_OPEN_PENDING            =   @02@  % SAME AS OFFERED
,   SPS_OPENED                  =   @03@
,   SPS_SHUTDOWN_IN_PROCESS     =   @04@
,   SPS_BLOCKED                 =   @05@
,   SPS_DEACTIVATION_PENDING    =   @07@
,   SPS_RMT_DEACTIVATED         =   @08@
 %
,   DEACT_RMT_CLOSE_V           =   @01@
,   DEACT_UNREACHABLE_V         =   @02@
,   DEACT_TIMEOUT_V             =   @03@
 %
,   SPS_CHNG_MASK               =   @800008@
;
    RECORD 1 PORT_STATE_VARIABLES BIT (FIB_SIZE_EXTRA_BNALIO)
          ,     3 PORT_STATE_VARS          BIT (200)
          ,          5 ACTUAL_MAX_MSG_SIZE     WORD
          ,          5 STATE                   WORD % PROTOCOL STATE
          ,          5 SEG_STATE               WORD
          ,          5 RETAINED                BOOLEAN % CLOSED W/ RETAIN.
          ,          5 IMPLIED                 BOOLEAN
          ,          5 TAG                     CHARACTER(8)
          ,          5 FRAMESIZE               WORD
          ,          5 B1000                   BOOLEAN %COOP HOST IS B1000
          ,          5 FILLER                  BIT(37)
          ,     3 IO_SUBPORT_STATE         BIT (4)
          ,     3 IN_Q                     ADDRESS
          ,     3 OUT_Q                    ADDRESS
          ,     3 ACTUAL_BUFFER_SIZE       WORD
          ,     3 AREAS_ALLOCATED_MASK     BIT(105)
          ;
```

APPENDIX H
TAPE ORGANIZATION

This appendix is in three parts. The first, Tape Labels, describes tape label formats acceptable to B 1000 systems. The second, Tape Format, describes the format of the information written on a magnetic tape. The third, Tape Status, describes the state of a tape after each of the various operations is performed.

TAPE LABELS

The MCP includes the capability to create and recognize two different forms of magnetic tape labels. The standard label format for the B 1000 system conforms to that specified in the publication entitled American National Standard Magnetic Tape Labels for Information Exchange, 1969, published by the American National Standards Institute (ANSI). These labels are commonly known as ANSI, Version 1 labels.

It should be noted that the standard label format for the system means that program file declaration requests for standard labels result in the writing of ANSI labels when the file is assigned to magnetic tape and opened output.

As of MCPII level 11.0.27, ANSI Version 3 tape labels also can be recognized.

ANSI labels as implemented on the B 1000 system contain several deviations from ANSI standards. The deviations insure compatibility with the B 5000/B 6000/B 7000 series systems. The most noteworthy deviation is the recording mode of the label itself. Unless the American Standard Code for Information Interchange (ASCII) is specifically requested by the user with the SN system command, the label is automatically written in EBCDIC.

ANSI Labels, though they are written when the file is opened output, are actually created on all magnetic tapes prior to that time. The SN (Serial Number) system command enables creation of the initial ANSI label on all tapes. The SN system command is described in section 5 of the *B 1000 Systems System Software Operation Guide, Volume 1.*

ANSI Tape Label Format

The ANSI tape label format as implemented consists of three physical blocks of tape, followed by a tape mark. The first of the three blocks, the Volume Header, has the following programmatic description.

```
RECORD
01 VOL_HEADER_RECORD     CHARACTER(80),
  02 FILLER              CHARACTER( 4),
  02 VOL_ID              CHARACTER( 6),
  02 ACCESSIBILITY       CHARACTER( 1),
  02 RFS                 CHARACTER(26),
    03 MFID              CHARACTER(17), % "0" IF NO MULTIPLE FILE ID
                                        % "X0" FOR 17 IF SCRATCH
                                        % "BACKUP" IF BACKUP
    03 SYS_SYMBOL        CHARACTER( 2), % "17"
    03 TAPE_TYPE         CHARACTER( 1), % 0 = SCRATCH
                                        % 1 = USER
                                        % 2 = BACKUP
                                        % 3 = LIBRARY
    03 FILLER            CHARACTER( 6),
  02 OWNER_ID            CHARACTER(14),
  02 FILLER              CHARACTER(28),
  02 VERSION             CHARACTER( 1); % 1 FOR THIS STANDARD
```

The second of the three physical blocks is Header One. The same format is also used for end of file and end of volume.

```
RECORD                                    % HDR1, EOV1, EOF1
01 HEADER1_RECORD          CHARACTER(80),
  02 FILLER                CHARACTER( 4),
  02 FILE_ID               CHARACTER(17),
  02 FILE_SET_ID           CHARACTER( 6),
  02 FILE_SECTION_NO       CHARACTER( 4),
  02 FILE_SEQ_NO           CHARACTER( 4),
  02 GENERATION_NO         CHARACTER( 4),
  02 GENERATION_VERSION_NO CHARACTER( 2),
  02 CREATION_DATE         CHARACTER( 6),
  02 EXPIRATION_DATE       CHARACTER( 6),
  02 ACCESSIBILITY         CHARACTER( 1),
  02 BLOCK_COUNT           CHARACTER( 6),%HDR1="000000", EOV, EOF = REA
  02 SYSTEM_CODE           CHARACTER(13),
  02 FILLER                CHARACTER( 7); %RFS
```

The third physical block, Header Two, is also used at end of file and end of volume.

```
RECORD                                  % HDR2, EOV2, EOF2
01 HEADER2_RECORD     CHARACTER(80),
  02 FILLER           CHARACTER( 4),
  02 RECORD_FORMAT    CHARACTER( 1), % F = FIXED, D = VARIABLE,
                                     % S = SPANNED, U = UNDEFINED
  02 BLOCK_LENGTH     CHARACTER( 5),
  02 RECORD_LENGTH    CHARACTER( 5),
  02 RESV_SYSTEM_USE  CHARACTER(35),
    03 DENSITY        CHARACTER( 1),%0 = 800, 1 = 556, 2 = 200, 3 = 1600
    03 SENTINAL       CHARACTER( 1),
    03 PARITY         CHARACTER( 1), %0 = ALPHA (EVEN), 1 = BINARY (ODD)
    03 EXT_FORM       CHARACTER( 1), % 0 = UNSPECIFIED
                                     % 1 = BINARY
                                     % 2 = ASCII
                                     % 3 = BCL
                                     % 4 = EBCDIC
    03 FILLER         CHARACTER(31),
  02 FILLER           CHARACTER(28); % RFS
```

The MCP writes labels in ANSI format whenever a file is opened output, and the LABEL.TYPE field in the FPB is set to zero. To write the old Burroughs format labels, the LABEL.TYPE in each file in the pertinent programs must be modified. This may be accomplished in any of the following four ways: (1) by recompilation, (2) by the use of a File Attribute communicate operation within the program, (3) by use of the MODIFY program control instruction (see section 4 of the *B 1000 Systems System Software Operation Guide, Volume 1),* or (4) by the use of a FILE card when the program is executed. Presently valid values for the LABEL.TYPE field are:

0 = ANSI
1 = Unlabelled
2 = Burroughs

The MCP writes tapemarks and ending labels on any output labeled tape that is not at beginning of tape (BOT) when a Clear/Start is done. This allows the user to read that tape and recover the data. There is one restriction. If the tape is to be read in reverse, the user must specify blocking information.

ANSI labels are also written as the standard label on 7-track tape. When this is done, the labels are written with translation to Burroughs Coded Language (BCL). Burroughs labels, when written to 7-track tape, are written in odd parity, with the EBCDIC/BCL translator enabled.

TAPE FORMAT

The format of the information that can be written on a magnetic tape is described in the following paragraphs. Both multifile tapes and multitape files are considered. A multifile tape is a labeled tape which contains zero or more data files. A multitape file is a data file which fills one or more tape reels and continues on another tape reel.

MULTIFILE TAPE

A multifile tape includes a label, optionally followed by data files, and a tape mark. The label contains a volume header and a header for the first data file, if any. A data file consists of a header, the data itself, and an end of file. The first data file uses the header in the label. Each of the additional data files has its own header. The last data file is followed by a tape mark, which causes the tape to be terminated with a double tape mark.

The label is an ANSI standard label with certain deviations required for compatibility with other Burroughs computer systems. The label consists of a volume header (VOL1) and a header.

The header consists of a header 1 record (HDR1), a header 2 record (HDR2), and a tape mark (TM).

The end of file consists of a tape mark, an end of file 1 record (EOF1), an end of file 2 record (EOF2), and a tape mark.

MULTITAPE FILE

A multitape file has the first part of a data file on one or more tape reels and the remaining part of the data file on a continuation tape reel.

The initial tape reel concludes with data and the end of tape photoreflective marker, followed by an end of volume and a tape mark. Valid data can be written beyond the end of tape mark. The tape is terminated with a double tape mark.

TAPE STATUS

The state of a magnetic tape after various operations are performed is described next. In the descriptions, which are numbered, a scratch ANSI label consists of a volume header block, a header 1 block, a header 2 block, and a tape mark.

1. Tape is initialized with the SN/SNL input message.

   The tape is:

   - Rewound.
   - Written with a scratch ANSI label.
   - Rewound.
   - Readied or locked depending on whether SN or SNL was used.

The tape contents and Read/Write head position are as follows:

* BOT VOL1 HDR1 HDR2 TM

* = R/W head position

2. Tape is opened for output for File 1.

The tape is:

- Rewound.
- Spaced over the volume 1 record.
- Rewritten for the header 1 record.
- Rewritten for the header 2 record.
- Rewritten for the tape mark.
- Accessible only to the program that opened it.

The tape contents and Read/Write head position are as follows:

BOT VOL1 HDR1 HDR2 TM *

* = R/W head position

3. Tape data is written for File 1.

The tape contents and Read/Write head position are as follows:

BOT VOL1 HDR1 HDR2 TM Data *

* = R/W head position

4. Tape is closed with no rewind.

The tape contents and Read/Write head position are as follows:

BOT VOL1 HDR1 HDR2 TM Data TM EOF1 EOF2 TM TM *

* = Read/Write head position

5. Tape is opened for output for File 2.

The tape contents and Read/Write head position are as follows:

BOT VOL1 HDR1 HDR2 TM Data TM EOF1 EOF2 TM . . . .
. . . . HDR1 HDR2 TM *

* = Read/Write head position

6. Tape data is written for File 2.

The tape contents and Read/Write head position are as follows:

BOT VOL1 HDR1 HDR2 TM Data TM EOF1 EOF2 TM . . . .
. . . . HDR1 HDR2 TM Data *

* = Read/Write head position

7. Data is written beyond EOF of the initial reel.

The tape contents and Read/Write head position are as follows:

BOT VOL1 HDR1 HDR2 TM Data TM EOF1 EOF2 TM . . . .
. . . . HDR1 HDR2 TM Data EOT Data TM EOV1 EOV2 TM *

* = Read/Write head position

8. Tape is positioned to read or write the continuation reel.

The tape contents and Read/Write head position are as follows:

BOT VOL1 HDR1 HDR2 TM *

* = Read/Write head position

9. Tape continues to be written for File 2.

The tape contents and Read/Write head position are as follows:

BOT VOL1 HDR1 HDR2 TM Data *

* = Read/Write head position

10. Tape is closed with release for File 2.

The tape contents and Read/Write head position are as follows:

BOT VOL1 HDR1 HDR2 TM Data TM EOF1 EOF2 TM TM *

* = Read/Write head position

APPENDIX I
RPG PROGRAM MEMORY DUMP

The following paragraphs describe how to obtain an RPG program memory dump and how to read the useful information contained in it. Example programs are included to illustrate the INVALID SUBSCRIPT and STACK OVERFLOW program aborts.

HOW TO OBTAIN AN RPG PROGRAM MEMORY DUMP

The RPG program memory dump is generated by entering either of the following system commands:

<job-number>DM

<job-number>DP

The DM system command causes a memory dump to be created and allows the program to continue executing.

The DP system command causes a memory dump to be created and discontinues the program.

The memory dump created by entry of the DM or DP system commands is a file with the name DUMPFILE/<integer>, where <integer> is a system-generated number. After the file is created, a human-readable listing of the memory-dump file may be obtained by entry of the following system command:

PM <integer>;

The PM system command causes the DUMP/ANALZER program to analyze the memory-dump file and produce the listing.

RPG DATA AREA DUMP INFORMATION

The RPG DATA AREA DUMP portion of the analyzed dump file begins with the following heading:

```
*** RPG DATA AREA DUMP ***
Next S-op at S=      11,D=      74
```

NEXT INSTRUCTION POINTER Information

The heading for the RPG data area dump portion of the memory dump includes a NEXT INSTRUCTION POINTER:

Next S-op at S= 11,D= 74

The dump also shows this pointer in the information that precedes the *** FULL DUMP ANALYSIS *** portion of the memory dump printout, following the job status information. Following are two examples, the first from a STACK OVERFLOW dump and the second from an INVALID SUBSCRIPT dump:

INVALID SUBSCRIPT: S=11, D=74 (@00B@,@0004A@); DS or DP

STACK OVERFLOW: S=12. D=31 (@00C@,@0001F@); DS or DP

This information identifies the next instruction to be performed. (S means segment, D means offset.)

The LOGIC and XMAP compiler-directing options must be specified in order to locate which RPG source record is being processed. The XMAP information associates the segment-displacement information with a paragraph name, and the LOGIC information associates the paragraph name with the RPG source record.

CONTAINER SIZES Information

The CONTAINER SIZES information has the following format.

```
*** CONTAINER SIZES ***
          23  :COP TABLE ENTRY
          11  :DATA DISPLACEMENT:
           8  :DATA LENGTH:
           6  :COP INDEX:
          12  :BRANCH DISPLACEMENT
```

INDICATORS SET Information

The INDICATORS SET information shows which indicators in the RPG program were ON when the memory dump was generated. The format of the INDICATORS SET information follows.

```
*** INDICATORS SET ***
          01
          10
          20
          LO
```

CURRENT OPERAND (COP) TABLE Information

The CURRENT OPERAND (COP) TABLE information lists the table address, COP index, data type, address (segment, displacement), digit length, and the data for each field name used in the RPG program. Note that some field names are used internally by the RPG cycle and are not available for use by the RPG programmer. The following is the format of the COP table information.

```
        *** CURRENT OPERAND (COP) TABLE ***
 TABLE     COP    DATA     ADDRESS      DIGIT
ADDRESS  INDEX   TYPE    (SEG,DISP)   LENGTH  DATA
@000040@     1     UN        0,128         3  001
@000057@     2     UN        0,131         3  000
@00006E@     3     UN        0,134         3  001
@000085@     4     UN        0,137         2  01
@00009C@     5     UN        0,139         1  0
@0000B3@     6     UN        0,140         1  0
@0000CA@     7     UA        0,141        30  ????????}???
@0000E1@     8     UN        0,30          2  00
@0000F8@     9     UN        0,41          2  00
@00010F@    10     UN        0,321         3  000
@000126@    11     UN        0,324         2  00
@00013D@    12     UN        0,326         2  00
@000154@    13     UN        0,328         8  00000000
@00016B@    14     UN        0,336         8  00000000
@000182@    15     UN        0,64          6  000000
@000199@    16     UA        0,344       180  9JJJJJ

@0001B0@    17     UN        0,524         3  001
@0001C7@    18     UA        0,527       264  5EEEEEE
                                               22222C33333D44444E55555
@0001DE@    19     UN        0,791         3  011
@0001F5@    20     UN        0,794         3  000
@00020C@    21     SN        0,797         1  +0
@000223@    22     SN        0,799         4  +0000
@00023A@    23     SN        0,804         6  +000000
@000251@    24     SN        0,811         6  SUBSCRIPTED -- 1 FACTOR
                                                     (TABLE BOUND = 28)
                                                   FACTOR 1 = 7
                                           (1)  +111110
                                           (2)  +222220
                                           (3)  +333330
                                           (4)  +444440
                                           (5)  +555550
                                           ---
@00027F@    26     UN        0,846         4  0005
@000296@    27     SN        0,850         6  +000000
@0002AD@    28     UN        0,857         4  0000
@0002C4@    29     UN        0,861         4  0000
@0002DB@    30     UA        0,865       180  9JJJJJ
@0002F2@    31     SN        0,1045        1  +9
@000309@    32     UA        0,1047       12  JJJJJ
```

SUBROUTINE STACK Information

The SUBROUTINE STACK information lists the current nesting levels of the subroutine stack. This information is useful whenever a STACK OVERFLOW abort occurs in the RPG program. In general, increasing the value of the STACK compiler-directing option solves the STACK OVERFLOW program abort.

The following is an example of the SUBROUTINE STACK information when no subroutines have been entered.

```
        *** SUBROUTINE STACK ***
@0013AC@              :STACK BASE
@0013AC@              :STACK TOP
*** NO ENTRIES ON SUBROUTINE STACK ***
```

ANALYZING AN INVALID SUBSCRIPT PROGRAM ABORT

The example source program in figure I-1 and the MAP information generated in figure I-2 show a method of associating field names with COP index entries. The subscript field name in the source program is IX and the array name is ARY. The MAP information associates IX with COP index 31 and ARY with COP index 32. The DATA column in the CURRENT OPERAND (COP) TABLE presented earlier shows that the COP index 31 has a value of +9. Since the number of entries for ARY is 5, an attempt to reference an index value of 9 causes an INVALID SUBSCRIPT program abort.

```
00100 $ NAMES
00110 $ XREF
00120 $ MAP
00130 $ PARMAP
00140 $ LOGIC
00150 $ XMAP
00200H

00300FIN      IPE  1800  90            DISK         U
00400FLINE    O          132           PRINTER

00500E                     ARY         5  6 2

00600IIN      NS  01
00700I                                          1  90 RECORD
00800I                                          1   101X
00900I                                          2   7 ARDATA

01000C                     MOVE ARDATA    ARY,IX
01100C                     SETON                    102030
01200C                     SETOF                    30

013000LINE    D         01
014000                          RECORD    90
015000                          ARY      100
016000                          IX    X  110
----*----1----*----2----*----3----*----4----*----5----*----6----*----7-
```

Figure I-1. Source Program with Compiler-Directing Options

```
C O D E   S P A C E
-------------------

        CODE SEG   0 SIZE -   179 BYTES
        CODE SEG   1 SIZE -   115 BYTES
        CODE SEG   2 SIZE -    43 BYTES
        CODE SEG   3 SIZE -    41 BYTES
        CODE SEG   4 SIZE -    25 BYTES
        CODE SEG   5 SIZE -    16 BYTES
        CODE SEG   6 SIZE -    13 BYTES
        CODE SEG   7 SIZE -    12 BYTES
        CODE SEG   8 SIZE -     0 BYTES
        CODE SEG   9 SIZE -   179 BYTES
        CODE SEG  10 SIZE -   229 BYTES
        CODE SEG  11 SIZE -    34 BYTES
        CODE SEG  12 SIZE -     8 BYTES
        CODE SEG  13 SIZE -    77 BYTES
        CODE SEG  14 SIZE -    16 BYTES
                                -----
 CUMULATIVE CODE SEG SIZE -   987 BYTES

                  LARGEST CODE SEGMENT -   229 BYTES
                       CODE DICTIONARY -   150 BYTES
                                          -----
** MINIMUM CODE SPACE REQUIRED TO RUN -   379 BYTES **
```

Figure I-2. Output from MAP Compiler-Directing Option

ANALYZING A STACK OVERFLOW PROGRAM ABORT

Figure I-3, an RPG source program, is included to illustrate the STACK OVERFLOW program abort. Figure I-4 shows the SUBROUTINE STACK information generated from the analysis of the dump file after this abort. Note that the segment/displacement values are repetitions. This signifies that the program is looping as it keeps trying to perform a subroutine that is already nested.

By comparing the segment and displacement values with those generated in the XMAP information, and then comparing the associated paragraphs with the LOGIC information, the RPG source statements that caused the STACK OVERFLOW program abort to occur can be identified.

```
00100H
00200FIN      IPE  1800  90            DISK         U
00300IIN      NS  01
00400I                                          1  90 DUMMY
00500C                    EXSR MAIN
00600CSR         MAIN     BEGSR
00700CSR                  EXSR OTHER
00800CSR                  ENDSR
00900CSR         OTHER    BEGSR
01000CSR                  EXSR MAIN
01100CSR                  ENDSR
----*----1----*----2----*----3----*----4----*----5----*----6----*----7-
```

Figure I-3. Source Program for STACK OVERFLOW Program Abort

```
      *** SUBROUTINE STACK ***
@000A3A@:      2618  :STACK BASE
@000B70@:      2928  :STACK TOP
  *** SUBROUTINE STACK OVERFLOW ***
                      STACK
                     ADDRESS     SEGMENT   DISPLACEMENT
                    @000A3A@       10            25
                    @000A59@       12            25 -+
                    @000A78@       12            56  !   These
                    @000A97@       12            25  !   stack
                    @000AB6@       12            56  !   entries
                    @000AD5@       12            25  +<- repeat
                    @000AF4@       12            56  !   the same
                    @000B13@       12            25  !   displacement
                    @000B32@       12            56  !   address.
                    @000B51@       12            25 -+
```

Figure I-4. SUBROUTINE STACK from STACK OVERFLOW Abort

INDEX

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

Documentation Evaluation Form

Title: B 1000 Systems Memory Dump Analysis
Functional Description Manual

Form No: 1152055
Date: March 1985

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:

Burroughs Corporation
Corporate Documentation – West
1300 John Reed Court
City of Industry, CA 91745
U.S.A.