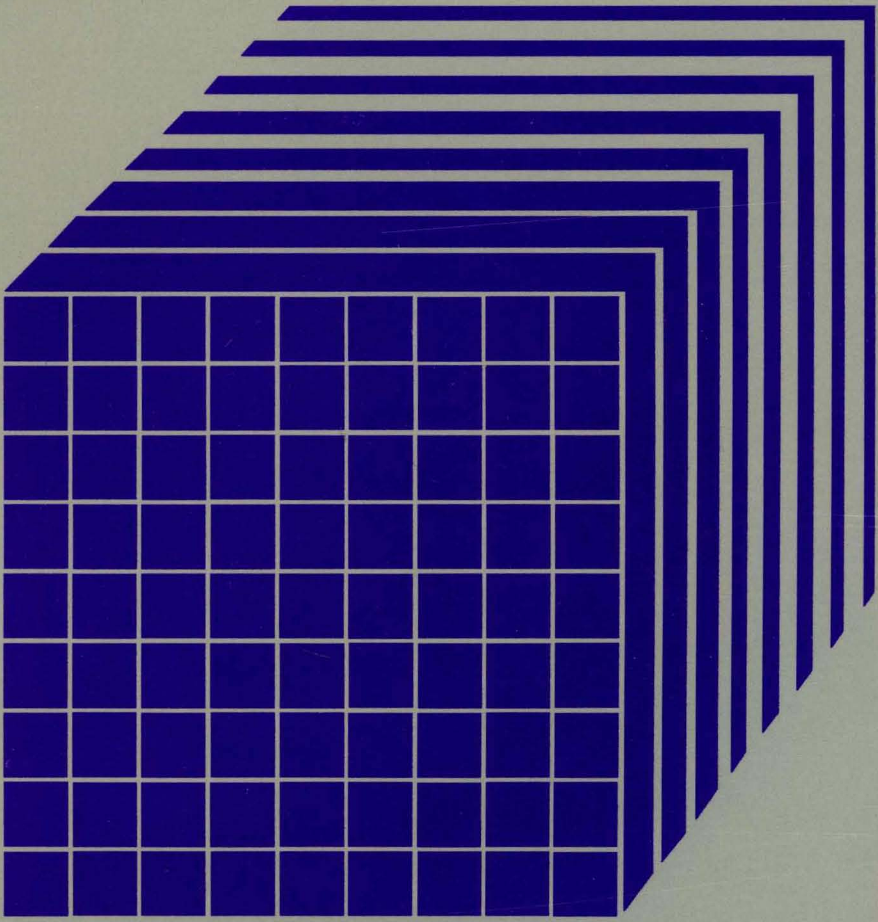
IBM

Virtual Machine

Diagnosis Guide

Release 5

LY24-5241-0



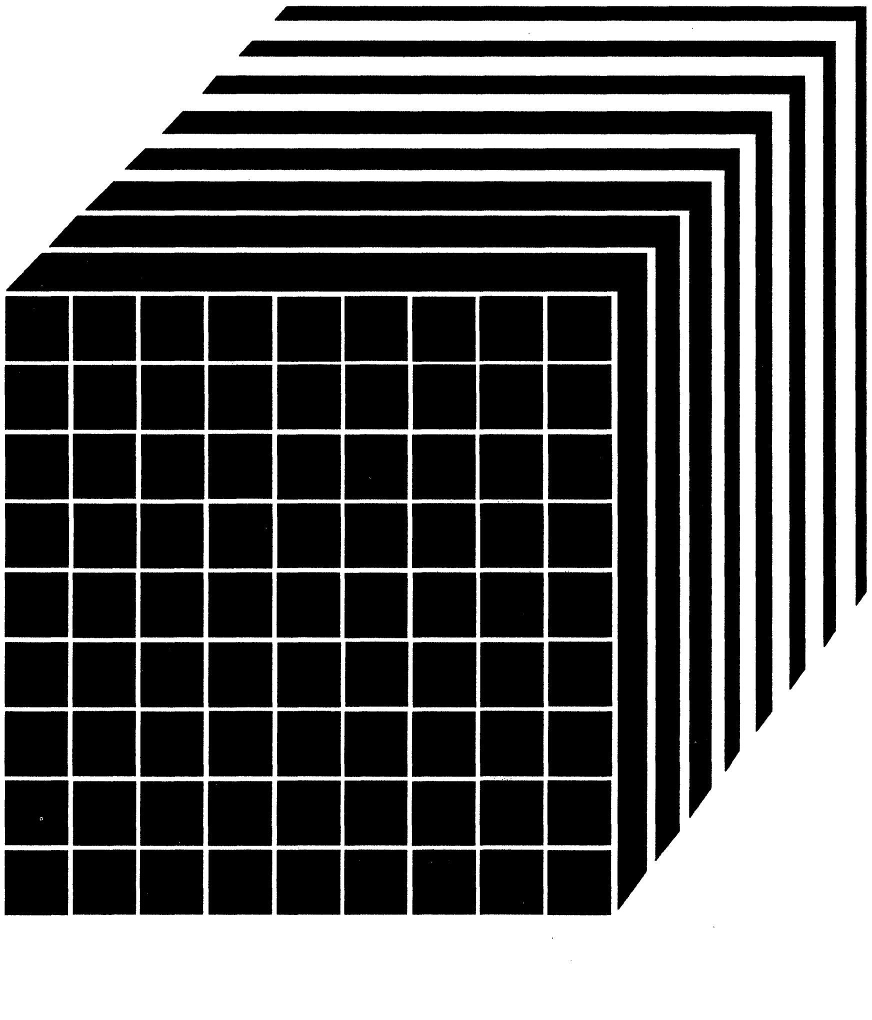
IBM

Virtual Machine

Diagnosis Guide

Release 5

LY24-5241-0



First Edition (December 1986)

This edition, LY24-5241-0, applies to Release 5 of IBM Virtual Machine/System Product (VM/SP), program number 5664-167; Release 5 of IBM Virtual Machine/System Product High Performance Option (VM/SP HPO), program number 5664-173; and to all subsequent releases of these products until otherwise indicated in new editions or Technical Newsletters. It contains material formerly included in the *VM/SP System Programmer's Guide* (discontinued after Release 4), *VM/SP HPO System Programmer's Guide* (discontinued after Release 4), *VM/SP Group Control System Guide* (discontinued after Release 4), *VM/SP Interactive Problem Control System Guide* (discontinued after Release 4), and *VM/SP CMS User's Guide*. Changes are continually made to the information contained herein; before using this publication in connection with the operation of IBM systems, consult the *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Summary of Changes

A cumulative Summary of Changes begins on page 365.

Changes and additions to the text and illustrations are indicated by a vertical bar to the left of the change.

Ordering Publications

Publications are not stocked at the address given below; requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication; if the form has been removed, comments may be addressed to IBM Corporation, Information Development, Dept. G60, P.O. Box 6, Endicott, New York, U.S.A. 13760. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever.

The form for readers' comments provided at the back of this publication may also be used to comment on the VM/SP and VM/SP HPO online HELP facility.

Preface

This is a guide to identify, report, solve, and collect information about problems in the Virtual Machine/System Product (VM/SP), program number 5664-167, and Virtual Machine/System Product High Performance Option (VM/SP HPO), program number 5664-173. In this manual, references to VM encompass both VM/SP and VM/SP HPO. It is intended for system programmers, system analysts, and others who know assembler language and have experience with programming concepts and techniques.

This manual is one of a set of reference manuals for VM/SP or VM/SP HPO system programmers. Other books in the set include:

- *VM/SP CP for System Programming* or *VM/SP HPO CP for System Programming*
- *VM/SP CMS for System Programming*
- *VM/SP Group Control System Command and Macro Reference*
- *VM/SP Transparent Services Access Facility Reference*
- *VM System Facilities for Programming.*

This manual consists mostly of material extracted from the:

- *VM/SP System Programmer's Guide* (discontinued after Release 4)
- *VM/SP HPO System Programmer's Guide* (discontinued after Release 4)
- *VM/SP Group Control System Guide* (discontinued after Release 4)
- *VM/SP Interactive Problem Control System Guide* (discontinued after Release 4)
- *VM/SP CMS User's Guide.*

This *VM Diagnosis Guide* consists of:

- **Chapter 1. Introduction to Debugging**, containing an overview of the debugging environment.
- **Chapter 2. Debugging the Virtual Machine**, containing a description of commands used to display or dump data, set and query system features, trace events, and alter storage.
- **Chapter 3. Debugging CP**, containing a description of commands and macros used to debug CP problems.

- **Chapter 4. Debugging CMS**, containing a description of commands used to debug CMS.
- **Chapter 5. Debugging GCS**, containing a description of the Tracing and Dumping facilities used to debug GCS problems.
- **Chapter 6. Debugging TSAF**, contains a description of methods used to collect information to debug TSAF.
- **Chapter 7. Debugging Using IPCS** , containing a description of the procedures and operations of IPCS.
- **Appendix A. Using DUMPSCAN Subcommands**, containing a description of all of the DUMPSCAN subcommands.
- **Appendix B. DUMPSCAN Scroll Interface**, containing the seven IPCS variables used to perform special scrolling.
- **Appendix C. IPCS SVC 199 Services**, containing a description of the IPCS communication facility.
- **Appendix D. Control Registers**, containing a description of the control register allocation and assignments.
- **Appendix E. Stand-Alone Dump Formats**, containing a description of the stand-alone dump tape format, DASD format, and printer format.
- **Appendix F. Converting Symptom Summary and Dump Files**, containing a description of the CONVERT command used to convert VM/370 IPCS symptom summary and dump files to a format required by VM/SP IPCS.
- **Appendix G. IPCS Interface Files**, containing a list of files that must be available to IPCS for successful dumps.
- **Summary of Changes**, is a cumulative summary of the changes to this manual for Release 5.
- **Glossary of Terms and Abbreviations**, explains or defines the terms, acronyms, and abbreviations that appear in this manual.
- **Bibliography**, lists prerequisite and corequisite publications.
- **Index**, lists topics alphabetically and points to the pages where they are discussed.

Contents

Figures

Chapter 1. Introduction to Debugging

Introduction to Debugging

The VM/SP and VM/SP HPO program products manage the resources of a single computer such that multiple computing systems appear to exist. Each "virtual computing system", or virtual machine, is the functional equivalent of an IBM System/370. Therefore, the person trying to determine the cause of a VM software problem must consider three separate areas:

1. The Control Program (CP), which controls the resources of the real machine
2. The virtual machine operating system running under the control of CP, such as CMS, RSCS, OS, DOS, GCS, or TSAF
3. The problem program, which executes under the control of a virtual machine operating system.

Information explaining how to debug CP or CMS is contained in this book; information explaining how to debug applications programs is in "Commands that Trace Events in Virtual Machines" on page 48. For information explaining how to use Interactive Problem Control System (IPCS) for debugging, refer to Chapter 7, "Debugging Using IPCS" on page 197. Refer to Chapter 5, "Debugging GCS" on page 153 for information on GCS and Chapter 6, "Debugging TSAF" on page 185 for information on TSAF.

If a problem is caused by a virtual machine operating system (other than VM), refer to the publications pertaining to that operating system for specific information.

If it becomes necessary to apply a Program Temporary Fix (PTF) to a component of VM/370 or VM/SP, refer to the *VM/SP Installation Guide* or the *VM/SP HPO Installation Guide* for information on applying PTFs.

How To Start Debugging

Before you can correct any problem, you must recognize that one exists. Next, you must identify the problem, collect information, and determine the cause so that the problem can be fixed. When running VM/SP or VM/SP HPO, you must also decide whether the problem is in CP, the virtual machine, or the problem program.

A good approach to debugging is:

1. Recognize that a problem exists.
2. Identify the problem type and the area affected.
3. Analyze the data you have available, collect more data if you need it, then isolate the data that pertains to your problem.
4. Determine the cause of the problem and correct it.

Does a Problem Exist?

There are four types of problems:

1. Loop
2. Wait state
3. Abend (abnormal end)
4. Incorrect results.

The most obvious indication of a problem is the abnormal termination of a program. Whenever a program abnormally terminates, a message is issued. Figure 1 lists the possible abend messages and identifies the type of abend for these messages.

Message	Type of Abend	Description

`(Alarm rings)`
`DMKDMP908I SYSTEM FAILURE; CODE - code PROCESSOR nn`
	CP	CP abend, system dumps to disk, printer, or tape. Restart is automatic.

`DMKCKP900W SYSTEM RECOVERY FAILURE; PROGRAM CHECK`
`DMKCKP901W SYSTEM RECOVERY FAILURE; MACHINE CHECK`
`DMKCKP902W SYSTEM RECOVERY FAILURE; FATAL I/O ERROR NUCLEUS AREA`
`DMKCKP902W SYSTEM RECOVERY FAILURE; FATAL I/O ERROR WARM AREA`
`DMKCKF922W System recovery failure; invalid spooling data`
`DMKCKH910W SYSTEM RECOVERY FAILURE; INVALID WARM START AREA`
`DMKCKH911W SYSTEM RECOVERY FAILURE; WARM START AREA FULL`
	CP	If the checkpoint program encounters a program check, a machine check, a fatal I/O error, or an error relating to a certain warm start area or warm start data conditions, a message is issued indicating the error and CP enters the wait state with code 007 in the PSW.

`DMKCKT903W System recovery failure; volid` <u>`volid`</u> `allocation area cylinder` <u>`cylinder`</u>
`DMKCKT912W System recovery failure; volid` <u>`volid`</u> `not mounted`
`DMKCKV912W System recovery failure; volid` <u>`volid`</u> `not mounted`
`DMKCKS915E Permanent I/O error on checkpoint area`
`DMKCKT916E Error allocating spool file buffers`
`DMKCKV916E Error allocating spool file buffers`
`DMKCKV917E Checkpoint area invalid; clear storage and cold start`
	CP	If the checkpoint start program encounters a severe error, a message is issued indicating the error and CP enters the wait state with code 00E in the PSW.

Figure 1 (Part 1 of 4). Abend Messages

Message	Type of Abend	Description
DMKWRM921W System recovery failure; unrecoverable I/O error DMKWRM903W System recovery failure; volid _volid_ allocation error cylinder _cylinder_ DMKWRM903W System recovery failure; volid _volid_ allocation error page _page_ DMKWRM904W System recovery failure; invalid warm start data DMKWRM912W System recovery failure; volid _volid_ not mounted DMKWRM920W No warm start data; checkpoint start for retry	CP	If the warm start program encounters a severe error, a message is issued indicating the error and CP enters the wait state with code 009 in the PSW.
DMKDMP908I SYSTEM FAILURE, CODE - code PROCESSOR nn DMKCKP960I System warm start data saved DMKCKP961W System shutdown complete	CP	CP abend, system dumps to disk, printer, or tape. The system stops; the operator must IPL the system to start again.
Optional Messages: DMKDMP905W SYSTEM DUMP FAILURE; PROGRAM CHECK DMKDMP906W SYSTEM DUMP FAILURE; MACHINE CHECK DMKDMP907W SYSTEM DUMP FAILURE; FATAL I/O ERROR	CP	If the dump program encounters a program check, a machine check, or a fatal I/O error, a message is issued indicating the error. CP enters the wait state with code 003 in the PSW. If the dump cannot find a defined dump device and if no printer is defined for the dump, CP enters a disabled wait state with code 004 in the PSW.
DMKMCH610W MACHINE CHECK; SUPERVISOR DAMAGE cpuid DMKMCT610W MACHINE CHECK; SUPERVISOR DAMAGE cpuid	CP	The machine check handler encountered an unrecoverable error with CP.
DMKMCH611W MACHINE CHECK; SYSTEM INTEGRITY LOST cpuid DMKMCT611W MACHINE CHECK; SYSTEM INTEGRITY LOST cpuid	CP	The machine check handler encountered an error that cannot be diagnosed; system integrity, at this point, is not reliable.
DMKMCH612W MACHINE CHECK; TIMING FACILITIES DAMAGE	CP	An error has occurred in the timing facilities. Probable hardware error.
DMKMCT620I MACHINE CHECK; ATTACHED PROCESSOR NOT BEING USED	CP	A malfunction alert, clock error or instruction processing error occurred on the attached processor. The system continues to run in uniprocessor mode.

Figure 1 (Part 2 of 4). Abend Messages

Message	Type of Abend	Description
`DMKMCH622W MACHINE CHECK; MULTIPLE CHANNEL ERRORS` `DMKACR622W MACHINE CHECK; MULTIPLE CHANNEL ERRORS`	CP	On a 303x processor, an error affecting one or more channels in a channel group has occurred. CP enters a disabled wait state with code 001 in the PSW.
`DMKCCH603W CHANNEL ERROR` `DMKACR603W CHANNEL ERROR`	CP	There was a channel check condition from which the channel check handler could not recover. CP enters the wait state with code 002 in the PSW.
`DMKOPE955W INSUFFICIENT STORAGE FOR VM/SP`	CP	The generated system requires more real storage than is available. CP enters the disabled wait state with code 00D in the PSW.
`DMKMCH622W MACHINE CHECK; MULTIPLE CHANNEL ERRORS`	CP	There was a group error machine check from which the machine check handler could not recover. CP enters a wait state with code 001 in the PSW.
`DMKMCH290E CP DISABLED WAIT; PMA GUEST MCT GIVEN CONTROL DUE TO xxx` **(This message will appear only if you are running a VM/SP HPO preferred machine assist guest.)**	CP	While preparing to load a disabled wait PSW, CP recognized that a V=R user has the PMA option. CP gives the user control in native mode in such a way that control cannot be passed back to VM/SP HPO. When convenient, operator can shut down MVS and re-IPL CP.
`DMSABN148T System abend` <u>`xxx`</u> `called from` <u>`vstor`</u>	CMS	CMS abend, system will accept commands from the terminal. Enter: `#CP VMDUMP 0-END FORMAT CMS DSS` to create a dump spool file in your reader. Re-IPL CMS and then enter: `IPCSDUMP` to format the dump.
CSIABD232E Abend *xxx-yyy* occurred during abend ESTAE processing	GCS	During the ESTAE abend processing an abend occurred.
CSIABD233E Abend *xxx-yyy* occurred during abend TASKEXIT processing	GCS	During the TASKEXIT abend processing an abend occurred.

Figure 1 (Part 3 of 4). Abend Messages

Introduction to Debugging

Message	Type of Abend	Description
CSIABD234E Abend *xxx-yyy* occurred during abend Resource Manager processing	GCS	Specified ABEND occurred while processing a GCS resource. Termination processing completed but all resources may not be cleaned up. Check dump for more details.
CSIABD235E Abend *xxx-yyy* occurred during abend internal processing	GCS	Specified ABEND occurred during the processing of another ABEND. Termination processing did not complete. Check dump for more details.
Others Refer to OS and DOS publications for the abnormal termination messages.	Other	When OS or DOS abnormally terminates on a virtual machine, the message issued and the dumps taken are the same as they would be if OS or DOS abnormally terminated on a real machine.

Figure 1 (Part 4 of 4). Abend Messages

Note: For TSAF abends see the *VM/SP System Messages and Codes* or *VM/SP HPO System Messages and Codes.*

Another obvious indication of a problem is unexpected output. If your output is missing, incorrect, or in a different format than expected, some problem exists.

Unproductive processing time is another symptom of a problem. This problem is not easily recognized, especially in a time-sharing environment.

Identifying the Problem

Two types of problems are easily identified: abnormal termination is indicated by an error message, and unexpected results become apparent once the output is examined. The looping and wait state conditions are not as easily identified.

When using VM/SP or VM/SP HPO, you are normally sitting at a terminal. You may have a looping condition if your program takes longer to execute than you anticipated. Also, check your output. If the number of output records or print lines is greater than expected, the output may really be the same information repeated many times. Repetitive output usually indicates a program loop.

Another way to identify a loop is to periodically examine the current PSW. If the PSW instruction address always has the same value, or if the instruction address has a series of repeating values, the program probably is looping.

The wait state is also difficult to recognize when at the terminal. If your program is taking longer than expected to execute, the virtual machine may be in a wait state. Issue the command:

```
DISPLAY PSW
```

to display the current PSW on the terminal. Check the wait bit (bit 14) in the PSW to determine if you are in a wait state.

You could also periodically issue the command:

```
#CP INDICATE USER
```

to display the execution characteristics of the program in terms of resources used. Compare the following resources:

- SIO, which is the total number of nonspooled I/O requests issued
- READS, which is the total number of page reads that have occurred
- WRITE, which is the total number of pages written.

When these resources don't change, the wait state probably exists.

Figure 2 on page 8 helps you to identify problem types and the areas where they may occur.

Analyzing the Problem

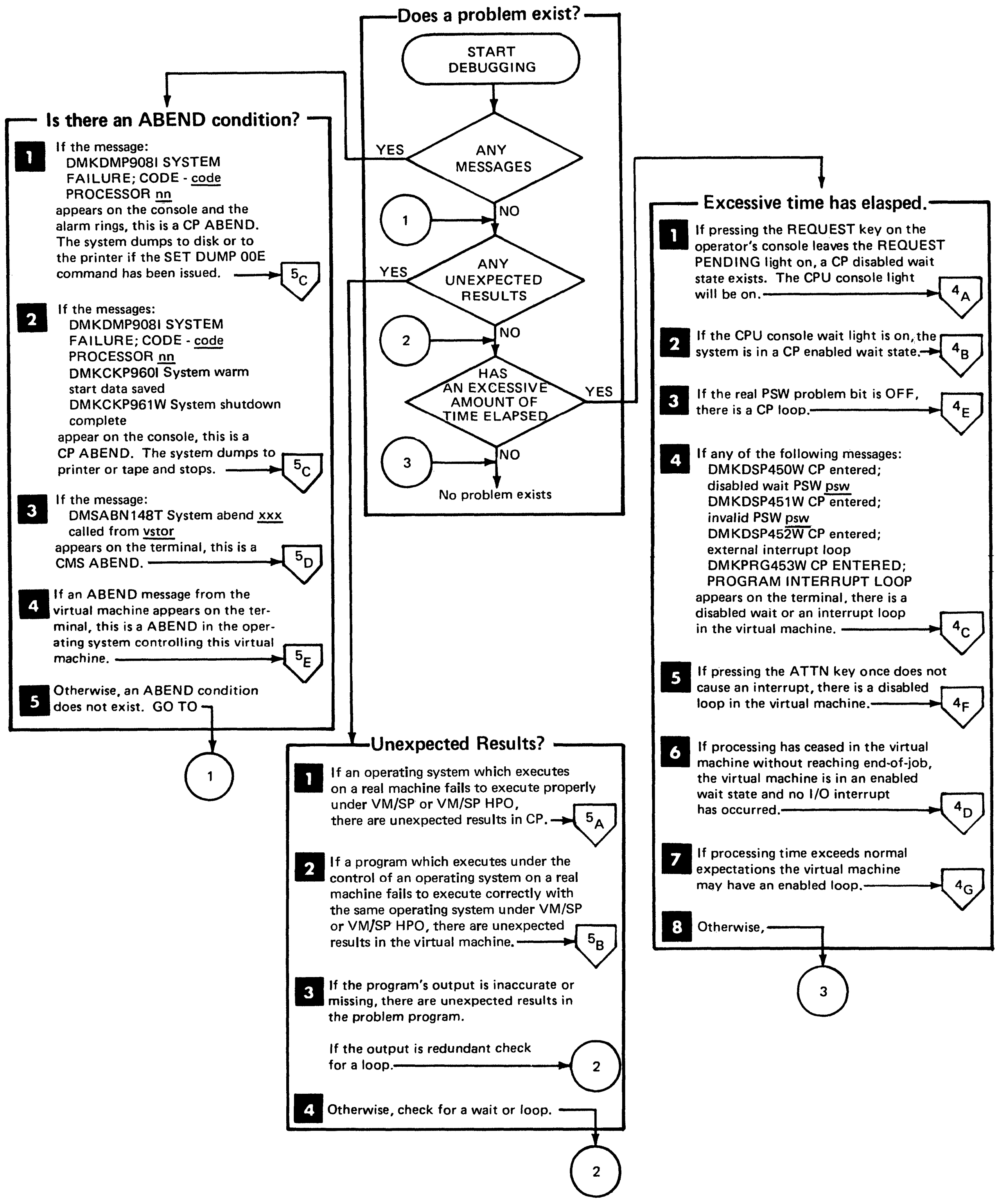
Once the type of problem is identified, its cause must be determined. There are recommended procedures to follow. These procedures are helpful, but do not identify the cause of the problem in every case. Be resourceful. Use whatever data you have available. If the cause of the problem is not found after the recommended debugging procedures are followed, it may be necessary to undertake the tedious job of desk-checking.

The section "How To Use VM/SP or VM/SP HPO Facilities To Debug" on page 16 describes procedures to follow in determining the cause of various problems that can occur in CP or in the virtual machine. See "Commands that Trace Events in Virtual Machines" on page 48 for information on using VM/SP facilities to debug a problem program.

If it becomes necessary to apply a PTF to a VM/SP or VM/SP HPO component, refer to the *VM/SP Installation Guide* or the *VM/SP HPO Installation Guide* for detailed information on applying PTFs. Figure 3 on page 9, Figure 4 on page 10, and Figure 5 on page 11 summarize the debugging process from identifying the problem to finding the cause.

Problem Type	Where Abend Occurs	Distinguishing Characteristics
Abend	CP abend CMS abend GCS abend TSAF abend	For a complete discussion of reasons for abends and system programmer's actions, see the CP, CMS, GCS, and TSAF abend codes charts in *VM/SP System Messages and Codes* or *VM/SP HPO System Messages and Codes.*
	Virtual machine abend (other than CMS)	When OS or DOS abnormally terminates on a virtual machine, the messages issued and the dumps taken are the same as they would be if OS or DOS abnormally terminated on a real machine. CP may terminate or reset a virtual machine if a nonrecoverable channel check or machine check occurs in that virtual machine. The system operator will receive a message at the processor console. Also, the virtual user will be notified by a message that his virtual machine was terminated or reset.
Unexpected Results	CP	If an operating system, other than CMS, executes properly on a real machine, but not properly with CP, a problem exists. Inaccurate data on disk or system files (such as spool files) is an error.
	Virtual machine	If a program executes properly under the control of a particular operating system on a real machine, but does not execute correctly under the same operating system with CP, a problem exists.
Wait	CP	For a complete discussion of CP, and loader wait state codes, see *VM/SP System Messages and Codes* or *VM/SP HPO System Messages and Codes.*
Loop	CP disabled loop	The processor console wait light is off. The problem state bit of the real PSW is off. No I/O interrupts are accepted.
	Virtual machine disabled loop	The program is taking longer to execute than anticipated. Signaling attention from the disabled loop terminal does not cause an interrupt in the virtual machine. The virtual machine operator cannot communicate with the virtual machine's operating system by signalling attention.
	Virtual machine enabled loop	Excessive processing time is often an indication of a loop. Use the CP QUERY TIME command to check the elapsed processing time. In CMS, the continued typing of the blip characters indicates that processing time is elapsing. If time has elapsed, periodically display the virtual PSW and check the instruction address. If the same instruction, or series of instructions, continues to appear in the PSW, a loop probably exists.

Figure 2. VM/SP Problem Types



Is there an ABEND condition?

1 If the message:
DMKDMP908I SYSTEM FAILURE; CODE - code PROCESSOR nn
appears on the console and the alarm rings, this is a CP ABEND. The system dumps to disk or to the printer if the SET DUMP 00E command has been issued. → 5C

2 If the messages:
DMKDMP908I SYSTEM FAILURE; CODE - code PROCESSOR nn
DMKCKP960I System warm start data saved
DMKCKP961W System shutdown complete
appear on the console, this is a CP ABEND. The system dumps to printer or tape and stops. → 5C

3 If the message:
DMSABN148T System abend xxx called from vstor
appears on the terminal, this is a CMS ABEND. → 5D

4 If an ABEND message from the virtual machine appears on the terminal, this is a ABEND in the operating system controlling this virtual machine. → 5E

5 Otherwise, an ABEND condition does not exist. GO TO → 1

Unexpected Results?

1 If an operating system which executes on a real machine fails to execute properly under VM/SP or VM/SP HPO, there are unexpected results in CP. → 5A

2 If a program which executes under the control of an operating system on a real machine fails to execute correctly with the same operating system under VM/SP or VM/SP HPO, there are unexpected results in the virtual machine. → 5B

3 If the program's output is inaccurate or missing, there are unexpected results in the problem program.

If the output is redundant check for a loop. → 2

4 Otherwise, check for a wait or loop. → 2

Excessive time has elasped.

1 If pressing the REQUEST key on the operator's console leaves the REQUEST PENDING light on, a CP disabled wait state exists. The CPU console light will be on. → 4A

2 If the CPU console wait light is on, the system is in a CP enabled wait state. → 4B

3 If the real PSW problem bit is OFF, there is a CP loop. → 4E

4 If any of the following messages:
DMKDSP450W CP entered; disabled wait PSW psw
DMKDSP451W CP entered; invalid PSW psw
DMKDSP452W CP entered; external interrupt loop
DMKPRG453W CP ENTERED; PROGRAM INTERRUPT LOOP
appears on the terminal, there is a disabled wait or an interrupt loop in the virtual machine. → 4C

5 If pressing the ATTN key once does not cause an interrupt, there is a disabled loop in the virtual machine. → 4F

6 If processing has ceased in the virtual machine without reaching end-of-job, the virtual machine is in an enabled wait state and no I/O interrupt has occurred. → 4D

7 If processing time exceeds normal expectations the virtual machine may have an enabled loop. → 4G

8 Otherwise, → 3

Figure 3. Does a Problem Exist?

Debug Procedures for a Wait

4A — **CP Disabled Wait**

1. Use ALTER/DISPLAY console mode (if available), to display real PSW and CSW. Also, display general and extended control registers and storage locations X'00' - X'10'.
2. **Force a SYSTEM RESTART (If not successful, do Stand-Alone Dump) to cause a CP ABEND dump to be taken. This automatically re-IPL's CP, if dumping to disk.**

4B — **CP Enabled Wait**

1. Force a SYSTEM RESTART (if not successful, do Stand-Alone Dump) to cause a CP ABEND dump to be taken.
2. Use the dump to check the status of each VMBLOK. Also, check RCHBLOK, RCUBLOK, and RDEVBLOK for each device.

4C — **Virtual Machine Disabled Wait ***

1. Use CP commands (CMS users may use the CMS DEBUG command) to display the PSW, CSW, general registers, and control registers.
2. Use the CP DUMP or CP VMDUMP command (or CMS DUMP subcommand) to take a dump.

4D — **Virtual Machine Enabled Wait ***

1. **Take a dump using the CP DUMP or CP VMDUMP command, using the correct FORMAT option.**

Debug Procedures for a Loop

4E — **CP Loop**

1. Use ALTER/DISPLAY console mode (if available), to display real PSW. Also, display general and extended control registers and storage locations X'00'– X'10'.
2. Trace the instruction loop on the processor and force a SYSTEM RESTART (if not successful, do Stand-Alone Dump) to cause a CP ABEND dump to be taken.
3. Examine the CP internal trace table to see where the loop is.

4F — **Virtual Machine Disabled Loop***

1. Use the CP TRACE or CP PER command to trace the loop.
2. Display the general registers and control registers via the CP DISPLAY command.
3. Take a dump using the CP DUMP or CP VMDUMP command, using the correct FORMAT option.
4. Examine the source code.

4G — **Virtual Machine Enabled Loop***

1. Trace the loop, using CP TRACE or CP PER.
2. Display the PSW, general registers, and extended control registers.
3. **Take a dump using the CP DUMP or CP VMDUMP command, using the correct FORMAT option.**
4. Examine the source code.

*Applies to GCS and/or TSAF.

Figure 4. Debug Procedures for Waits and Loops

5A

Debug Procedures for Unexpected Results

Unexpected Results in CP

1 Check that the program is not violating any CP restrictions.

2 Check that the program and operating system running on the virtual machine are exactly the same as those that ran on the real machine.

3 Use the CP TRACE command to trace CCWs, SIOs, and interrupts. Look for an error in CCW translation or interrupt reflection.

4 If disk I/O error, use the CP DDR (DASD Dump Restore) program to print the contents of any disk.

5B

Unexpected Results in a Virtual Machine

1 Check that the program executing on the virtual machine is exactly the same as the one that ran on the real machine.

2 Make sure that operating system restrictions are not violated.

3 Use CP TRACE to trace all I/O operations.

5C

Debug Procedures for an ABEND

CP ABEND

1 Find out why CP abended. Examine the PROPSW, INTPR, SVCOPSW, and CPABEND fields in the PSA from the dump.

2 Identify the module that caused the ABEND. Examine the SAVEAREA, BALRSAVE, and FREESAVE areas of the dump.

3 If I/O operation, examine the real and virtual I/O control blocks.

5D

CMS ABEND

1 Determine reason for ABEND from code in ABEND message DMSABN148T.

2 Enter debug environment or CP console function mode to use the commands, to display the PSW, and to examine low storage areas:
LASTLMOD and LASTTMOD
LASTCMND and PREVCMND
LASTEXEC and PREVEXEC and DEVICE
Look at the last instruction executed.
Take dump if needed.

5E

Virtual Machine ABEND (other than CMS)*

1 Examine dump, if there is one.

2 Use CP commands to examine registers and control words.

3 Use CP TRACE or CP PER to trace the processing up to the point where the error occurred.

*Applies to GCS and/or TSAF.

Figure 5. Debug Procedures for Unexpected Results and an Abend

Data Needed Before Calling IBM for Assistance

Note: **This section contains general information for all VM/SP based operating systems.**

If you should need to call your IBM Support Center for assistance, it is very important for you to have the following information:

1. Problem Inquiry Data Sheet
2. List of all applied maintenance for the module(s) involved
3. Operator's console log
4. Verification that all known errors against the PUT have been applied
5. NUCMAP for the failing system.

Problem Inquiry Data Sheet

The Problem Inquiry Data Sheet must be accurately filled-in to ensure that you get the correct solution from IBM. It might be a good idea to make copies of the Problem Inquiry Data Sheet (see Figure 6 on page 15), to have blank sheets available in case you have to call IBM.

System Information: When completing the Problem Inquiry Data Sheet, the **q cplevel** command should be used to help you to determine:

- Operating system
- Release level
- Service level

of your system.

For example, if you were on a VM/SP system and you entered:

Q CPLEVEL

you could get something that looked like this:

```
VM/SP RELEASE 5, SERVICE LEVEL 501
GENERATED AT 07/01/86 11:33:48 EDT
IPL AT 07/06/86 15:08:31 EDT
```

Make sure that you record system information (first output line from the q cplevel command) on the Problem Inquiry Data Sheet.

CPU Information: The **q cpuid** command should be used to help you to determine what to enter for the CPU Serial on the Problem Inquiry Data Sheet.

If you entered:

Q CPUID

you could get something that looked like this:

```
CPUID = FF13066043810000
```

This is the 16-digit processor identification associated with the virtual machine. Ignore the FF. The ten digits that follow the FF are the CPU Serial (first six digits representing the processor identification number and the next four digits representing the processor model number). Ignore the last four digits of this 16-digit field.

Note: The system release level, service level, and CPU serial number could also be obtained via IPCS from the Problem Record or through the SYMP subcommand of DUMPSCAN. See Chapter 7, "Debugging Using IPCS" on page 197 for more information about IPCS and see "SYMP Subcommand" on page 317 for more information about SYMP.

Data Sheet Fields: The Problem Inquiry Data Sheet consists of the following fields:

Customer
Enter business name.

Date
Enter date.

Problem #
Enter the problem number that IBM will assign to you when you call.

Access Code
Enter the customer number that the IBM marketing representative gave to you.

CPU Serial
Enter the 10-digit number from using **q cpuid** command, as described above.

Severity
Enter 1, 2, 3, or 4. The severity codes mean:

1
You are unable to use the program, resulting in a critical impact on your operations.

2
You are able to use the program, but you are severely restricted.

3
You are able to use the program with limited functions which are not critical to overall operations.

4
You have found a way to circumvent the problem.

Output of Q CPLEVEL Command
Enter the system information exactly as displayed in the first line of output from the **q cplevel** command.

Failing Component
Enter suspected component where problem exists.

Problem/Inquiry Description
Enter reason for calling the IBM Support Center.

Keywords
Indicate words which may best describe the problem, using the provided checklist.

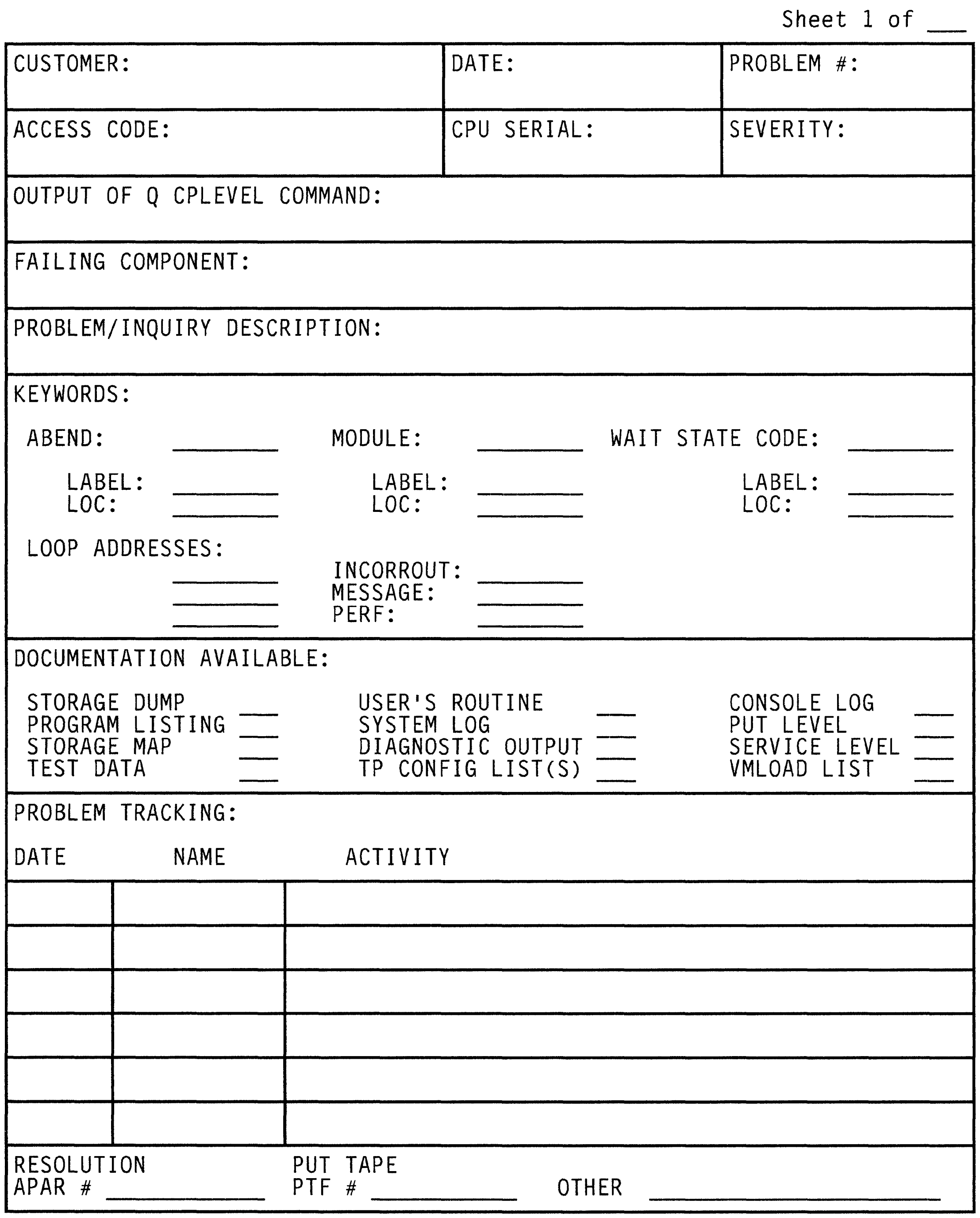
Documentation Available
Indicate available documentation, using the provided checklist.

Problem Tracking
Enter a log of your activity on the problem, including dates, names, and activity.

Resolution APAR #
Enter APAR number assigned to problem (if defect related).

PUT Tape PTF #
Enter PUT tape number on which the PTF for the resolution APAR resides.

Other
Enter other pertinent information to this problem.



Sheet 1 of ___

CUSTOMER:	DATE:	PROBLEM #:
ACCESS CODE:	CPU SERIAL:	SEVERITY:

OUTPUT OF Q CPLEVEL COMMAND:

FAILING COMPONENT:

PROBLEM/INQUIRY DESCRIPTION:

KEYWORDS:

```
 ABEND:      ________    MODULE:     ________   WAIT STATE CODE:  ________

    LABEL:   ________       LABEL:   ________            LABEL:   ________
    LOC:     ________       LOC:     ________            LOC:     ________

 LOOP ADDRESSES:
             ________    INCORROUT:  ________
             ________    MESSAGE:    ________
             ________    PERF:       ________
```

DOCUMENTATION AVAILABLE:

STORAGE DUMP	___	USER'S ROUTINE	___	CONSOLE LOG	___
PROGRAM LISTING	___	SYSTEM LOG	___	PUT LEVEL	___
STORAGE MAP	___	DIAGNOSTIC OUTPUT	___	SERVICE LEVEL	___
TEST DATA	___	TP CONFIG LIST(S)	___	VMLOAD LIST	___

PROBLEM TRACKING:

DATE	NAME	ACTIVITY

```
RESOLUTION             PUT TAPE
APAR # ____________    PTF # ___________     OTHER ______________________
```

Figure 6. Problem Inquiry Data Sheet. Use this sheet to collect pertinent data before calling IBM.

How To Use VM/SP or VM/SP HPO Facilities To Debug

Once the problem and the area where it occurs are identified, you can gather the information needed to determine the cause of the problem. The type of information you want to look at varies with the type of problem. The tools used to gather the information vary depending upon the area in which the problem occurs. For example, if the problem is a loop condition, you will want to examine the PSW. For a CP loop, you have to use the operator's console to display the PSW, but for a virtual machine loop you can display the PSW via the CP DISPLAY command.

The following sections describe specific debugging procedures for the various error conditions. The procedures tell you what to do and what debug tool to use. For example, the procedure may say dump storage using the CP DUMP command. The procedure does not tell you how to use the debug tool. Refer to "Summary of VM Debugging Tools" on page 32 and "Debugging Commands" on page 140 for a detailed description of each debug tool, including how to invoke it.

Abend

Five types of abnormal terminations (abend) can occur in VM/SP:

- CP
- CMS
- GCS
- TSAF
- Virtual machine.

The following descriptions provide guidelines for debugging each type of abend.

CP Abend

When CP abnormally terminates, a dump is taken. This dump can be directed to tape or printer, or dynamically allocated to a direct access storage device. The output device for a CP abend dump is specified by the CP SET DUMP command. See *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for a description of the SET DUMP command.

Use the dump to determine why CP terminated and then determine how to correct the condition. See "Reading CP Abend Dumps" on page 76 for detailed information on reading a CP abend dump. You can view the dump interactively, using IPCS.

Reason for the CP Abend: CP will terminate and take an abnormal termination dump under three conditions:

1. Program Check in CP

   Examine the PROPSW and INTPR fields in the Prefix Storage Area (PSA) to determine the failing module.

2. Module Issuing an SVC 0

   Examine the SVC old PSW (SVCOPSW) and abend code (CPABEND) fields in the PSA to determine the module that issued the SVC 0 and the reason it was issued.

   CPABEND contains an abnormal termination code. The first three characters identify the failing module (for example, abend code TRC001 indicates DMKTRC is the failing module).

3. Operator forcing a CP system restart on Processor Console

   Examine the old PSW at location X'08' to find the location of the instruction that was executing when the operator forced a CP system restart. The operator forces a CP system restart when CP is in a disabled wait state or loop. (Refer to your processor manual for the appropriate method to force a CP system restart.)

Note: The same conditions that cause an abnormal termination on a uniprocessor configuration cause an abnormal termination on an attached processor.

Examine Low Storage Areas: The information in low storage specifies the status of the system at the time CP terminated. Status information is stored in the PSA. You should be able to tell the module that was executing by looking at the PSA. Refer to the appropriate save area (SAVEAREA, BALRSAVE, or FREESAVE) to see how that module started to execute. The PSA is described in *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP.*

Examine the real and virtual control blocks to find the status of I/O operations. Figure 8 on page 73 shows the relationship of CP control blocks.

Examine the CP internal trace table. This table can be extremely helpful in determining the events that preceded the abend. See “CP Internal Trace Table” on page 72 for a description of how to use the trace table.

If you are using IPCS to view the dump, you can use the TRACE subcommand. For details, see “TRACE Subcommand” on page 324.

The values in the general purpose registers can help you to locate the current IOBLOK and VMBLOK and the save area. Refer to “Reading CP

Abend Dumps" on page 76 for detailed information on the contents of the general purpose registers.

If the program check old PSW (PROPSW) or the SVC old PSW (SVCOPSW) points to an address beyond the end of the resident nucleus, the module that caused the abend is a pageable module. Refer to "Reading CP Abend Dumps" to find out how to identify that pageable module. Use the CP load map that was created when the VM/SP system was generated to find the address of the end of the resident nucleus.

CMS Abend

When CMS abnormally terminates, any abend exit routines established via the ABNEXIT macro receive control. These exit routines allow you to bypass CMS abend recovery and continue processing elsewhere. If no routine exists or the exit routine returns to CMS, the following error message appears on the terminal:

```
DMSABN148T System abend xxx called from vstor
```

where *xxx* is the abend code and *vstor* is the address of the instruction causing the abend. The DMSABN module issues this message. Then, CMS waits for a command to be entered from the terminal.

Because CMS is an interactive system, you will probably want to use its debug facilities to examine status. You may be able to determine the cause of the abend without taking a dump.

The debug program is located in the resident nucleus of CMS and has its own save and work areas. Because the debug program itself does not alter the status of the system, you can use its options knowing that routines and data cannot be overlaid unless you specifically request it. Likewise, you can use the CP commands in debugging knowing that you cannot inadvertently overlay storage because the CP and CMS storage areas are completely separate.

Reason for the CMS Abend: First determine the reason CMS abnormally terminated. There are four types of CMS abnormal terminations:

1. Program Exception

   Control is given to the DMSITP routine whenever a hardware program exception occurs. If no SPIE or STAE exits have been specified, DMSITP issues the message:

   ```
   DMSITP141T exception exception occurred at vstor in
              routine routine
   ```

   and invokes DMSABN (the abend routine). The possible programming exceptions and related abend codes are listed in the *VM/SP System Messages and Codes* and *VM/SP HPO System Messages and Codes,* under 141T.

2. ABEND Macro

   Control is given to the DMSSAB routine whenever a user routine executes the ABEND macro. The abend code specified in the ABEND macro appears in the abnormal termination message DMSABN155T.

3. Halt Execution command (HX)

   Whenever the virtual machine operator signals attention and types HX, CMS terminates and types "CMS".

4. System Abend

   A CMS system routine can abnormally terminate by issuing the DMSABN macro. The first three hexadecimal digits of the system abend code appear in the CMS abend message, DMSABN148T. The format of the DMSABN macro is:

[*label*]	**DMSABN**	$\left\{\begin{matrix}\textit{code}\\ \textit{(reg)}\end{matrix}\right\} \left[\textbf{,TYPCALL=}\left[\begin{matrix}\underline{\textbf{SVC}}\\ \textbf{BALR}\end{matrix}\right]\right]$

where:

label
: is any valid Assembler language label.

code
: is the abnormal termination code (X'0' through X'FFF') that appears in the DMSABN148T system termination message.

(reg)
: is the register containing the abnormal termination code.

$\textbf{TYPCALL=}\left[\begin{matrix}\underline{\textbf{SVC}}\\ \textbf{BALR}\end{matrix}\right]$
: specifies how control is passed to the abnormal termination routine, DMSABN. Routines that do not reside in the nucleus should use TYPCALL=SVC to generate CMS SVC 203 linkage. Nucleus-resident routines should specify TYPCALL=BALR so that a direct branch to DMSABN is generated.

If a CMS SVC handler abnormally terminates, that routine can set an abend flag and store an abend code in NUCON (the CMS nucleus constant area). After the SVC handler has finished processing, the abend condition is recognized. The DMSABN abend routine types the abend message, DMSABN148T, with the abend code stored in NUCON.

What to do when CMS Abnormally Terminates: After an abend, two courses of action are available in CMS. In addition, by signalling attention, you can enter the CP command mode and use CP's debugging facilities.

Two courses of action available in CMS are:

1. Issue the DEBUG command and enter the debug environment. After using all the DEBUG subcommands that you wish, exit from the debug environment. Then, either issue the RETURN command to return to DMSABN so that abend recovery will occur, or issue the GO command to resume processing at the point the abend occurred.

   The most common problem you might encounter is an abnormal termination resulting from a program interruption. When a program running on a CMS virtual machine abnormally terminates (abends), you receive, at your terminal, the message:

```
| DMSITP141T exception exception occurred at vstor in routine routine
```

   and your virtual machine is returned to the CMS environment. From the message you can determine the types of program checks (operation, privileged operation, execution, protection, addressing, etc.) and, often, the instruction address in your program at which the error occurred.

   Sometimes this is enough information for you to correct the error in your source program, recompile it and attempt to execute it again.

   When this information does not immediately identify the problem in your program, you can begin debugging procedures using VM/SP. To access your program's storage areas and registers you can enter the command:

   DEBUG

   immediately after receiving the abend message. This command places your virtual machine in the debug environment.

   To check the contents of GPRs 0 through 15, issue the DEBUG subcommand:

   GPR 0 15

   If you want to look at only one register, enter:

   GPR 3

   You might also wish to check the PSW. Use the PSW subcommand:

   PSW

You can examine storage areas in your program using the X subcommand:

```
X 201AC 20
```

In this example, the subcommand requests a display of 20 bytes, beginning at location X'201AC' in your program. User programs executing in CMS are always loaded beginning at location X'20000' unless you specify a different address on the LOAD or FETCH command. To identify the virtual address of any instruction in a program, you only need to add X'20000' to the hexadecimal instruction address.

2. Issue a CMS command other than DEBUG, and the abend routine, DMSABN, performs its abend recovery and then passes control to the DMSINT routine to process the command just entered.

The abend recovery function:

1. Clears the console input buffer and program stack.
2. Terminates all VMCF activity.
3. Reinitializes the work area stack for reentrant CMS nucleus modules.
4. Reinitializes the SVC handler, DMSITS, and frees all stacked save areas.
5. Clears the auxiliary directories, if any. Invokes "FINIS * * *", to close all files, and to update the master file directory.
6. Frees storage, if the DMSEXT module is in virtual storage.
7. Zeroes out the MACLIB directory pointers.
8. Frees the CMS work area, if the CMS subset was active.
9. Frets the RLDDATA buffer, used by the CMS loader to retain relocation information for the GENMOD process, if it is still allocated.
10. Issues the STAE, SPIE, TTIMER, and STAX macros to cancel any outstanding OS exit routines. Frees any TXTLIB, MACLIB, or LINK tables.
11. Calls with a purge PLIST, all nucleus extensions that have the "SERVICE" attribute defined.
12. Drops all nucleus extensions that do not have the "SYSTEM" attribute. Also drops any nucleus extensions that are in type user storage.
13. Drops all SUBCOM SCBLOCKS that do not have the "SYSTEM" attribute.

14. Frees console path and device entry control blocks.

15. Drops all storage resident execs that do not have the "SYSTEM" attribute.

16. Clears all immediate commands that are not nucleus extensions with the "SYSTEM" attribute; returns all associated free storage.

17. Calls DMSCLN to zero out the userword of the SRPI command.

18. Calls DMSWITAB to delete all windows and vscreens that do not have the "SYSTEM" attribute.

19. Resets the storage keys for the whole virtual machine, except the nonshared pages, according to FREETAB. Saves the setting for KEYPROTECT.

20. Zeroes out all FCB, DOSCB, and LABSECT pointers.

21. Frees all storage of type user.

22. Restores the setting for KEYPROTECT.

23. Zeroes out all interrupt handler pointers in IOSECT.

24. Turns the SVCTRACE command off.

25. Closes the virtual punch and printer; closes the virtual reader with the HOLD option.

26. Reinitializes the VSE lock table used by CMS/DOS and CMS/VSAM.

27. Zeroes out all OS loader blocks, and frees the FETCH work area.

28. Cleans up the CMS IUCV environment based on the existence of the CMS id block.

29. Clears all ABNEXIT set and returns storage.

30. Computes the amount of system free storage that should be allocated and compares this amount with the amount of free storage actually allocated. Types a message to the user if the two amounts are unequal.

31. Issues a STRINIT and releases any pages remaining in the flush list via a call to DMSPAGFL, if all storage is accounted for.

After abend recovery has completed, control passes to DMSINT at entry point DMSINTAB to process the next command.

When the amount of storage actually allocated is less than the amount that should be allocated, the message

```
DMSABN149T nnn (HEX xxx) doublewords of system storage
           have been destroyed; re-IPL CMS
```

appears on the terminal. If the amount of storage actually allocated is greater than the amount that should be allocated, the message

```
DMSABN150W nnn (HEX xxx) doublewords of system storage
           were not recovered
```

A Debugging Procedure: When a CMS abend occurs, use the CP commands or DEBUG subcommands to examine the PSW and specific areas of low storage. For instructions on how to use the CP commands, see "Summary of VM Debugging Tools" on page 32 and "Commands that Trace Events in Virtual Machines" on page 48.

The following procedure may be useful in determining the cause of a CMS abend:

1. Display the PSW. (Use the CMS DEBUG PSW subcommand.) Compare the PSW instruction address with the current CMS load map trying to determine the module that caused the abend. The CMS storage-resident nucleus routines reside in fixed storage locations.

   Also check the interruption code in the PSW.

2. Examine areas of low storage. The information in low storage can tell you more about the cause of the abend:

Field	**Contents**
LASTLMOD	Contains the name of the last module loaded into storage via the LOADMOD command.
LASTTMOD	Contains the name of the last module loaded into the transient area.
LASTCMND	Contains the name of the last command issued from the CMS or XEDIT command line. If a command issued in a CMS EXEC abnormally terminates, this field contains the name of the command. When a CMS EXEC completes, this field contains the name 'EXEC'. EXEC 2 and System Product Interpreter do not update this field.
PREVCMND	Contains the name of the next-to-last command issued from the CMS or XEDIT command line. If a command issued in a CMS EXEC abnormally terminates, this field contains the name 'EXEC'. When a CMS EXEC completes, this field contains the last command issued from the CMS EXEC. EXEC 2 and System Product Interpreter do not update this field.

LASTEXEC	Contains the name of the last CMS EXEC procedure. EXEC 2 and System Product Interpreter do not update this field.
PREVEXEC	Contains the name of the next-to-last CMS EXEC procedure. EXEC 2 and System Product Interpreter do not update this field.
DEVICE	Identifies the device that caused the last I/O interrupt. The low storage areas examined depend on the type of abend.

3. Once you have identified the module that caused the abend, examine the specific instruction. Refer to the listing.

4. If you have not identified the problem at this time, take a dump by issuing the VMDUMP command. Refer to "Reading CMS Abend Dumps" on page 145 for information on reading a CMS dump. If you can reproduce the problem, try the CP or CMS tracing facilities.

GCS Abend

For information on GCS abends, see *VM/SP Group Control System Diagnosis Reference.*

| ### TSAF Abend

| For information on TSAF abends, see *VM/SP Transparent Services Access*
| *Facility Reference* and Chapter 6, "Debugging TSAF" on page 185.

Virtual Machine Abend (Other than CMS)

The abnormal termination of an operating system (such as OS or DOS) running under CP appears the same as termination of the operating system on a real machine. Refer to publications for that operating system for debugging information. However, all of the CP debugging facilities may be used to help you gather the information you need. Because certain operating systems (OS/VS1, OS/VS2, and DOS/VS) manage their virtual storage themselves, CP commands that examine or alter virtual storage locations should be used only in virtual=real storage space with OS/VS1, OS/VS2, and DOS/VS.

The VMDUMP command dumps virtual storage to a specified virtual machine's reader spool file. The IPCS component of VM/SP may be used to process the file created by the VMDUMP command. For details, see Chapter 7, "Debugging Using IPCS" on page 197.

If you choose to run a stand-alone dump program to dump the storage in your virtual machine, be sure to specify the NOCLEAR option (which is the default) when you issue the CP IPL command. At any rate, a portion of your virtual storage is overlaid by CP's virtual IPL simulation.

If the problem can be reproduced, it may be helpful to trace the processing using the CP TRACE or CP PER commands. Also, you can set address stops, and display and alter registers, control words (such as the PSW), and data areas. The CP commands can be very helpful in debugging because you can gather information at various stages in processing. A dump is static and represents the system at only one particular time. Debugging on a virtual machine can often be more flexible than debugging on a real machine.

VM/SP or VM/SP HPO may terminate or reset a virtual machine if a non-recoverable machine check occurs in that virtual machine. Hardware errors usually cause this type of virtual machine termination. The following message:

```
DMKMCH616I MACHINE CHECK; USER userid TERMINATED cpuid
```

appears on the processor console.

If the message:

```
DMKMCT621I  MACHINE CHECK; AFFINITY SET OFF
```

appears, then a machine check has occurred on the attached processor, and the attached processor is no longer being used. The virtual machine is placed into console function mode and can be made to continue processing on the main processor by the entry of a BEGIN command.

Channel checks no longer cause the virtual machine to be reset as they did in early releases of VM/370. If the problem appears to be associated with attempts to recover from a channel check, see the channel model-dependent functions described in the *VM/SP Planning Guide and Reference* or *VM/SP HPO Planning Guide and Reference.*

Unexpected Results

The type of errors classified as unexpected results vary from operating systems improperly functioning under CP to printed output in the wrong format.

Unexpected Results in CP

If an operating system executes properly on a real machine but does not execute properly with CP, a problem exists. Also, if a program executes properly under control of a particular operating system on a real machine but does not execute correctly under the same operating system with CP, a problem exists.

First, there are conditions (such as time-dependent programs) that CP does not support. Be sure that one of these conditions is not causing the unexpected results in CP. Refer to the *VM/SP Planning Guide and Reference* or *VM/SP HPO Planning Guide and Reference* for a list of the restrictions.

Next, be sure that the program and operating system running on the virtual machine are the same as those that ran on the real machine. Check for the same:

- Job stream
- Copy of the operating system (and program)
- Libraries.

If the problem still is not found, look for an I/O problem. Try to reproduce the problem, while tracing all Channel Command Words (CCWs), SIOs, and interrupts with the CP TRACE or CP PER commands. Compare the real and virtual CCWs from the trace. A discrepancy in the CCWs may indicate that one of the CP restrictions was violated, or that an error occurred in CP.

Unexpected Results in a Virtual Machine

When a program executes correctly under control of a particular operating system on a real machine but has unexpected results executing under control of the same operating system with VM/SP or VM/SP HPO, a problem exists. Usually you will find that something was changed. Check that the job stream, the operating system, and the system libraries are the same.

If unexpected results occur (such as TEXT records interspersed in printed output), you may wish to examine the contents of the system or user disk files. Non-CMS users may execute any of the utilities included in the operating system they are using to examine and rearrange files. Refer to the utilities publication for the operating system running in the virtual machine for information on how to use the utilities.

CMS users should use the DASD Dump/Restore (DDR) service program to print or move the data stored on direct access devices. The VM/SP or VM/SP HPO DDR program can be invoked by the CMS DDR command in a virtual machine controlled by CMS.

CMS users should refer to the *VM/SP CP for System Programming* or *VM/SP HPO CP for System Programming* for instructions on using the DDR command.

Loop

The real cause of a loop is usually an instruction that sets or branches on the condition code incorrectly. The existence of a loop can usually be recognized by the ceasing of productive processing and a continual returning of the PSW instruction address to the same address. If I/O operations are involved, and the loop is a very large one, it may be extremely difficult to define, and may even comprise nested loops. Probably, the most difficult case of looping to determine is entry to the loop from a wild branch. The problem in loop analysis is finding either the instruction that should open the loop or the instruction that passed control to the set of looping instructions.

CP Disabled Loop

The processor operator should perform the following sequence when gathering information to find the cause of a disabled loop:

1. Operator should trace the CP instruction currently executing in the processor. In an attached processor (AP) or multiprocessor (MP) system, you may have to trace both processors.

2. Force a CP system restart to cause an abend dump to be taken.

3. Save the information collected for the system programmer or system support personnel.

After the processor operator has collected the information, the system programmer or system support personnel should examine it:

1. Use the instructions traced by the operator and the load map to determine the modules that may be involved in the loop.

2. If the cause of the loop is not apparent, examine the CP internal trace table to determine the modules that may be involved in the loop.

3. Other information, such as:

   - PSW
   - General purpose registers
   - Control registers
   - Storage locations X'00' through X'100'

   can be used to determine the condition that caused the loop.

Virtual Machine Disabled Loop

When a disabled loop in a virtual machine exists, the virtual machine operator cannot communicate with the virtual machine's operating system. That means that signalling attention does not cause an interrupt.

Enter the CP console function mode.

1. Use the CP TRACE or CP PER commands to trace the entire loop. Display general purpose and extended control registers using the CP DISPLAY command.

2. Take a dump using the CP DUMP or CP VMDUMP command. The IPCS component of VM/SP may be used to process the file created by the VMDUMP command. For details, see Chapter 7, "Debugging Using IPCS" on page 197.

3. Examine the source code.

Use the information just gathered, along with listings, to try to find the entry into the loop.

If the operating system in the virtual machine itself manages virtual storage, it is usually better to use that operating system's dump program. CP does not retrieve pages that exist only on the virtual machine's paging device.

Virtual Machine Enabled Loop

The virtual machine operator should perform the following sequence when trying to find the cause of an enabled loop:

1. Use the CP TRACE or CP PER commands to trace the entire loop. Display the PSW and the general purpose registers.

2. If your virtual machine has the Extended Control (EC) mode and the EC option, also display the control registers.

3. Use the CP DUMP or CP VMDUMP command to dump your virtual storage. The IPCS component of VM/SP may be used to process the file created by the VMDUMP command. For details, see Chapter 7, "Debugging Using IPCS" on page 197.

4. Consult the source code to search for the faulty instructions, examining previously executed modules if necessary. Begin by scanning for instructions that set the condition code or branch on it.

5. If the manner of loop entry is still undetermined, assume that a wild branch has occurred and begin a search for its origin.

Wait

No processing occurs in the virtual machine when it is in a wait state. When the wait state is an enabled one, an I/O interrupt causes processing to resume. Likewise, when CP is in a wait state, its processing ceases.

CP Disabled Wait

A disabled wait state usually results from a hardware malfunction. During the Initial Program Load (IPL) process, normally correctable hardware errors may cause a wait state because the operating system error recovery procedures are not accessible at this point. These conditions are recorded in the current PSW.

CP may be in an enabled wait state with channel 0 disabled when it is trying to acquire more free storage. Examine EC register 2 to see whether or not the multiplexer channel is disabled. A severe machine check could also cause a CP disabled wait state.

The following three types of severe machine checks can cause CP to terminate or cause a CP disabled wait state:

- Unrecoverable machine check in CP
- Machine check that cannot be diagnosed

- Timing facilities damage.

A machine check error cannot be diagnosed if either the machine check old PSW or the machine check interrupt code is invalid. These severe machine checks cause CP to terminate.

If a severe machine check or channel check caused a CP disabled wait state, one of the following messages appears:

```
DMKCCH603W CHANNEL ERROR
DMKMCH612W MACHINE CHECK; TIMING FACILITIES DAMAGE
DMKMCT612W MACHINE CHECK; TIMING FACILITIES DAMAGE
```

If an unrecoverable machine check occurs in CP, a message:

```
DMKMCH610W MACHINE CHECK; SUPERVISOR DAMAGE cpuid
```

-- or --

```
DMKMCT610W MACHINE CHECK; SUPERVISOR DAMAGE cpuid
```

appears on the processor console. CP is terminated and enters wait state 001 or wait state 013.

If the machine check handler cannot diagnose a certain machine check, the integrity of the system is questionable. A message:

```
DMKMCH611W MACHINE CHECK; SYSTEM INTEGRITY LOST cpuid
```

-- or --

```
DMKMCT611W MACHINE CHECK; SYSTEM INTEGRITY LOST cpuid
```

appears on the processor console. CP is terminated and enters wait state 001 or wait state 013.

Hardware errors are probably the cause of these severe machine checks. The system operator should run the CPEREP program and save the output for the installation hardware maintenance personnel.

If the generated system cannot run on the real machine because of insufficient storage, CP enters the disabled wait state with code X'00D' in the PSW. The insufficient storage condition occurs if:

- The generated system is larger than the real machine size

  -- or --

- A hardware malfunction occurs which reduces the available amount of real storage to less than that required by the generated system.

The message:

```
DMKOPE955W INSUFFICIENT STORAGE FOR VM/SP
```

appears on the processor console.

If CP cannot continue because consecutive hardware errors are occurring on one or more VM/SP or VM/SP HPO paging devices, a message:

```
DMKPAG415E  CONTINUOUS PAGING ERRORS FROM DASD rdev
```

-- or for VM/SP HPO only --

```
DMKMCH635I  MACHINE CHECK; PAGING STORAGE CONTROL FAILURE nn
```

appears on the processor console and CP enters the disabled wait state with code X'00F' in the PSW.

If more than one paging device is available, disable the device on which the hardware errors are occurring and IPL the system again. If CP is encountering hardware errors on its only paging device, move the paging volume to another physical device and IPL again.

Note: This error condition may occur if the CP paging volume was not properly formatted.

The following procedure should be followed by the processor operator to record the needed information:

1. Using the alter/display mode of the processor console, display the real PSW and CSW. Also, display the general purpose registers and the control registers.

2. Force a CP system restart to get a system abend dump.

3. IPL the system.

Examine this information and try to find what caused the wait. If you cannot find the cause, try to reconstruct the situation that existed just before the wait state was entered.

CP Enabled Wait

If you determine that CP is in an enabled wait state, but that no I/O interrupts are occurring, there may be an error in the CP routine or CP may be failing to get an interrupt from a hardware device. Force a CP system restart at the operator's console to cause an abend dump to be taken. Use the abend dump to determine the cause of the enabled (and noninterrupted) wait state. After the dump is taken, IPL the system.

Using the dump, examine the VMBLOK for each user and the real device, channel, and control unit blocks. If each user is waiting because of a request for storage and no more storage is available, there is an error in

CP. There may be looping in a routine that requests storage. Refer to "Reading CP Abend Dumps" on page 76 for specific information on how to analyze a CP dump.

Virtual Machine Disabled Wait

CP does not allow the virtual machine to enter a disabled wait state or certain interrupt loops. Instead, CP notifies the virtual machine operator of the condition with one of the following messages:

```
DMKDSP450W   CP entered; disabled wait PSW psw
DMKDSP452W   CP entered; external interrupt loop
DMKPRG453W   CP ENTERED; PROGRAM INTERRUPT LOOP
```

and enters the console function mode. Use the CP commands to display the following information on the terminal:

- PSW
- CSW
- General purpose registers
- Control registers.

Then use the CP DUMP or VMDUMP command to take a dump. The IPCS component of VM/SP may be used to process the file created by the VMDUMP command. For details, see Chapter 7, "Debugging Using IPCS" on page 197.

If you cannot find the cause of the wait or loop from the information just gathered, try to reproduce the problem, this time tracing the processing via the CP TRACE or CP PER commands.

If CMS is running in the virtual machine, the CMS debugging facilities may also be used to display information, take a dump, or trace the processing. The CMS SVCTRACE, CP TRACE, and CP PER commands record different information.

Virtual Machine Enabled Wait

If the virtual machine is in an enabled wait state, try to find out why no I/O or external interrupts have occurred to allow processing to resume.

CP treats one case of an enabled wait in a virtual machine the same as a disabled wait. If the virtual machine does not have the "real timer" option, CP issues the message:

```
DMKDSP450W   CP entered; disabled wait PSW psw
```

Since the virtual timer is not decreased while the virtual machine is in a wait state, it cannot cause the external interrupt. A "real timer" runs in both the problem state and wait state and can cause an external interrupt which allows processing to resume. The clock comparator can also cause an external interrupt.

Summary of VM Debugging Tools

Figure 7 summarizes the VM commands that are useful for interactively debugging a problem that currently exists. The commands are classified by the function they perform. See Chapter 7, "Debugging Using IPCS" on page 197 and Appendix A, "Using DUMPSCAN Subcommands" on page 259 for information on interactive dump handling.

Function	Comments	Commands

Stop execution at a specified location	Set the address stop before the program reaches the specified address. For CP, ADSTOP allows 1 active address stop; PER allows multiple address stops.	`ADSTOP hexloc`
		`PER Instruct Range single-addr`

Resume execution	Resume execution where program was interrupted	`BEGIN`
	Continue execution at a specific location	`BEGIN hexloc`

Dump data	Dump the contents of specific storage locations	`DUMP {hexloc1 | Lhexloc1} [ {- | :} [hexloc2 | END] | {.} [bytecount | END] ]`
		`[*dumpid]`

Dump data	VMDUMP provides the same information that DUMP provides but in a different format; The format is also compatible with VM/SP IPCS.	`VMDUMP {hexloc1 | 0} [ {- | :} [hexloc2 | END] | {.} [bytecount | END] ]`
		`[TO * | TO userid | SYSTEM]`
		`[FORMAT vmtype]`
		`[DSS]`
		`[*dumpid]`

Figure 7 (Part 1 of 8). Summary of VM Debugging Tools

Function	Comments	Commands
Display virtual data	Display contents of storage locations in hexadecimal.	`DISPLAY hexloc1 [ {- :} [hexloc2 / END] ; {.} [bytecount / END] ]`
	Display contents of storage locations (in hexadecimal and EBCDIC)	`DISPLAY Thexloc1 [ {- :} [hexloc2 / END] ; {.} [bytecount / END] ]`
	Display storage key of specific storage locations in hexadecimal	`DISPLAY Khexloc1 [ {- :} [hexloc2 / END] ; {.} [bytecount / END] ]`
	Display general purpose registers	`DISPLAY Greg1 [ {- :} [reg2 / END] ; {.} [regcount / END] ]`
	Display floating point registers	`DISPLAY Yreg1 [ {- :} [reg2 / END] ; {.} [regcount / END] ]`
	Display control registers	`DISPLAY Xreg1 [ {- :} [reg2 / END] ; {.} [regcount / END] ]`
	Display contents of current virtual PSW in hexadecimal format	`DISPLAY PSW`

Figure 7 (Part 2 of 8). Summary of VM Debugging Tools

Function	Comments	Commands
	Display contents of CAW	`DISPLAY CAW`
	Display contents of CSW	`DISPLAY CSW`
	Display contents of Paging Storage (**VM/SP HPO only**).	`DDR`
	Display contents of vector register (**VM/SP HPO only**).	$\texttt{Display}\ \left\{\begin{matrix}\texttt{VR}\\\texttt{VF}\end{matrix}\right\}\ \left[\begin{matrix}\underline{\texttt{0}}\\\texttt{vreg1}\end{matrix}\right]\ \left[\begin{matrix}\left\{\begin{matrix}\texttt{-}\\\texttt{:}\end{matrix}\right\} & \left[\begin{matrix}\texttt{vreg2}\\\texttt{END}\end{matrix}\right]\\ \{\texttt{.}\} & \left[\begin{matrix}\texttt{vregcount}\\\texttt{END}\end{matrix}\right]\end{matrix}\right]$ $\left[\texttt{,element1}\ \left[\begin{matrix}\left\{\begin{matrix}\texttt{-}\\\texttt{:}\end{matrix}\right\} & \left[\begin{matrix}\texttt{element2}\\\texttt{END}\end{matrix}\right]\\ \{\texttt{.}\} & \left[\begin{matrix}\texttt{elementcount}\\\texttt{END}\end{matrix}\right]\end{matrix}\right]\right]$
	Display contents of vector activity counter (**VM/SP HPO only**).	`DISPLAY VAC`
	Display contents of vector status register (**VM/SP HPO only**).	`DISPLAY VSR`
	Display contents of vector mask register (**VM/SP HPO only**).	`DISPLAY VMR`
Display Real CP Data	Display contents of processor storage locations (in hexadecimal)	$\texttt{DCP}\ \ \texttt{hexloc1}\ \left[\begin{matrix}\left\{\begin{matrix}\texttt{-}\\\texttt{:}\end{matrix}\right\} & \left[\begin{matrix}\texttt{hexloc2}\\\underline{\texttt{END}}\end{matrix}\right]\\ \{\texttt{.}\} & \left[\begin{matrix}\texttt{bytecount}\\\underline{\texttt{END}}\end{matrix}\right]\end{matrix}\right]$ $\texttt{DCP}\ \ \texttt{Lhexloc1}\ \left[\begin{matrix}\left\{\begin{matrix}\texttt{-}\\\texttt{:}\end{matrix}\right\} & \left[\begin{matrix}\texttt{hexloc2}\\\underline{\texttt{END}}\end{matrix}\right]\\ \{\texttt{.}\} & \left[\begin{matrix}\texttt{bytecount}\\\underline{\texttt{END}}\end{matrix}\right]\end{matrix}\right]$

Figure 7 (Part 3 of 8). Summary of VM Debugging Tools

Function	Comments	Commands
	Display contents of processor storage locations (in hexadecimal and EBCDIC)	DCP Thexloc1 $\left[\begin{array}{ll}\left\{\begin{array}{c}-\\ :\end{array}\right\} & \left[\begin{array}{l}\text{hexloc2}\\ \underline{\text{END}}\end{array}\right]\\ \{.\} & \left[\begin{array}{l}\text{bytecount}\\ \underline{\text{END}}\end{array}\right]\end{array}\right]$ DCP T
	Display contents of storage locations in IPL processor (in hexadecimal)	DCP Mhexloc1 $\left[\begin{array}{ll}\left\{\begin{array}{c}-\\ :\end{array}\right\} & \left[\begin{array}{l}\text{hexloc2}\\ \underline{\text{END}}\end{array}\right]\\ \{.\} & \left[\begin{array}{l}\text{bytecount}\\ \underline{\text{END}}\end{array}\right]\end{array}\right]$ DCP MLhexloc1 $\left[\begin{array}{ll}\left\{\begin{array}{c}-\\ :\end{array}\right\} & \left[\begin{array}{l}\text{hexloc2}\\ \underline{\text{END}}\end{array}\right]\\ \{.\} & \left[\begin{array}{l}\text{bytecount}\\ \underline{\text{END}}\end{array}\right]\end{array}\right]$ DCP M DCP ML
	Display contents of storage locations in nonIPL processor (in hexadecimal)	DCP Nhexloc1 $\left[\begin{array}{ll}\left\{\begin{array}{c}-\\ :\end{array}\right\} & \left[\begin{array}{l}\text{hexloc2}\\ \underline{\text{END}}\end{array}\right]\\ \{.\} & \left[\begin{array}{l}\text{bytecount}\\ \underline{\text{END}}\end{array}\right]\end{array}\right]$ DCP NLhexloc1 $\left[\begin{array}{ll}\left\{\begin{array}{c}-\\ :\end{array}\right\} & \left[\begin{array}{l}\text{hexloc2}\\ \underline{\text{END}}\end{array}\right]\\ \{.\} & \left[\begin{array}{l}\text{bytecount}\\ \underline{\text{END}}\end{array}\right]\end{array}\right]$ DCP N DCP NL

Figure 7 (Part 4 of 8). Summary of VM Debugging Tools

Function	Comments	Commands
	Display contents of storage locations in IPL processor (in hexadecimal and EBCDIC)	DCP MThexloc1 [{- :} [hexloc2 END] {.} [bytecount END]] DCP MT
	Display contents of storage locations in nonIPL processor (in hexadecimal and EBCDIC)	DCP NThexloc1 [{- :} [hexloc2 END] {.} [bytecount END]] DCP NT
Store virtual data	Store specified information into consecutive storage locations without alignment	STORE Shexloc hexdata...

	Store specified words of information into consecutive fullword storage locations	STORE {hexloc
		Ihexloc}
		{hexword1 [hexword2...]}

	Store specified words of information into consecutive general purpose registers	STORE Greg hexword1 [hexword2...]
	Store specified words of information into consecutive floating-point registers	STORE Yreg hexword1 [hexword2...]
	Store specified words of data into consecutive control registers	STORE Xreg hexword1 [hexword2...]
	Store information into PSW	STORE [PSW hexword1] hexword2

Figure 7 (Part 5 of 8). Summary of VM Debugging Tools

Function	Comments	Commands
	Store information in CSW	`STORE [40 hexword1]`
	Store information in CAW	`STORE [48 hexword1]`
	Store information in vector register elements **(VM/SP HPO only).**	`STORE { VR [vreg] [,element] hexword1 [hexword2...] | VP [vreg] [,element] hexword1 [hexword2...] }`
	Store information in vector status register **(VM/SP HPO only).**	`STORE VSR hexword1 [hexword2]`
	Store information in vector mask register **(VM/SP HPO only).**	`STORE VMR hexword1 [hexword2]`

Store Real CP Data	Store specified words of information into consecutive processor storage locations	`STCP hexloc hexword1 [hexword2 ...]`
		 `STCP Lhexloc hexword1 [hexword2 ...]`

	Store specified words of information into consecutive IPL processor storage locations	`STCP Mhexloc hexword1 [hexword2 ...]`
		 `STCP MLhexloc hexword1 [hexword2 ...]`

	Store specified words of information into consecutive nonIPL processor storage locations	`STCP Nhexloc hexword1 [hexword2 ...]`
		 `STCP NLhexloc hexword1 [hexword2 ...]`

	Store specified information into consecutive storage locations without alignment	`STCP Shexloc hexdata`
	Store specified information into consecutive storage locations without alignment (in IPL processor)	`STCP MShexloc hexdata`

Figure 7 (Part 6 of 8). Summary of VM Debugging Tools

Function	Comments	Commands
	Store specified information into consecutive storage locations without alignment (in nonIPL processor)	`STCP NShexloc hexdata`
Trace execution	Trace all instructions, interrupts, and branches	`TRACE ALL`

	Trace SVC interrupts	`TRACE SVC`
		`PER Instruct DATA 0Axx`
		`SVCTRACE ON (note this is a CMS command)`

	Trace I/O interrupts	`TRACE I/O`
	Trace program interrupts	`TRACE PROGRAM`
	Trace external interrupts	`TRACE EXTERNAL`

	Trace privileged instructions	`TRACE PRIV`
		`PER Instruct DATA xx`
		`(PER can trace specific privileged instructions.)`

	Trace all user I/O operations	`TRACE SIO`
		`PER Instruct DATA xx`

	Trace virtual and real CCWs	`TRACE SIO`
		`TRACE CCW`

	Trace all user interrupts and successful branches	`TRACE BRANCH`
		`PER BRANCH`

	Trace successful branches	`PER BRANCH [[INTO] into-addr-range]`

	Trace instructions	`TRACE INSTruct`
		`PER Instruct`
		`[Range instruction-addr-range]`

	Trace instructions that alter storage	`PER STORE`
		`[[INTO] storage-addr-range ]`
		`[[INTO] addr [DATA] hex-data]`

	Trace instructions that alter general purpose registers	`PER G[reg1] [ {- or :} [reg2] ]`
		`[ {.} [regcount] ]`

Figure 7 (Part 7 of 8). Summary of VM Debugging Tools

Function	Comments	Commands
	Trace instructions that alter specific bits at specific storage locations	PER MASK [INTO] addr [DATA] mask-field
	End tracing activity	TRACE END PER END { ALL CUrrent element-number event-type traceset name } SVCTRACE OFF (note this is a CMS command)
Trace real machine events	Trace events in real machine	MONITOR START CPTRACE
	Stop tracing events in the real machine	MONITOR STOP CPTRACE
	Enable a virtual machine to enter data in CPTRAP file	CPTRAP { ALLOWid userid GRoupid group-name }
	Specify selectivity in collecting CPTRAP data	CPTRAP ALL [ON OFF] CPTRAP typenum [Vmblok address DEVaddr cuu COde code-value MACHtype { nn machname } OFF]

Figure 7 (Part 8 of 8). Summary of VM Debugging Tools

Chapter 2. Debugging the Virtual Machine

Debugging the Virtual Machine

The Control Program (CP) provides interactive commands that control the system and enable the user to control his virtual machine and associated control program facilities. The virtual machine operator using these commands can gather much the same information about his virtual machine as the operator of a real machine gathers using the processor console.

Several of these commands (for example, STORE or DISPLAY) examine or alter virtual storage locations. When CP is in complete control of virtual storage (as in the case of DOS, MFT, MVT, PCP, CMS, and GCS) these commands execute as expected. However, when the operating system in the virtual machine itself manipulates virtual storage (as in the case of OS/VS1, OS/VS2, or DOS/VS) these CP commands should not be used.

This chapter presents an overview of the VM/SP commands used for debugging. It supplements the preceding section which discussed debugging procedures and techniques. Instructions for using the commands discussed in this section are in the following:

- *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference*
- Chapter 7, "Debugging Using IPCS" on page 197.

The following categories of commands are discussed:

- Commands that display or dump virtual machine data
- Commands that set and query system features, conditions, and events
- Commands that collect and analyze system information
- Commands that trace events in virtual machines
- Commands that alter the contents of storage.

Commands that Display or Dump Virtual Machine Data

Commands that display or dump virtual machine data are: DUMP, VMDUMP, DISPLAY, DCP, and DMCP. See *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for more information on these commands.

The DUMP and DISPLAY commands of CP are privilege class G commands and are used to display control information describing the status of virtual machines.

The DUMP command spools the following information to your virtual printer:

- Virtual Program Status Word (PSW)
- General purpose registers
- Floating-point registers
- Control registers (if you have set ECMODE ON)
- Storage keys
- Virtual storage locations (first-level storage only).

For more information on control registers see Appendix D, "Control Registers" on page 351.

When a program you execute under CMS abnormally terminates, you do not automatically receive a program dump. If, after attempting to use CMS and CP to debug interactively, you still have not discovered the problem, you may want to obtain a dump. You might also want to obtain a dump if you find that you are displaying large amounts of information, which is not practical on a terminal.

Issue the command:

```
#CP VMDUMP 0-END FORMAT CMS DSS
```

Then use IPCS to format and view the dump (see Chapter 7, "Debugging Using IPCS" on page 197).

You can selectively dump portions of your virtual storage, your entire virtual storage area, or portions of real storage. For example, in the debug environment, to dump the virtual storage space that contains your program, you would enter:

```
CP DUMP T20000-20810
```

The second value depends upon the size of your program.

The CP DUMP command allows you to request EBCDIC translation with the hexadecimal dump.

The class G VMDUMP command dumps virtual storage to a specified reader spool file. VMDUMP provides the same dump information that the DUMP command provides but in a different format. For example, if a byte of storage contains X'00', DUMP records it in printable format, X'F0F0'; VMDUMP records it as it appears in storage, X'00'. The IPCS component of VM/SP may be used to process the file created by the VMDUMP command. For details, see Chapter 7, "Debugging Using IPCS" on page 197. For a description of the format and contents of the VMDUMP records, see "VMDUMP Records: Format and Content" on page 94.

The DISPLAY command displays at your terminal the following kinds of control information:

- Virtual storage locations (first-level storage only)
- Storage keys
- General purpose registers
- Floating-point registers
- Vector registers (**VM/SP HPO only**)
- Control registers
- PSW
- Channel Address Word (CAW)
- Channel Status Word (CSW).

When you use the display command, you can request an EBCDIC translation of the display by prefacing the location you want displayed with a "T".

```
CP DISPLAY T20000.10
```

This command requests a display of X'10' (16) bytes beginning at location X'20000' The display is formatted four words to a line, with EBCDIC translation at the right, much as you would see it in a dump.

You can also use the DISPLAY command to examine the general registers. For example, the commands:

```
CP DISPLAY G
CP DISPLAY G1
CP DISPLAY G2-5
```

result in displays of all the GPRs, of GPR1, and of a range of GPRs 2 through 5.

The DISPLAY command also displays the PSW, CAW, and CSW:

```
CP DISPLAY PSW
CP DISPLAY CAW
CP DISPLAY CSW
```

The DCP and DMCP commands of CP are privilege class C and E commands and are used to display real storage locations. The DMCP command spools the contents of real storage to your virtual printer. The DCP command displays at your terminal the contents of real storage locations.

Terminal Output

With the DISPLAY command, you can display virtual storage at your terminal in either of the following formats:

- Four-byte groups, aligned on fullword boundaries, hexadecimal format, printed four fullwords per line
- 16-byte groups, aligned on 16 byte boundaries, hexadecimal format, printed four fullwords plus EBCDIC translation per line.

For the first format, enter the DISPLAY command as:

```
DISPLAY 1026-102C
```

you receive the response:

```
001024 xxxxxxxx xxxxxxxx xxxxxxxx
```

For the second format, enter the command as:

DISPLAY T1026-102C

and the response is:

```
                                                 (EBCDIC trans.)
001020 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx *................*
```

You can also specify the area of storage to be displayed by entering a hexadecimal byte count such as:

DISPLAY 1024.12

The response displays 20 bytes as follows:

```
001024 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
001034 xxxxxxxx
```

Byte Alignment on Terminal Output

The previous responses illustrate the byte alignment that takes place in each of the two display formats.

If the first location to be displayed is not on the appropriate 4 or 16 byte boundary, it is rounded down to the next lower boundary that applies.

If the last location to be displayed does not fall at the end of the appropriate 4 or 16 byte group, it is rounded up to the end of that group.

If you enter:

DISPLAY K1024-3200

the storage keys that are assigned to each 2K segment of the specified storage area are displayed. Contiguous 2K segment with identical storage keys are combined; for example, the response could have been:

```
001000 TO 0027FF  KEY=F0
002800 TO 003800  KEY=E0
```

To display all storage keys, enter:

DISPLAY K

If your virtual machine is in extended control mode (ECMODE ON), you can interrogate any of the control registers:

DISPLAY X1 4 A

and receive the response:

```
ECR  1 = xxxxxxxx
ECR  4 = xxxxxxxx
ECR 10 = xxxxxxxx
```

However, the same command entered while your virtual machine does not have ECMODE ON results in the response:

```
ECR 0 = xxxxxxxx
ECR 0 = xxxxxxxx
ECR 0 = xxxxxxxx
```

As each operand in the command line is processed, VM/SP determines that ECMODE is OFF and replaces any reference to a control register with ECR 0, the only control register available in Basic Control (BC) mode.

Printer Output

With the DUMP command you can dump the contents of all registers, the PSW and the storage keys, along with any specified area of virtual storage, to the virtual machine's spooled printer. The printer format for storage locations is 8 fullwords per line plus the EBCDIC translation on the right.

To print only the registers, the PSW, and the storage keys, you need only enter:

DUMP 0

To also print an area of virtual storage, you can specify the beginning and ending hexadecimal locations:

DUMP 1064-10FF

You can also specify the beginning location and the number of bytes to be dumped; both values are entered in hexadecimal:

DUMP 1064.9B

If you are printing a series of dumps, you can identify each one by including its identification on the DUMP command line, following an asterisk:

DUMP 1000-2000 * DUMP NO. 1

To print the dump data on the real printer you must first close the virtual printer. Issue the command:

CLOSE PRINTER

and the dump data spool file is placed on an appropriate system printer queue.

You can use the VMDUMP command that dumps storage for guest virtual machines. VMDUMP provides IPCS with header information to identify the owner of the dump; it also maintains dump information, writes the dump to a class v reader spool file, and formats the dump.

When you enter at the terminal:

```
VMDUMP 150-200
```

-- or --

```
VMDUMP 400:500
```

CP dumps the contents of virtual machine storage at the hexadecimal addresses between X'150' and X'200' or between X'400' and X'500', respectively.

If you enter:

```
VMDUMP 150.50
```

CP dumps the contents of virtual storage starting at X'150' for a total of X'50' bytes.

Commands that Set and Query System Features, Conditions, and Events

The SYSTEM and SET commands set system-controlled functions and events; the QUERY command allows you to determine the status of those settings.

The SYSTEM command is a privilege class G command that simulates the RESET and RESTART functions on a real computer console. It can also be used to clear storage.

The functions of the SET command are described in detail in the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference.* For debugging, the SET command provides the MSG, WNG, and EMSG operands. These provide messages that may be useful while you are debugging.

The SET MSG function determines whether you receive messages sent by other users via the MSG command.

The SET SMSG command turns on or off a virtual machine's special message flag. If the virtual machine has issued DIAGNOSE code X'68' (AUTHORIZE), this flag determines whether the virtual machine accepts or rejects messages sent via the SMSG command -- when the flag is on, messages are accepted.

The SET WNG function determines whether you receive warning messages from the system operator.

The SET EMSG function controls error message handling. The EMSG operand gives you the ability to specify that you want message code, message text, or both to be displayed at your terminal. You can also specify that no messages be displayed (except in the case where you have spooled your console output).

The SET IMSG command controls whether certain informational responses issued by some CP commands are displayed at the terminal or not. Also, the SET IMSG command determines whether you receive messages from CP when other users spool reader, printer, or punch files to your virtual machine.

When you are debugging, it is useful to have all messages displayed at your terminal.

The QUERY command displays the status of features and conditions set by the SET command for your virtual machine. When you logon, the MSG and WNG operands of the SET command are set ON; the EMSG operand is set to TEXT; and the SMSG operand is set OFF. To verify these settings, use the QUERY SET command.

Commands that Trace Events in Virtual Machines

This section discusses the TRACE and PER commands. The TRACE command traces virtual machine events. The PER command selectively traces the execution of the instructions that cause specific events. The SVCTRACE command provides additional information about SVCs that the TRACE command could not provide. See "Using the SVCTRACE command" on page 141 for more information.

Stopping Virtual Machine Execution at a Specific Address

The ADSTOP command stops the execution of a virtual machine at a specific address; BEGIN causes the virtual machine to resume execution. See *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for more information concerning the ADSTOP and BEGIN commands.

To stop execution of your virtual machine at a given address in virtual storage, use the ADSTOP command and specify the hexadecimal address of a virtual instruction. The command:

#CP ADSTOP 3000

stops the virtual machine when the instruction at hexadecimal location 3000 is the next instruction to be executed. When the machine stops running, you receive the message:

```
ADSTOP AT 3000
```

and your terminal is placed in CP console function mode. At this point, you can enter other CP debugging commands to display and alter storage or to trace certain instructions. When you want to resume running your virtual machine, enter:

```
BEGIN
```

Unlike the hardware address stop, ADSTOP is turned off when:

- Requested address is reached
- Next ADSTOP command is issued
- IPL or a system reset is performed
- ADSTOP OFF command is issued.

While ADSTOP is on, the SVC portion of virtual machine assist is not executed. When ADSTOP is turned off, SVCs are again handled by virtual machine assist.

The address stop should be set after the program is loaded but before it executes. When the specified location is reached during program execution, execution halts and the CP command environment is entered. You may then enter other CP commands to examine and alter the status of the program.

Set an address stop at a location where you suspect the error in the program. You can then display the registers, control words, and data areas to check the program at that point in its execution. This procedure helps you locate program errors. You may be able to alter the contents of storage in such a way that the program will execute correctly. You can then correct the error you have detected and, if necessary, compile and execute the program again.

To successfully set an address stop, the instruction must be in first-level storage of the virtual machine at the time the ADSTOP command is issued.

Using the CP TRACE Command

You can trace the following kinds of activity in a program using the CP TRACE command:

- Instructions (privileged and PSW)
- Branches
- Interrupts
  - Program
  - External
  - I/O
  - SVC
- I/O and channel activity.

See *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for more information concerning the TRACE command.

When the TRACE command executes, it traces all your virtual machine's activity; when your program issues a supervisor call, or calls any CMS routine, the TRACE continues.

Tracing resumes when these two conditions are met:

- CP gains control, such as for a real I/O interruption, and
- Virtual machine encounters one of the specified activities to be traced, except for successful branching.

Whenever you are recording trace output at your terminal, the virtual machine stops execution and enters the CP console read environment after each output line. This is the default mode of operation when, for example, you enter:

TRACE ALL

-- or --

TRACE SVC PROGRAM BRANCH

If you only want to record the trace and not stop after each output line, add the RUN operand as the last entry on the command line.

If, having specified multiple activities to be traced, you decide to stop tracing one or more of them, enter:

TRACE PROGRAM BRANCH OFF

and tracing is now confined to SVCs only.

To trace all activity with the output directed to the virtual printer, enter:

TRACE ALL PRINTER

When you stop tracing, you must also issue the CLOSE command to print the spooled trace output on the real printer:

TRACE END
CLOSE PRINTER

If your virtual machine configuration contains only one printer, trace output is intermixed with application output. You should define another virtual printer with an address lower than the previously defined printer. Application output is still directed to the original printer; however, trace output is always directed to the printer with the lowest address.

While trace is running, portions of virtual machine assist are disabled. When the trace is complete, they are enabled.

You can make most efficient use of the TRACE command by starting the trace at a specific instruction location. You should set an address stop for the location. For example, if you are going to execute a program and you want to trace all of the branches made, you would enter the following sequence of commands to begin executing the program and start the trace:

LOAD PROGRESS
CP ADSTOP 20004
START
ADSTOP AT 20004
CP TRACE BRANCH
CP BEGIN

Now, whenever your program executes a branch instruction, you receive information at the terminal that might look like this:

```
02001E BALR  05E6    ==> 020092
```

This line indicates that the instruction at address X'2001E' resulted in a branch to the address X'020092'. When this information is displayed, your virtual machine is placed in the CP environment, and you must use the BEGIN command to continue execution:

CP BEGIN

When you locate the branch that caused the problem in your program, you should terminate tracing activity by entering:

CP TRACE END

and then you can use CP commands to continue debugging or you can use the EXTERNAL command to cause an external interruption that places your virtual machine in the debug environment:

CP EXTERNAL

You receive the message:

```
DEBUG ENTERED.
EXTERNAL INTERRUPT
```

And you can use the DEBUG subcommands to investigate the status of your program.

Controlling a CP Trace

There are several things you can do to control the amount of information you receive when you are using the TRACE command, and how it is received. For example, if you do not want program execution to halt every time a trace output message is issued, you can use the RUN option:

CP TRACE SVC RUN

Then, you can halt execution by pressing the Attention key when the interruption you are waiting for occurs. You should use this option if you do not want to halt execution at all, but merely want to watch what is happening in your program.

Similarly, if you do not require your trace output immediately, you can specify that it be directed to the printer, so that your terminal does not receive any information at all:

```
CP TRACE INST PRINTER
```

When you direct trace output to a printer, the trace output is mixed in with any printed program output. If you want trace output separated from other printed output, use the CP DEFINE command to define a second printer at a virtual address lower than that of your printer at 00E. For example:

```
CP DEFINE PRINTER 006
```

Then, trace output will be in a separate spool file. CMS printed output always goes to the printer at address 00E.

When you finish tracing, use the CP CLOSE command to close the virtual printer file:

```
CP CLOSE E
```

-- or --

```
CP CLOSE 006
```

If you want trace output at the printer and at the terminal, you can use the BOTH option:

```
CP TRACE ALL BOTH
```

Suspending Tracing

If you are debugging a program that does a lot of I/O, or that issues many SVCs, and you are tracing instructions or branches, you might not wish to have tracing in effect when the supervisor or I/O routine has control. When you notice that addresses being traced are not in your program, you can enter:

```
CP TRACE END
```

and then set an address stop at the location in your program that receives control when the supervisor or I/O routine has completed:

```
CP ADSTOP 20688
BEGIN
```

Then, when this address is encountered, you can re-enter the CP TRACE command.

What To Do When Your Program Loops

If, when your program is executing, it seems to be in a loop, you should first verify that it is looping, and then interrupt its execution and either:

- Halt it entirely and return to the CMS environment, or
- Resume its execution at an address outside of the loop.

The first indication of a program loop may be either what seems to be an unreasonably long processing time, or, if you have a blip character defined, an inordinately large number of blips.

You can verify a loop by checking the PSW frequently. If the last word repeatedly contains the same address, it is a fairly good indication that your program is in a loop. You can check the PSW by using the Attention key (except with 3270 type terminals) to enter the CP environment. You are notified by the message:

```
CP
```

that your virtual machine is in the CP environment. You can then use the CP command DISPLAY to examine the PSW:

```
CP DISPLAY PSW
```

and then enter the command BEGIN to resume program execution:

```
CP BEGIN
```

If you are checking for a loop, you might enter both commands on the same line using the logical line end:

```
CP DISPLAY PSW#BEGIN
```

When you have determined that your program is in a loop, you can halt execution using the CMS Immediate command HX. To enter this command, you must press the Attention key once (except with 3270 type terminals) to interrupt program execution, then enter:

```
HX
```

If you want your program to continue executing at an address past the loop, you can use the CP command BEGIN to specify the address at which you want to continue execution:

```
CP BEGIN 20CD0
```

Or, you could use the CP command STORE to change the instruction address in the PSW before entering the BEGIN command:

CP STORE PSW 0 20CD0#BEGIN

Debugging with CP After a Program Check

When a program that is executing under CMS abends because of a program check, the DEBUG routine is in control and saves your program's registers, so that if you want to begin debugging, you must use the DEBUG command to enter the debug environment.

You can prevent DEBUG from gaining control when a program interruption occurs by setting the wait bit in the program new PSW:

CP TRACE PROG NORUN

You should do this before you begin executing your program. Then, if a program check occurs during execution, when CP tries to load the program new PSW, the wait bit forces CP into a disabled wait state and you receive the message:

```
start
EXECUTION BEGINS...
***024602 PROG 0001 ==> 1E3D18
```

All of your program's registers and storage areas remain exactly as they were when program interruption occurred. The PSW that was in effect when your program was interrupted is in the program old PSW, at location X'28'. Use the DISPLAY command to examine its contents:

CP DISPLAY 28.8

The program new PSW, or the PSW you see if you enter the command DISPLAY PSW, contains the address of the DEBUG routine.

If, after using CP to examine your registers and storage areas, you can recover from the problem, you must use the STORE command to restore the PSW, specifying the address of the instruction just before the one indicated at location X'28'. For example, if the instruction address in your program is X'566' enter:

CP STORE PSW 0 20566
CP BEGIN

In this example, setting the first word of the PSW to 0 turns the wait bit off and clears all other information in the first word, so that execution can resume.

Using the CP PER Command

The CP PER command can be used to trace all:

- Instructions
- Successful branches
- Register alterations
- Instructions executed in your virtual machine that alter storage.

See *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for more information concerning the PER command.

The CP PER command has many options that allow you selectivity in choosing which events are to be monitored. Trace output for the CP PER command is always produced *after* the instruction executes. The RANGE keyword of the CP PER command can be used to set multiple address stops. However, unlike the CP ADSTOP command, the program execution halts *after* the execution of the instruction at the given address. Note also that address stops set using the PER command remain in effect until you turn off the trace element set up by the PER command. There is no need for the program to already be in storage before setting address stops with the CP PER command.

Setting up multiple address stops with PER is accomplished by using RANGE as an option to the INSTRUCT keyword. The instruction-addr-range, in this case, is a single value corresponding to the address of the instruction where program execution is to be halted.

For example,

PER INSTRUCT RANGE 20000

causes program execution to halt after the instruction at location X'20000' executes.

PER INSTRUCT RANGE 20000 RANGE 20400

causes a program to halt after an instruction at either location X'20000' or X'20400' executes.

Note: Although output is produced only after the instruction at X'20000' or X'20400' executes, the hardware causes a PER interrupt for every instruction executed in the range X'20000' to X'20400'. This may degrade the performance of the virtual machine.

The CP QUERY command with the PER option can be used to determine what events are currently being traced. For example:

QUERY PER

may result in:

```
1 INSTRUCT RANGE 020000-0204FF TERMINAL NORUN
```

If in addition to instructions, you wish to trace instructions that alter registers, enter:

PER G RANGE 20000.500

To see which events you are monitoring (your current traceset), enter:

QUERY PER

You will see the following:

```
1 INSTRUCT RANGE 020000-0204FF TERMINAL NORUN
2 G RANGE 020000-0204FF TERMINAL NORUN
```

To change *just* the instruction trace element to PRINTER, you can enter:

PER INSTRUCT RANGE 020000-0204FF PRINTER RUN

To see which events you are monitoring (your current traceset), enter:

QUERY PER

You will see the following:

```
1 INSTRUCT RANGE 020000-0204FF PRINTER RUN
2 G RANGE 020000-0204FF TERMINAL RUN
```

If you continue program execution by entering BEGIN you will receive information at your terminal that might look like this:

```
020004 BALR 05C0    000000 CC=0 G12=40020006
```

This line indicates a BALR instruction at address X'020004' changed register 12 to X'40020006'.

As with CP TRACE, you can specify the printer and/or run for any event. However, CP PER has additional options that can be used with all events:

- The RANGE or FROM option can be used to set up multiple instruction address ranges. This can increase the selectivity with which instruction execution is monitored.
- The PASS option allows you to suppress a specific number of events between displays.
- The CMD option can be used if you want to execute CP command(s) whenever a given event occurs.
- The STEP option can be used to permit a specified number of events to be displayed before the CP command environment is entered.

Selectivity

PER options can be used to increase selectivity. Using PER, it is possible to limit tracing to a specific instruction or instructions. For example, to monitor only LR instructions (op code X'18'), enter:

PER INSTRUCT DATA 18

When the NORUN option is in effect, program execution halts after each monitored event. When using the RUN option, program execution continues after each event. PER also has an execution rate between NORUN and RUN. This option is called STEP. STEP specifies the number of events that should be displayed before program execution halts and the CP command environment is entered. For example, to halt program execution after 5 instructions in the range X'20000' to X'204FF' have been executed, enter:

PER INSTRUCT RANGE 20000.500 STEP 5

When your program has been loaded and started, you will receive information at your terminal that might look like this:

```
==>020000 STM   90ECD00C 00DFAC CC=0
   020004 BALR  05C0     000000 CC=0
   020006 ST    50D0C342 020348 CC=0
   02000A LA    41D0C33E 020344 CC=0
   020012 OC    D607C3A2C3AA 0203A8 0204B0 CC=1
```

and then program execution would halt, and the CP command environment would be entered.

Although the STEP option allows you to step through your program more quickly without giving up all control, every monitored instruction is displayed. If many instructions are executed before the problem occurs, the need arises to frequently clear your screen. The frequency with which events are displayed can be changed by using the PASS option. Ordinarily, every successful event is displayed. However, using the PASS option makes it possible to specify how many monitored events should be skipped before displaying one. For example, to skip the display of 100 instructions and display the 101st, enter:

PER INSTRUCT PASS 100

Terminating PER

To end PER tracing with the current traceset, enter:

PER END CURRENT

To end just the branch trace elements in the current traceset, enter:

PER END BRANCH

It is not always desirable to end all the trace elements of a particular event type. When the current traceset is displayed using the QUERY PER command, each trace element is preceded by a number. This number can be used to selectively end PER tracing.

To see which events you are monitoring (your current traceset), enter:

QUERY PER

You may see the following:

```
1 INSTRUCT PRINTER RUN
2 G5 DATA 00000023 TERMINAL NORUN
3 G7 TERMINAL NORUN
4 STORE INTO 020628-020630 TERMINAL NORUN
```

If you no longer want to monitor instructions that alter register 7 but want to continue monitoring register 5, enter:

PER END 3

To terminate all PER tracing, enter:

PER END ALL

This ends the current and all saved tracesets.

Suspending PER

The PER SAVE subcommand can be used to save the current traceset. The saved traceset is saved only while the user is logged on.

To save the current traceset under the name TRACE1, enter:

PER SAVE TRACE1

The current traceset is still active. To suspend the PER tracing, enter:

PER END CURRENT

PER tracing is no longer active. The traceset is saved under the name TRACE1. You can now execute normally, or create another current traceset using PER commands. This new traceset (or tracesets) can also be saved.

To resume PER tracing with the original traceset, enter:

PER GET TRACE1

A copy of traceset TRACE1 is still saved under the name of TRACE1. Changes made to TRACE1 while it is the current traceset have no effect on this saved copy.

To end the saved traceset TRACE1, enter:

PER END TRACE1

If you save a traceset under the same name as an existing traceset, the two are appended (only if you specify the append option).

Additional Program Debugging Using PER

TRACE can be used to trace program interrupts. However, the trace information is displayed after the interrupt has occurred and cannot always be used to determine the cause of the problem. PER, used in conjunction with TRACE, can greatly reduce the difficulty of finding the cause of the problem. If the problem is an operation exception, it may have been caused by a bad branch instruction.

The Branch Traceback Table

The first step is to trace program interrupts using TRACE:

TRACE PROG

Run the failing program until the program interrupt occurs. When the program interrupt occurs, the address of the instruction causing the interrupt is the address that precedes the address of the instruction that is being displayed. For example:

```
start
EXECUTION BEGINS...
***024602 PROG 0001 ==> 1E3D18
```

Next end TRACE and allow the program to finish. Reload the failing program and trace successful branches to the address of the bad instruction. For example:

PER BRANCH 24600

Note: The branch might be to an address before X'24600'. The branch might have encountered a valid op code. Therefore, it is sometimes necessary to use a larger branch into address. For example:

PER BRANCH 245F0-24600

When the branch to the bad instruction occurs, the branch instruction as well as the previous 5 successful branches are displayed. For example:

```
start
EXECUTION BEGINS...
==>020012 BR    07F1      024600 CC=0
TRACEBACK TABLE:
:1D1320 BR    07F3      1D125A
:1D1268 BR    07FE      1D1322
:1D1356 BNZ   4770E07C 1D139E
:1D13A2 BZ    4780E090 1D13B2
:1DFE98 BR    07FF      020000
```

Note: If control is transferred to the bad address by a LPSW or an interrupt (for example an SVC) PER BRANCH does not trace this event. Therefore, it is a good idea to issue a TRACE PROG before starting the program. Then, if the program interrupt occurs before any PER output is produced, the PER TABLE command can be used to display the branch traceback table containing the last 6 successful branches. The last entry in the table is the last successful branch instruction executed before the program interrupt. While this is not necessarily the instruction causing the problem, hopefully it is near the failing instruction. It is now possible to restart the program using PER to trace the execution of instructions in the range beginning with this branch instruction, and ending at the program interrupt address.

The PER COUNT Subcommand

Another method of finding the failing instruction is to use the PER COUNT sub-command with TRACE. This method, as well as the use of the PER TABLE command, is well suited for problems other than just operation exceptions. If the program is abending with any sort of program exception, then load the failing program, and issue the CP command:

TRACE PROG

followed by:

PER INSTRUCT RANGE 20000.500 (assuming the program is X'500' bytes in length)

and then:

PER COUNT

Next start the failing program. No trace output from PER is produced while the COUNT option is in effect. When the program interrupt occurs, issue the QUERY PER command to display the current count:

QUERY PER

You may see the following:

```
1 INSTRUCT RANGE 020000-0204FF TERMINAL NORUN
  PER COUNT 2159
```

This means that 2159 instructions were executed before the instruction that caused the program interrupt. It is now possible to trace as many instructions leading up to the program interrupt as desired. To trace the last 15 instructions before the program interrupt, reload the failing program, and issue the following PER command:

PER PASS 2144

the response is:

```
PER COUNT 2159
PER COUNT ENDED
```

This command has two effects. First, it turns off the PER COUNT option, and second it applies the PASS option to the current traceset. The current traceset now contains:

```
1 INSTRUCT RANGE 020000-0204FF TERMINAL NORUN PASS 2144
```

Next start the failing program. The first 2144 instructions executed in the range X'20000' through X'204FF' are not displayed. The 2145th instruction is displayed. When the instruction is displayed, issue:

PER PASS

This command resets the PASS option to the default (display every instruction). The current traceset now contains:

```
1 INSTRUCT RANGE 020000-0204FF TERMINAL NORUN
```

It is now possible to trace the last 15 instructions, and to use the DISPLAY command to display storage and register contents.

PER COUNT can also be used in conjunction with more specific trace elements to produce the desired results. For example, if a problem occurs as a result of the execution of an SVC 202 and the failing program issues many SVC 202s before failing, it may not be productive to use TRACE.

An alternative is to use PER to set up a traceset that traces only SVC 202s (op code X'0ACA') and to use PER COUNT to count the occurrences. First, load the failing program and then issue:

PER INSTRUCT 0ACA RANGE 20000.500
PER COUNT

and start the program. When the failure occurs, issue a QUERY PER to check the count.

QUERY PER

You may see the following:

```
1 INSTRUCT 0ACA RANGE 020000-0204FF TERMINAL NORUN
  PER COUNT 623
```

The program can then be traced after using the PER PASS option as above to get close to the problem.

The PER Command Option

The PER CMD option can be used to execute any CP command (except SLEEP) whenever a particular event occurs. For example:

PER INSTRUCT RANGE 20000.500 RUN

PER STORE 204F0-204FF RANGE 20000.500 RUN CMD DISPLAY 204F0-204FF

traces the execution of every instruction in the range X'20000' through X'204FF' and displays the contents of storage at X'204F0' through X'204FF' every time any storage within the range X'204F0' through X'204FF' is altered by an instruction in the range X'20000' through X'204FF'.

Also, the CMD option can be used to cause execution of a program to continue at a specific address whenever a particular event occurs. For example:

PER INSTRUCT RANGE 20000.500 PRINTER
PER BRANCH 0 RUN CMD BEGIN 24F28

causes program execution to continue at location X'24F28' whenever a branch to location 0 occurs. The execution continues after the instruction is displayed. If, when program execution is resumed at location X'24F28', a subsequent branch to zero occurs, execution again begins at location X'24F28'. This can result in a loop. The CMD option can also be used to prevent this. For example, if LINEDIT is on, and the escape character is set to " and the line end character is #, then:

PER INSTRUCT 20000.500 PRINTER
PER BRANCH 0 RUN CMD PER END BRANCH"#BEGIN 24F28

turns off the branch trace element and causes program execution to continue at location X'24F28' after the instruction is displayed. The current traceset would then be:

```
1 INSTRUCT RANGE 020000-0204FF PRINTER RUN
```

The commands associated with each trace element are executed whenever the event described by the trace element occurs. The commands are executed in the order in which they appear in the traceset. Therefore, if the current traceset is:

```
1 INSTRUCT TERMINAL RUN CMD cmd1#cmd2
2 BRANCH TERMINAL RUN CMD cmd3
3 G TERMINAL RUN CMD cmd4#cmd5
```

and an instruction is executed that alters a register, but does not cause a successful branch, then the CP commands cmd1, cmd2, cmd4, and cmd5 are executed (in that order).

Note: If a CP command is entered while commands are being executed by PER, the output from the commands may be interleaved.

Once the command option has been specified for a particular trace element, the command option remains in effect until the trace element is turned off. However, the command can be changed. For example, if the current traceset contains:

```
1 BRANCH TERMINAL RUN CMD DISPLAY G15
2 G TERMINAL RUN CMD DISPLAY PSW
```

the command associated with the branch trace element can be changed to DISPLAY G14-15 by issuing:

PER BRANCH TERMINAL RUN CMD DISPLAY G14-15

or both commands can be changed to DISPLAY G14-15 by issuing:

PER CMD DISPLAY G14-15

Storage Alteration

PER can be used to trace the alteration of storage in the user's virtual machine. If PER STORE is specified, then whenever an instruction places a value into storage, that event is traced. It is *not* necessary that this value be *different* from the previous value.

It is also possible to monitor the alteration of storage to a specific value. For example:

PER STORE INTO 20100 DATA 112757

monitors instructions that cause the storage at location X'20100' to become X'112757'. Note that these instructions are traced even if the value at location X'20100' was already X'112757' before the execution of the instructions.

It is also possible to monitor only alterations of storage when the storage value actually changes. For example:

PER MASK INTO 20100 DATA FFFFFFFFFFFFFFFF

monitors instructions that actually change one or more bytes in the doubleword field starting at location X'20100'. PER MASK can also be used to monitor changes to specific bits in storage. For example:

PER MASK INTO 20100 DATA 8040

monitors instructions that change the status of the first bit in the byte at location X'20100' or the second bit in the byte at location X'20101'.

GUESTR and GUESTV

PER traces virtual machine activity in both second- and third-level storage. Using the GUESTR and GUESTV options, it is possible to choose which level activity will be traced. For example, when debugging VM/SP within a virtual machine it is sometimes helpful to limit trace output to either just second- or just third-level activity. CP itself runs with Dynamic Address Translation (DAT) off. When CP dispatches a virtual machine, the DAT bit in the PSW is on. Therefore, if CP is IPLed into a virtual machine, then that CP is executing in second-level storage. When CMS is IPLed on top of the second-level CP, then that CMS is executing in third-level storage. If the command:

```
PER I RANGE 20000-21000
```

is issued, then both second- and third-level activity is traced (that is, both CP and CMS). To only trace the CP activity (second level) in the range X'20000' through X'21000', enter:

```
PER I RANGE 20000-21000 GUESTR
```

To only trace the CMS activity (third level) in the range X'20000' through X'21000', enter:

```
PER I RANGE 20000-21000 GUESTV
```

It is also possible to selectively trace specific activity in both second- and third-level storage. For example, to trace successful branches and storage alterations of location X'1024' in second-level storage and to trace branches to location X'20000', and alterations to register 7 in third-level storage, enter:

```
PER BRANCH STORE INTO 1024 GUESTR
PER BRANCH INTO 20000 G7 GUESTV
```

Commands that Alter the Contents of Storage

You can use the STORE and STCP commands to alter the contents of virtual machine storage and real storage.

ZAP and ZAPTEXT commands are used to alter TEXT libraries or TEXT decks before the code is loaded and executed.

STORE and STCP are described in the following subparagraphs. See *VM/SP Installation Guide* or *VM/SP HPO Installation Guide* for information on ZAP and ZAPTEXT.

Altering the Contents of Virtual Machine Storage (STORE command)

Use the STORE command to alter the contents of specified registers and locations in virtual machine storage. The contents of the following can be altered:

- Virtual machine storage locations (first-level virtual storage only)
- General purpose registers
- Floating-point registers
- Vector registers (**VM/SP HPO only**)
- Control registers (if available)
- Program Status Word.

The STORE STATUS command can save certain information contained in low storage.

When debugging, you may find it advantageous to alter storage, registers, or the PSW and then continue execution. This is a good procedure for testing a proposed change. Also, you can make a temporary correction and then continue to ensure that the rest of execution is trouble-free. A procedure for using the STORE STATUS command when debugging is as follows:

- Issue the STORE STATUS command before entering a routine you wish to debug.

- When execution stops (because an address stop was reached or because of failure), display the extended logout area. This area contains the status that was stored before entering the routine.

- Issue STORE STATUS again and display the extended logout area again. You now have the status information before and after the failure. This information should help you solve the problem.

Altering Virtual Storage

You can alter the contents of your first level virtual storage, GPRs, floating-point registers, control registers (if available), and the PSW with the STORE command.

Virtual storage can be altered in either fullword or byte units.

When using fullword units, the address of the first positions to be stored must have either an L or no prefix:

STORE 1024 46A2

-- or --

STORE L1024 46A2

results in X'000046A2' being stored in locations X'1024' through X'1027'.

STORE 1024 46 A2

on the other hand, implies storing 2 fullwords and results in the storing of X'00000046000000A2' in locations X'1024' through X'102B'.

If the starting location is not a multiple of a fullword, it is automatically rounded down to the next lower fullword boundary. Each fullword operand can be from one to eight hexadecimal characters in length. If less than 8 characters are specified, they are right justified in the fullword unit and padded to the left with zeros.

You can store in byte units by prefixing the start address with an S.

STORE S1026 D1D6C5

stores X'D1D6C5' in locations X'1026', X'1027', and X'1028'. Note that the data storage is byte aligned. If an odd number of hexadecimal characters is specified, CP does not store the last character, you receive an error message, and CP terminates the function. For example, if you specify:

STORE S1026 D1D6C

CP stores d1 at X'1026', and d6 at X'1027'; when CP attempts to store c, it recognizes an incomplete hexadecimal character, and does not store the last character.

You can store data into one or multiple consecutive registers.

General and control registers are loaded in fullword units. For example,

STORE G4 123456

loads GPR 4 with X'00123456'.

STORE G4 12 34 56

loads GPRs 4, 5, and 6 with X'00000012', X'00000034', and X'00000056', respectively.

Floating-point registers are loaded in doubleword units. Each doubleword operand can be from 1 to 16 hexadecimal characters in length. If less than 16 characters are specified, they are left justified in the doubleword unit and padded to the right with zeros. For example:

STORE Y2 00123456789

loads floating-point register 2 with the value X'0012345678900000'.

You can use the STATUS operand of the STORE command to simulate the hardware store status facility. Selected virtual machine data is stored in permanently assigned areas in low storage. Your virtual machine must be in extended control mode for the command:

STORE STATUS

to be accepted. To place your virtual machine in extended control mode, issue the command:

SET ECMODE ON

Be aware that this command resets your virtual machine and you must reload (IPL) your operating system.

The data stored by the STORE STATUS command is summarized in the following table:

Virtual Dec.	**Address Hex**	**No. of Bytes**	**Data**
216	D8	8	Processor Timer
224	E0	8	Clock Comparator
256	100	8	Current PSW
352	160	32	Floating-Point Registers (0,2,4, and 6)
384	180	64	General Purpose Registers (0-15)
448	1C0	64	Control Registers (0-15)

Note: If the operating system that is running in your virtual machine operates in the basic control mode, these areas of low storage may be used for other purposes. You should not use this facility under these conditions.

Altering the Contents of Real Storage (STCP command)

Use the STCP command to alter the contents of real storage. The STCP command can alter the real PSW or real registers, but the user has to know where in storage these are located.

Chapter 3. Debugging CP

Commands to Collect and Analyze System Information

The following commands are used to collect and analyze system information when debugging:

- LOCATE
- MONITOR
- INDICATE
- QUERY SRM.

The MONITOR command provides a data collection tool that samples and records a wide range of data. The INDICATE command provides a method to observe the load conditions on the system while it is running. The QUERY SRM command provides observation facilities for analyzing internal activity counters and parameters.

See *VM/SP CP for System Programming* or *VM/SP HPO CP for System Programming* and *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for more information on the MONITOR, INDICATE, and QUERY SRM commands.

Locating CP Control Blocks in Storage

Use the class C or E LOCATE command to find the address of CP control blocks associated with a particular user, a user's device, or a real system device. The control blocks and their functions are described in the *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP.*

Once you know the location of the control blocks, you can examine the block you want to look at. When you want to examine specific control blocks, use the LOCATE and the DCP command to display or the DMCP command to print the control blocks. A discussion of the most important fields of the VMBLOK, VCUBLOK, VDEVBLOK, RCHBLOK, RCUBLOK, and RDEVBLOK are included in "Reading CP Abend Dumps" on page 76.

Debugging CP in a Virtual Machine

Many CP problems can be isolated without stand-alone machine testing. It is possible to debug CP by running it in a virtual machine. In most instances, the virtual machine system is an exact replica of the system running on the real machine. To set up a CP system on a virtual machine, use the same procedure that is used to generate a CP system on a real machine. However, remember that the entire procedure of running service programs is now done on a virtual machine. Also, the virtual machine must be described in the real directory. See *VM Running Guest Operating Systems* for directions on how to set up the virtual machine.

CP Internal Trace Table

CP has an internal trace table that records events that occur in the real machine. The events that are traced are:

- External interrupts
- SVC interrupts
- Program interrupts
- Machine check interrupts
- I/O interrupts
- Free storage requests
- Release of free storage
- Entry into scheduler
- Queue drop
- Run user requests
- Start I/O
- Unstack I/O interrupts
- Storing a virtual CSW
- Test I/O
- Halt Device
- Unstack IOBLOK or TRQBLOK
- Network Control Program (NCP) Basic Transmission Unit (BTU)
- Spinning on a lock (AP or MP environment)
- SIGP issued
- Clear Channel instruction
- IUCV communications
- SNA Console Communication Services (CCS)
- MSSF DIAGNOSE X'80'
- Start I/O fast release
- Simulated I/O interruptions
- Run user through fast path **(VM/SP HPO only)**
- Clear I/O
- Resetting virtual machine pages dropped from a queue **(VM/SP HPO only)**
- Logical swap-in data **(VM/SP HPO only)**
- Obtain prime storage **(VM/SP HPO only)**
- Return prime storage **(VM/SP HPO only)**
- Test channel.

An installation may optionally specify the size of the CP internal trace table. To do so, use the SYSCOR macro in module DMKSYS. Information on using this macro instruction is in the *VM/SP Planning Guide and Reference* or *VM/SP HPO Planning Guide and Reference.*

If an installation does not specify the CP internal trace table size or the size specified is smaller than the default size, VM/SP CP assigns the default size. In VM/SP HPO, an installation can specify a CP internal trace table size that is smaller than the default size.

Type of Event	Module	ID Code[1] (hex)	Format of Trace Table Entry															
			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
External Interrupt	DMKEXT	01	X'01'	5X'00'					Interrupt Code		External Old PSW							
SVC Interrupt	DMKSVC	02	X'02'	GPR 14 or GPR 15[2]			Instruction Length Code[5]		SVC Interrupt Code[5]		SVC Old PSW							
Program Interrupt	DMKPMA[7] DMKPRG	03	X'03'	First 3 bytes of VMPSW			Instruction Length Code[5]		Interrupt Code		Program Old PSW							
Machine Check Interrupt	DMKMCH	04	X'04'	VMBLOK Address			First 4 bytes of 8-byte Interrupt Code				Machine Check Old PSW							
I/O Interrupt	DMKIOT	05	X'05'	Measurement Byte	Device Address		I/O Old PSW + 4				CSW							
Obtain Storage (FREE)	DMKFRE	06	X'06'	VMBLOK Address			GPR 0 at entry				GPR 1 at exit				GPR 14			
Return Storage (FRET)	DMKFRE DMKFRT	07	X'07'	VMBLOK Address			GPR 0 at entry				GPR 1 at entry				GPR 14			
Enter Scheduler	DMKSCH	08	X'08'	VMBLOK Address			VMR-STAT	VMD-STAT	VMO-STAT	VMQ-STAT	VMQ-LEVEL	VMT-LEVEL	VMIOINT		VMPEND	GPR 14		
Queue Drop	DMKSCH	09	X'09'	VMBLOK Address			VMEPRIOR				DMK-SCHAL	VMU-PRIOR	VMQPRIOR		VMUHS			
Run User	DMKDSP	0A	X'0A'	3X'00'			RUNUSER value from PSA				RUNPSW value from PSA							
Start I/O	DMKACS DMKCNS DMKCPM DMKCPU[7] DMKIOS DMKIOT[7] DMKMNT	0B	X'0B'	CC	Device Address		IOBLOK Address				CAW				For CC = 1, CSW + 4. Otherwise, this field is 4X'00'.			
Unstack I/O Interrupt	DMKDSP	0C	X'0C'	X'00'	Virtual Device Address		VMBLOK Address				Virtual CSW							
Virtual CSW Store	DMKVSJ	0D	X'0D'	Instruction OP-Code	Virtual Device Address		VMBLOK Address				Virtual CSW							
Test I/O	DMKACS DMKCPM DMKCPU[7] DMKIOS DMKIOT[7] DMKMNT	0E	X'0E'	CC	Device Address		IOBLOK Address				CAW				For CC = 1, CSW + 4. Otherwise, this field is 4X'00'.			
Halt Device	DMKCNS DMKIOS DMKMNT DMKVSJ	0F	X'0F'	CC	Device Address		IOBLOK Address				CAW				For CC = 1, CSW + 4. Otherwise, this field is 4X'00'.			
Unstack IOBLOK or TRQBLOK	DMKDSP	10	X'10'	VMBLOK Address			VMR-STAT	VMD-STAT	VMO-STAT	VMQ-STAT	IOBLOK or TRQBLOK Address				Interrupt Return Address			
NCU BTU[3]	DMKRNH	11	X'11'	X'00'	CONSRID		CONDEST		CONRTAG		CON-SYSR	CON-EXTR	CONTCMD		CON-FUNC	CON-DFLG	CONDCNT	
Spinning on Lock	DMKLOK	12	X'12'	VMBLOK Address			Lockword Address				Return Address				Lockword Contents			
SIGP Issued	DMKEXT	13	X'13'	Return Address			X'00'	CC	Real Processor Address		Function Code[4]	Order Code			Status of CC = 1			
Clear Channel Issued	DMKVSJ	14	X'14'	CC	Device Address		VMBLOK Address				Virtual CSW							
IUCV Communication	DMKIUA	15	X'15'	Function Code[4]	Path Id or Unused		IUCVBLOK Address				Use varies by Function Code (See *VM System Facilities for Programming (SC24-5288)* for details on IUCV trace table format.)					IUCV Instruction Address		
SNA CCS	DMKVCV DMKVCX	16	X'16'	Transaction Type	See *VM System Facilities for Programming (SC24-5288)* for details on SNA CCS entries.													
MSSF DIAGNOSE X'80'	DMKMHC	17	X'17'	CC	2X'00'		HCBLOK Address				MSSF Command Word				MSFBLOK Address			
Start I/O Fast Release	DMKIOS	18	X'18'	CC	Device Address		IOBLOK Address				CAW				For CC =1, CSW + 4. Otherwise, this field is 4X'00'.			
Simulated I/O Interrupt	DMKIOT	19	X'19'	X'00'	Device Address		DMKDID or DMKACS Address in GPR 12 at entry				CSW							
Run User Through Fast Path[6]	DMKDSP	1A	X'1A'	3X'00'			RUNUSER value from PSA				RUNPSW value from PSA							
Clear I/O	DMKIOS	1B	X'1B'	CC	Device Address		IOBLOK Address				CAW				For CC = 1, CSW + 4. Otherwise, this field is 4X'00'.			
Resetting VM Pages[6]	DMKPTS	1C	X'1C'	VMBLOK Address			VMR-STAT	VMD-STAT	VMO-STAT	VMQ-STAT	VMS-WSTAT	VMS-WPFL1	VMS-WPFL2	X'00'	GPR 14			
Logical Swap-In[6]	DMKSWAP1	1D	X'1D'	VMBLOK Address			VMR-STAT	VMD-STAT	VMO-STAT	VMQ-STAT	VMQ-LEVEL	VMT-LEVEL	VMS-WPL1	VMS-WSTAT	VMRSWPGS		SCBNWS	
Obtain Prime Storage[6]	DMKFRE	1E	X'1E'	VMBLOK Address			GPR 0 at entry				GPR 1 at exit				GPR 14			
Return Prime Storage[6]	DMKFRE DMKFRT	1F	X'1F'	VMBLOK Address			GPR 0 at entry				GPR 1 at exit				GPR 14			
		20	Unused															
Test Channel	DMKIOS	21	X'21'	CC	Device Address		IOBLOK Address				CAW				Not Used			
			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Notes:
1. If the installation is running in AP or MP mode, the identification code will be ORed with a X'40' if the activity occurred on the non-IPL processor. If the installation is running ECPS, the identification code is ORed with an X'80' if the activity occurred in microcode.
2. If the interrupt code (bytes 6 and 7) is X'000C', the contents of GPR 14 are displayed. For all other interrupt codes, the contents of GPR 15 are displayed.
3. Bytes 2 through 15 of a code 11 trace record represent a Basic Transmission Unit, sent or received by a 37XX. If CONSYSR/CONEXTR are zero, the BTU was transmitted to the 37XX. If they are non-zero, the BTU was received. If CONTCMD equals X'7700', this is an unsolicited BTU response.
4. For the SIGP instruction, trace table entry 13 byte 8 contains the function code for emergency signal and external call order codes.
5. For VM/SP HPO without ECPS, byte 4 contains the Interrupt Code and bytes 5, 6, and 7 contain GPR 13. For SVC 8 and SVC 20, GPR 13 on exit. For SVC 12 and SVC 16, GPR 13 on entry.
6. These trace table entries are for VM/SP HPO only.
7. These modules are for VM/SP HPO only.

Figure 8. VM CP Internal Trace Table

For each 256K bytes (or part thereof) of real storage available at IPL time, one page (4096 bytes) is allocated to the CP internal trace table. Each entry in the CP internal trace table is 16 bytes long. There are CP internal trace table entries for each type of event recorded. The first byte of each CP internal trace table entry, the identification code, identifies the type of event being recorded. Figure 8 on page 73 describes the format of each type of CP internal trace table entry. See *VM CP Internal Trace Table (Poster)* for a 20 inch by 26 inch poster of the CP internal trace table. Also see the *VM Problem Determination Reference Information* for a reference card that may be easily carried that shows the CP internal trace table. The entry shown in Figure 8 on page 73 for IUCV communications illustrates the general format of an IUCV entry. See *VM System Facilities for Programming* for the formats of the CP internal trace table entries for each IUCV function, and for a description of each field in the CP internal trace table entry.

In addition, some CP internal trace table entries are generated by ECPS:VM/370. The first bit of these entries is set to 1 to indicate the entry was generated by the hardware assist. For example, a CP internal trace table entry of type X'86' (FREE) is the same as an entry of type X'06'. The only difference is that the first entry was generated by the hardware assist.

The CP internal trace table is allocated by DMKSTA which is called by the main initialization routine, DMKCPI. The first event traced is placed in the lowest CP internal trace table address. Each subsequent event is recorded in the next available CP internal trace table entry. Once the CP internal trace table is full, events are recorded at the lowest address (overlaying the data previously recorded there). Tracing continues with each new entry replacing an entry from a previous cycle.

Use the CP internal trace table to determine the events that preceded a CP system failure. An abend dump contains the CP internal trace table and the pointers to it. The address of the start of the CP internal trace table, TRACSTRT, is at location X'0C'. The address of the byte following the end of the CP internal trace table, TRACEND, is at location X'10'. The address of the next available CP internal trace table entry, TRACCURR, is at location X'14'. Subtract 16 bytes (X'10') from the address stored at X'14' (TRACCURR) to obtain the CP internal trace table entry for the last event completed.

The CP internal trace table is initialized during IPL. If you do not wish to record events in the CP internal trace table, issue the MONITOR STOP command to suppress recording. The pages allocated to the CP internal trace table are not released and recording can be restarted at any time by issuing the MONITOR START command. If the VM/SP system should abnormally terminate and automatically restart, the tracing of events on the real machine will be active. After a VM/SP IPL (manual or automatic), CP internal tracing is always active.

Abend Dumps

There are three kinds of abnormal termination dumps possible when using CP. If the problem program cannot continue, it terminates and, in some cases, tries to issue a dump. Likewise, if the operating system for your virtual machine cannot continue, it terminates and, in some cases, tries to issue a dump. In the virtual machine environment, the problem program dump always goes to the virtual printer. Depending on installation operating procedures, the virtual machine operating system dump may also go to the virtual printer. A CLOSE must be issued to the virtual printer to have either dump print on the real printer.

The third type of dump occurs when the CP system cannot continue. CP abend dumps can be directed to a:

- Printer
- Tape
- DASD.

If the dump is directed to a printer, then the data is printed online and system operations as well as virtual machine operations are suspended until the dump operation finishes. This dump is unformatted.

If the dump is directed to a tape, the dumped data must fit on one reel of tape. VM/SP does not support multiple tape volumes. The data on the tape is in print-line format and can be processed by user-created programs or CMS commands. See "How to Print a CP Abend Dump from Tape" on page 76 for an example of how CMS can do this.

If the dump is directed to DASD, it is done by spooling the data to the virtual card reader of a specific userid. The userid is either assigned during system generation to a specific virtual machine user, or defaults to the printer. The dump spool file can be manipulated by the user just like any other spool file, except that it can be interpreted correctly only by the IPCSDUMP command (for more details see "IPCSDUMP Command" on page 233). By issuing a QUERY DUMP you can determine where the dump is being directed.

The extent of the CP abend dump can be specified to dump:

- CP storage only
- CP and all real storage

to the selected device.

Use the CP SET DUMP command to specify the output device and extent of CP abend dumps. Refer to the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for the format of the SET DUMP command.

How to Print a CP Abend Dump from Tape

When the CP abend dump is sent to a tape, the records are 132 characters long, unblocked, with a blocksize of 132 and carriage control characters.

If the CP dump unit has been specified as a tape drive, and one or more dumps have been placed on the tape, use the following procedure to print the dumps:

1. Log on to the system with any userid that has the capability of running CMS. No other special privilege classes or options are required.

2. Attach a tape drive to the virtual machine as address 181.

3. Mount the tape that has the CP abend dumps.

4. IPL the CMS system and perform the usual access requirements.

5. Issue the following CMS commands:

   FILEDEF *dname1* **PRINTER (RECFM FM LRECL 132)**
   FILEDEF *ddname2* **TAP1 (DEN 1600 RECFM U LRECL 132)**
   MOVE *ddname2 ddname1*
   CP CLOSE PRT

   Note: Refer to the FILEDEF command description in *VM/SP CMS Command Reference.*

Step 5 can be repeated for as many dumps as are on the tape. Note that the CP dump routines write two tape marks at the end of each file. Therefore, to process the next dump, the TAPE FSF command must be issued to position the tape for reading the next dump file.

Reading CP Abend Dumps

Two types of printed dumps occur when CP abnormally ends, depending upon the options specified in the CP SET DUMP command. When the dump is directed to a direct access device, Interactive Problem Control System (IPCS) must be used to format and print the dump. For IPCS use, see Chapter 7, "Debugging Using IPCS" on page 197. IPCS commands format and print:

- Control blocks
- General purpose registers
- Floating-point registers
- Control registers
- TOD (Time-of-Day) Clock
- Processor Timer
- Storage (for VM/SP HPO, includes vector control block and save areas)
- If in AP or MP mode, formats and prints both PSAs' storage.

Storage is printed in hexadecimal notation, eight words to the line, with EBCDIC translation at the right. The hexadecimal address of the first byte printed on each line is indicated at the left.

If the CP SET DUMP command directed the dump to tape or the printer, the printed format of the printed dump will not contain formatted control blocks. If the system was an attached processor or multiprocessor, all of the registers, etc., are printed for the abending processor. Also, each PSA is printed before printing main storage.

When CP can no longer continue and abnormally terminates, you must first determine the condition that caused the abend, and then find the cause of that condition. You should know the structure and function of CP. See *VM/SP CP for System Programming* or *VM/SP HPO CP for System Programming* for information that will help you understand the major functions of CP. The following discussion on reading CP dumps includes many references to CP control blocks and control block fields. Refer to *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP* for a description of the CP control blocks. Figure 9 on page 82 shows the CP control block relationships. Also, you will need the current load map for CP to be able to identify the modules from their locations. The load map is created at initial CP generation time. See the *VM/SP Installation Guide* or *VM/SP HPO Installation Guide* for obtaining the original copy of the CP load map.

Reason for the Abend

Determine the immediate reason for the abend. You need to examine several fields in the Prefix Storage Area (PSA), to find the reason for the abend. In a uniprocessor system, the PSA is in page 0. In an Attached Processor (AP) or Multiprocessor (MP) system, each processor has its own PSA in addition to the absolute PSA in page 0.

1. Examine the program old PSW and program interrupt code to find whether or not a program check occurred in CP. The program old PSW (PROPSW) is located at X'28' and the program interrupt code (INTPR) is at X'8E'. If a program check has occurred in supervisor mode, use the CP system load map to identify the module. If you cannot find the module using the load map, refer to "Identifying and Locating a Pageable Module" on page 92. Figure 70 on page 352 describes the format of an Extended Control PSW.

2. Examine the SVC old PSW, the SVC interrupt code, and the abend code to find whether or not a CP routine issued an SVC 0. The SVC old PSW (SVCOPSW) is located at X'20', the SVC interrupt code (INTSVC) is at X'8A', and the abend code (CPABEND) is at X'374'.

   The abend code (CPABEND) is a fullword. The first three bytes identify the module that issued the SVC 0 and the fourth byte is a binary field whose value indicates the reason for issuing an SVC 0.

Use the CP system load map to identify the module issuing the SVC 0. If you cannot find the module using the CP system load map, refer to "Identifying and Locating a Pageable Module" on page 92. Figure 70 on page 352 describes the format of an Extended Control PSW.

3. Examine the old PSW at X'08'. If an abnormal termination occurs because the operator caused a system restart, the old PSW at location X'08' points to the instruction that was executing when CP recognized the abnormal termination. Figure 70 on page 352 describes the format of an Extended Control PSW.

4. For a machine check, examine the machine check old PSW and the logout area. The machine check old PSW (MCOPSW) is found at X'30' and the fixed logout area is at X'100'. Also examine the machine check interrupt code (INTMC) at X'E8'.

Collect Information

Examine several other fields in the PSA to analyze the status of the system. As you progress in reading the dump, you may return to the PSA to pick up pointers to specific areas (such as pointers to the real control blocks) or to examine other status fields. For specific fields within the PSA control block, refer to *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP.*

The following areas of the PSA may contain useful debugging information:

The CP running status is stored in CPSTAT at location X'348'.

1. CP running status field (CPSTAT) The value of this field indicates the running status of CP since the last entry to the dispatcher.

2. Current User

   The PSW that was most recently loaded by the dispatcher is saved in RUNPSW at location X'330', and the address of the dispatched VMBLOK is saved in RUNUSER at location X'338'. Also, examine the contents of control registers 0 and 1 as they were when the last PSW was dispatched. See RUNCR0 (X'340') and RUNCR1 (X'344') for the control registers.

Also, examine the CP internal trace table to determine the events that preceded the abnormal termination. Start with the last event recorded in the trace table and read backward through the trace table entries. The last event recorded is the last event that was completed.

The TRACSTRT field (location X'0C') contains the address of the start of the trace table. The TRACEND field (location X'10') contains the address of the byte following the end of the trace table. The address of the next available trace table entry is found in the TRACCURR field (location X'14'). To find the last recorded trace table entry, subtract X'10' from the value at location X'14'. The result is the address of the last recorded entry. Figure 8 on page 73 describes the format of each type of trace table entry.

Note: If the system was in AP or MP mode, the trace table pointers are in absolute page zero.

Register Use

To trace control blocks and modules, it is necessary to know the CP register-use conventions.

The 16 General Purpose Registers (GPRs) have many uses that vary depending upon the operation. The following table shows the use of some of the general purpose registers:

Register	Contents
GPR 1	The virtual address to be translated.
GPR 2	The real address or parameters.
GPR 6,7,8	The virtual or real channel, control unit, and device control blocks.
GPR 10	The address of the active IOBLOK.
GPR 14, 15	The external branch linkage.

The following general purpose registers usually contain the same information:

Register	Contents
GPR 11	The address of the active VMBLOK.
GPR 12	The base register for the module executing.
GPR 13	The address of the current save area if the module was called via an SVC.

Use these registers along with the CP control blocks and the data in the prefix storage area to determine the error that caused the CP abend.

Save Area Conventions

The save areas that may be helpful in debugging CP are: SAVEAREA, BALRSAVE, FREESAVE, FREEWORK, and DUMPSAVE. If a module was called by an SVC, examine the SAVEAREA storage area. SAVEAREA is not in the PSA; the address of the SAVEAREA is found in general purpose register 13. If a module was called by a branch and link, the general purpose registers are saved in the PSA in an area called BALRSAVE (X'240'). The work area and save area for DMKFRE and DMKFRT are also in the PSA; these areas are used only by the DMKFRE and DMKFRT routines. The save area (FREESAVE) for DMKFRE and DMKFRT can be found at location X'280' and the work area (FREEWORK) follows at location X'2C0'.

Save areas used by attached processor and multiprocessor support are SIGSAVE, LOKSAVE, MFASAVE, SWTHSAVE, LOCKSAVE, and SVCREGS. These save areas are all in the PSA. All except LOCKSAVE and SVCREGS are 16 words in size.

Use the save areas to trace backwards and find the previous module executed.

1. SAVEAREA

   An active save area contains the caller's return address in SAVERETN (displacement X'00'). The caller's base register is saved in SAVER12 (displacement X'04'), and the address of the save area for the caller is saved.

2. BALRSAVE

   All the general purpose registers are saved in BALRSAVE after branching and linking (via BALR) to another routine. Look at BALR14 for the return address saved, BALR13 for the caller's save area, and BALR12 for the caller's base register, and you can trace module control backwards.

3. FREESAVE

   All the general purpose registers are saved in FREESAVE before entries in DMKFRE or DMKFRT execute. Use this address to trace module control backwards.

Field	Contents
FREER15	The entry point (in DMKFRE or DMKFRT).
FREER14	The saved return address.
FREER13	The caller's save area (unless the caller was called via BALR).
FREER12	The caller's base register.
FREER1	Points to the block returned (for FRET entries).
FREER0	Contains the number of doublewords requested or returned.

4. FREEWORK (**VM/SP HPO only**)

   The DMKFRE and DMKFRT work areas are also in the PSA. It follows the FREESAVE area.

5. DUMPSAVE

   All the general purpose registers at the time of the error are saved in DUMPSAVE (displacement X'500') before DMKDMP is called. They are saved by DMKPSA after a restart, by DMKSVC after an SVC 0, and by DMKPRG. The registers are stored in DUMPSAVE in the order GPR0 through GPR15. GPR12 usually contains the base register for the module executing at the time of the error.

6. SIGSAVE

   SIGSAVE (displacement X'540') is used as a save/work area by DMKEXT, a MP/AP-only module that handles all signaling requests. When a signal request is issued, DMKEXTSP is called. On entry, DMKEXTSP stores GPR12 through GPR15, and GPR0 through GPR6. GPR7 through GPR11 are not saved. The remainder of SIGSAVE is used as a work area. GPR14 contains the caller's return address.

7. LOKSAVE

   All the general purpose registers are stored in LOKSAVE (displacement X'580') before DMKLOK executes. DMKLOK is an MP/AP-only module that manipulates certain locks. The registers are stored in the order GPR0 through GPR15. GPR14 contains the caller's return address.

8. MFASAVE

   All the general purpose registers are stored in MFASAVE (displacement X'5C0') before DMKMCTMA executes. DMKMCTMA is the entry into DMKMCT, a MP/AP-only module, that handles malfunction alert interrupts. The registers are stored space in the order GPR0 through GPR15. GPR14 and GPR15 contain the caller's return address.

9. SWTHSAVE

   All the general purpose registers are stored in SWTHSAVE (displacement X'600') by DMKSTK and DMKVMASW. DMKVMASW is an entry that is used only in MP/AP systems to switch a user's page table pointers. The registers are stored in the order GPR0 through GPR15. GPR14 contains the caller's return address. All entries to DMKSTK store registers GPR0 through GPR15 in SWTHSAVE.

10. LOCKSAVE

    LOCKSAVE (displacement X'640') is a four-word save area used by the LOCK macro to save GPR14, GPR15, GPR0, and GPR1 if the SAVE option of the LOCK macro is specified.

11. SVCREGS

    SVCREGS (displacement X'650') is a four-word save area used to save GPR12 through GPR15 at the time of an SVC interrupt.

Virtual and Real Control Block Status

Examine the virtual and real control blocks for more information on the status of the CP system. Figure 9 on page 82 describes the relationship of the CP control blocks; several are described in detail in the following paragraphs. For even more detail on the following control blocks, refer to *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP.*

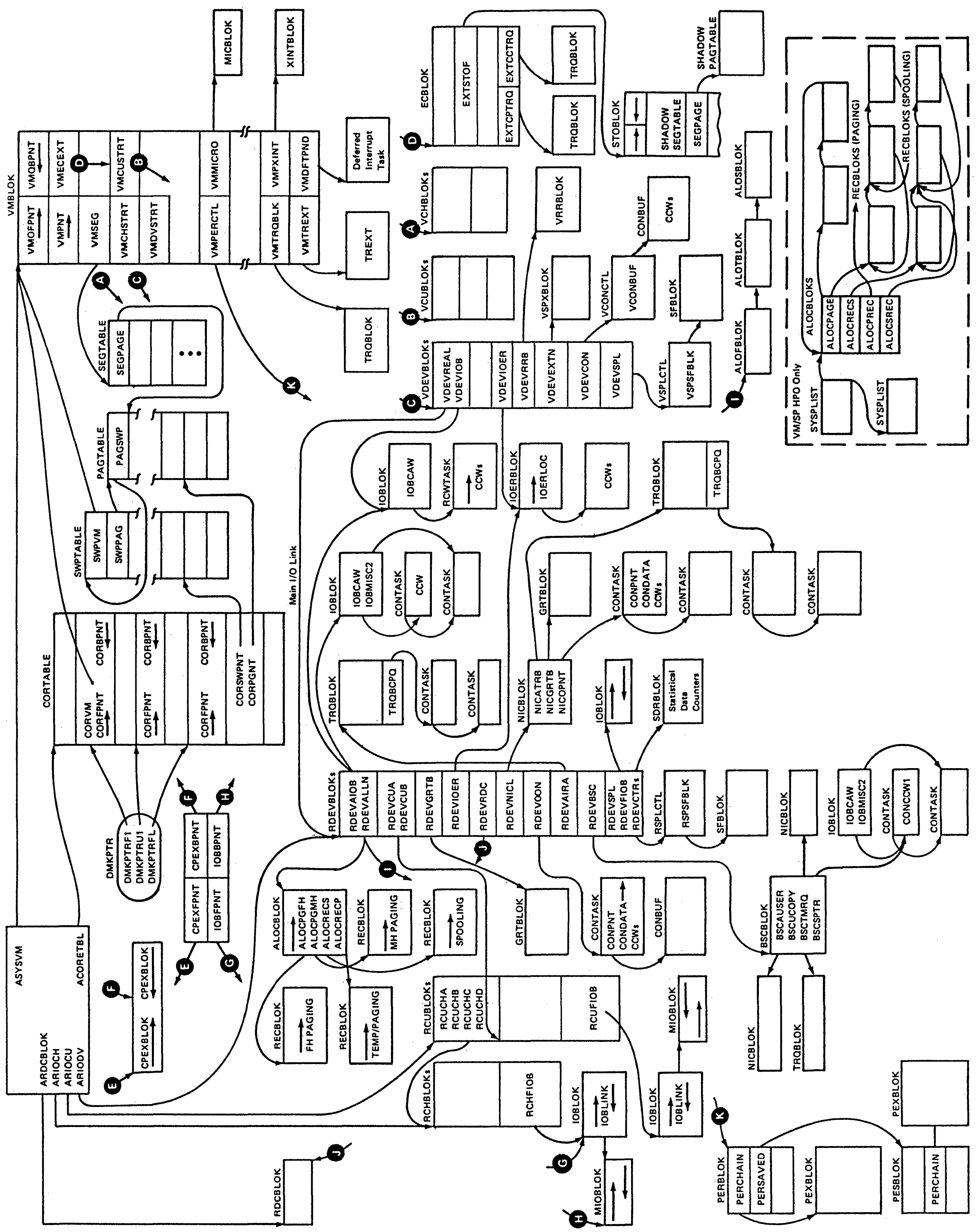


Figure 9. CP Control Block Relationships

VMBLOK

The address of the VMBLOK is in general purpose register 11.

Examine the following VMBLOK fields:

1. VMRSTAT (displacement X'58') contains the virtual machine running status.

2. VMDSTAT (displacement X'59') contains the virtual machine dispatching status.

3. VMPSW (displacement X'A8') saves the virtual machine PSW.

4. VMINST (displacement X'98') saves the virtual machine privileged or tracing instruction.

5. VMCOMND (displacement X'148') contains the name of the last CP command that executed.

6. For checking the status of I/O activity, the following fields contain pertinent information.

    a. VMPEND (displacement X'63') contains the interrupt pending summary flag. The value of VMPEND identifies the type of interrupt.

    b. VMFSTAT (displacement X'68') contains the virtual machine features.

    c. VMIOINT (displacement X'6A') contains the I/O interrupt pending flag. Each bit represents a channel (0 through 15). An interrupt pending is indicated by a 1 in the corresponding bit position.

    d. VMIOACTV (displacement X'36') is the active channel mask. An active channel is indicated by a 1 in the corresponding bit position.

VCHBLOK

The address of the VCHBLOK table is found in the VMCHSTRT field (displacement X'18') of the VMBLOK. General purpose register 6 contains the address of the active VCHBLOK. Examine the following fields:

1. VCHADD (displacement X'00') contains the virtual channel address.

2. VCHSTAT (displacement X'06') contains the status of the virtual channel.

3. VCHTYPE (displacement X'07') contains the virtual channel type.

VCUBLOK

The address of the VCUBLOK table is found in the VCUSTRT field (displacement X'1C') of the VMBLOK. General purpose register 7 contains the address of the active VCUBLOK. Useful information is contained in the following fields:

1. VCUADD (displacement X'00') contains the virtual control unit address.
2. VCUSTAT (displacement X'06') contains the status of the virtual control unit.
3. VCUTYPE (displacement X'07') contains the type of the virtual control unit.

VDEVBLOK

The address of the VDEVBLOK table is found in the VMDVSTRT field (displacement X'20') of the VMBLOK. General purpose register 8 contains the address of the active VDEVBLOK. Useful information is contained in the following fields:

1. VDEVADD (displacement X'00') contains the virtual device address.
2. VDEVSTAT (displacement X'06') contains the status of the virtual device.
3. VDEVFLAG (displacement X'07') contains the device-dependent information.
4. VDEVCSW (displacement X'08') contains the virtual channel status word for the last interrupt.
5. VDEVREAL (displacement X'24') is the pointer to the real device block, RDEVBLOK.
6. VDEVIOB (displacement X'34') is the pointer to the active IOBLOK.
7. VDEVCFLG (displacement X'26') describes the virtual console flags for console devices.
8. VDEVSFLG (displacement X'27') describes the virtual spooling flags for spooling devices.
9. VDEVEXTN (displacement X'10') is the pointer to the virtual spool extension block, VSPXBLOK, for output spooling devices.
10. VDEVFLG2 (displacement X'38') describes the Reserve/Release flags and other miscellaneous conditions.
11. VDEVRRB (displacement X'3C') contains the address of the VRRBLOK for Reserve/Release minidisks.

RCHBLOK

The address of the first RCHBLOK is found in the ARIOCH field (displacement X'3B4') of the PSA. General purpose register 6 contains the address of the active RCHBLOK. Examine the following fields:

1. RCHADD (displacement X'00') contains the real channel address.

2. RCHSTAT (displacement X'04') describes the status of the real channel.

3. RCHTYPE (displacement X'05') describes the real channel type.

4. RCHFIOB (displacement X'08') is the pointer to the first IOBLOK in the queue and RCHLIOB (displacement X'0C') is the pointer to the last IOBLOK in the queue.

RCUBLOK

The address of the first RCUBLOK is found in the ARIOCU field (displacement X'3B8') of the PSA. General purpose register 7 points to the current RCUBLOK. Examine the following fields:

1. RCUADD (displacement X'00') contains the real control unit address.

2. RCUSTAT (displacement X'04') describes the status of the control unit.

3. RCUCHA (displacement X'10') is the pointer to the Primary RCHBLOK.

4. RCUCHB (displacement X'14') is the pointer to the first alternate RCHBLOK.

5. RCUCHC (displacement X'18') is the pointer to the second alternate RCHBLOK.

6. RCUCHD (displacement X'1C') is the pointer to the third alternate RCHBLOK.

7. RCUTYPE (displacement X'05') describes the type of the real control unit.

8. RCUFIOB (displacement X'08') points to the first IOBLOK in the queue and the RCULIOB field (displacement X'0C') points to the last IOBLOK in the queue.

9. RCUOWNER (**VM/SP HPO only**) (displacement X'50') is the userid of cache owner (3880-13 or 3880-23).

RDEVBLOK

The address of the first RDEVBLOK is found in the ARIODV field (displacement X'3BC') of the PSA. General purpose register 8 points to the current RDEVBLOK. Also, the VDEVREAL field (displacement X'24') of each VDEVBLOK contains the address of the associated RDEVBLOK. Examine the following fields of the RDEVBLOK:

1. RDEVADD (displacement X'00') contains the real device address.
2. RDEVSTAT (displacement X'04'), RDEVSTA2 (displacement X'45'), and RDEVSTA4 (displacement X'60') describe the status of the real device.
3. RDEVFLAG (displacement X'05') indicates device flags. These flags are device-dependent.
4. RDEVTYPC (displacement X'06') describes the device type class and the value of the RDEVTYPE field (displacement X'07') describes the device type. Refer to Figure 10 on page 87 for the list of possible device type class and device type values.
5. RDEVAIOB (displacement X'24') contains the address of the active IOBLOK.
6. RDEVUSER (displacement X'28') is the pointer to the VMBLOK for a dedicated user.
7. RDEVATT (displacement X'2C') contains the attached virtual address.
8. RDEVIOER (displacement X'48') contains the address of the IOERBLOK for the last CP error.
9. For spooling unit record devices, RDEVSPL (displacement X'18') is the pointer to the active RSPLCTL block.
10. For real 370X Communications Controllers, several pointers are defined. RDEVEPDV (displacement X'1C') is the pointer to the start of the free RDEVBLOK list for EP lines. RDEVNICL (displacement X'38') is the pointer to the network control list and RDEVCKPT (displacement X'3C') is the pointer to the CKPBLOK for re-enable. Also, RDEVMAX (displacement X'2E') is the highest valid NCP resource name and RDEVNCP (displacement X'30') is the reference name of the active 3705 NCP.
11. For terminal devices, additional flags are defined. RDEVTFLG (displacement X'3A') describes the additional flags.
12. For terminals, an additional flag is defined. RDEVTMCD (displacement X'34') describes the line code translation to be used.

DEVICE CLASS CODES

Code	**Device Class**
X'80'	Terminal Device
X'40'	Graphics Device
X'20'	Unit Record Input Device
X'10'	Unit Record Output Device
X'08'	Magnetic Tape Device
X'04'	Direct Access Storage Device
X'02'	Special Device
X'01'	Fixed-Block Storage

DEVICE TYPE CODES

- For Terminal Device Class

Code	**Device Type**
X'80'	Binary Synchronous Line for Remote
X'40'	2700 Binary Synchronous Line
X'40'	2955 Communication Line
X'30'	Start/Stop Console
X'20'	Telegraph Terminal Control Type II
X'20'	Teletype Terminal
X'1C'	Undefined Terminal Device
X'18'	IBM 2741 Communication Terminal
X'14'	IBM 1050 Data Communication System
X'10'	IBM Terminal Control Type I
X'08'	Synchronous Data Link Control
X'00'	IBM 3210 Console
X'00'	IBM 3215 Console
X'00'	IBM 2150 Console
X'00'	IBM 1052 Console

Figure 10 (Part 1 of 5). CP Device Classes, Types, Models, and Features

- For Graphics Device Class

Code	Device Type
X'C0'	High Function Graphics Device
X'80'	IBM 2250 Display Unit
X'40'	IBM 2260 Display Station
X'20'	IBM 2265 Display Station
X'10'	IBM 3066 Console
X'08'	IBM 1053 Printer
X'04'	IBM 3138 System Console
X'04'	IBM 3148 System Console
X'04'	IBM 3158 System Console
X'04'	IBM 3277 Display Station
X'01'	IBM 3278 Display Station
X'01'	IBM 3279 Display Station
X'01'	IBM 3290 Information Panel
X'02'	IBM 3284 Printer
X'02'	IBM 3286 Printer
X'02'	IBM 3287 Printer
X'02'	IBM 3288 Printer
X'02'	IBM 3289 Printer
X'02'	IBM 4250 Printer

- For Unit Record Input Device Class

Code	Device Type
X'90'	IBM 2520 Card Reader/Punch
X'88'	IBM 1442 Card Reader/Punch
X'84'	IBM 3505 Card Reader
X'82'	IBM 2540 Card Reader
X'81'	IBM 2501 Card Reader
X'80'	Card Reader
X'40'	Timer
X'24'	IBM 1017 Paper Tape Reader
X'22'	IBM 2671 Paper Tape Reader
X'21'	IBM 2495 Magnetic Tape Cartridge Reader
X'20'	Tape Reader

Figure 10 (Part 2 of 5). CP Device Classes, Types, Models, and Features

- For Unit Record Output Device Class

Code	Device Type
X'90'	IBM 2520 Card Punch
X'88'	IBM 1442 Card Punch
X'84'	IBM 3525 Card Punch
X'82'	IBM 2540 Card Punch
X'80'	Card Punch
X'4D'	IBM 3800 Model 8 Printing Subsystem
X'4B'	IBM 4248 Printer
X'4A'	IBM 4245 Printer
X'49'	IBM 3800 Model 3 Printing Subsystem
X'47'	IBM 3262 Printer
X'46'	IBM 3289 Printer
X'45'	IBM 3800 Model 1 Printing Subsystem
X'44'	IBM 1443 Printer
X'43'	IBM 3203 Printer
X'42'	IBM 3211 Printer
X'41'	IBM 1403 Printer
X'40'	Printer
X'24'	IBM 1018 Paper Tape Punch
X'20'	Tape Punch

- For Magnetic Tape Device Class

Code	Device Type
X'82'	IBM 3422 Tape Drive
X'80'	IBM 2401 Tape Drive
X'40'	IBM 2415 Tape Drive
X'20'	IBM 2420 Tape Drive
X'10'	IBM 3420 Tape Drive
X'08'	IBM 3410/3411 Tape Drive
X'04'	IBM 8809 Tape Drive
X'02'	IBM 3430 Tape Drive
X'01'	IBM 3480 Tape Drive

Figure 10 (Part 3 of 5). CP Device Classes, Types, Models, and Features

- For Direct Access Storage Device Class

Code	Device Type
X'80'	IBM 2301 Parallel Drum
X'80'	IBM 2303 Serial Drum
X'80'	IBM 2311 Disk Storage Drive
X'80'	IBM 2321 Data Cell Drive
X'40'	IBM 2314 Disk Storage Facility
X'40'	IBM 2319 Disk Storage Facility
X'20'	IBM 3380 Disk Storage Facility
X'10'	IBM 3330 Disk Storage Facility
X'10'	IBM 3333 Disk Storage and Control
X'08'	IBM 3350 Disk Storage Facility
X'04'	IBM 3375 Disk Storage Facility
X'02'	IBM 2305 Fixed Head Storage Device
X'01'	IBM 3340 Disk Storage Facility

- For Special Device Class

Code	Device Type
X'80'	Channel-to-Channel Device (CTCA or 3088)
X'40'	370X Programmable Communications Controller
X'20'	3851 Mass Storage Controller
X'04'	SRF (7443) device
X'01'	Device unsupported by VM/SP or VM/SP HPO

- For Fixed-Block Storage Device Class

Code	Device Type
X'02'	3370, Model A1, A2, B1, and B2
X'01'	3310
X'00'	Generic Fixed-Block (see Note)

Note: Code X'00' applies to a device whose specific type CP has not yet determined. The proper bit value is assigned when a 'Read Device Characteristics' command is issued at IPL.

MODEL CODES (Column 35 in Accounting Card)

As specified in the RDEVICE macro at system generation.

Note: FB-512 device model codes are specified as:

Code	Device Type
X'00'	3310, 3370 Models A1 and B1
X'04'	3370 Models A2 and B2

Figure 10 (Part 4 of 5). CP Device Classes, Types, Models, and Features

FEATURE CODES (Column 36 in Accounting Card)

- For Printer Devices

Code	**Feature**
X'80'	3800 has four Writable Character Graphic Modifications (WCGM)
X'40'	Extended Sense Bytes
X'01'	UCS

- For Magnetic Tape Devices

Code	**Feature**
X'80'	7 Track
X'40'	Dual Density
X'20'	Translate
X'10'	Data Conversion

- For Direct Access Storage Devices

Code	**Feature**
X'80'	Rotational Position Sensing (RPS)
X'80'	Fixed Head Device
X'40'	Extended Sense Bytes (24 bytes)
X'20'	Top Half of 2314 Used as 2311
X'20'	Device is a 3330V system virtual machine
X'10'	Bottom Half of 2314 Used as 2311
X'08'	35MB Data Module (mounted)
X'04'	70MB Data Module (mounted)
X'02'	Reserve/Release are valid CCW operation codes
X'01'	3330V Virtual MSS volume

- For special devices

Code	**Feature**
X'80'	Type Five Channel Adapter
X'40'	Channel-to-Channel is type 3088
X'20'	Type II channel adapter for 370X
X'10'	Type I channel adapter for 370X

- For terminal devices

Code	**Feature**
X'02'	3270 Mode, Virtual 3215 Device
X'01'	Dial Feature

- For Graphic Devices

Code	**Feature**
X'80'	Operator Identification Card Reader
X'01'	Device Supports WRITE STRUCTURED FIELD QUERY

Figure 10 (Part 5 of 5). CP Device Classes, Types, Models, and Features

Identifying and Locating a Pageable Module

If a program check PSW or SVC PSW points to an address beyond the end of the CP resident nucleus, the failing module is a pageable module. The CP system load map identifies the end of the resident nucleus, labeled as DMKCPEND.

Go to the address indicated in the PSW. Backtrack to the beginning of *that* page frame. The first eight bytes of that page frame (the page frame containing the address pointed to by the PSW) contains the name of the first pageable module loaded into the page. If multiple modules exist within the same page frame, identify the module using the load map and failing address displacement within the page frame. In most cases, register 12 points directly to the name.

To locate a pageable module whose address is shown in the load map, use the system VMBLOK segment and page tables. For example, if the address in the load map is 55000, use the segment and page tables to locate the module at segment 5, page 5.

Debugging an AP/MP System

When you debug an AP/MP problem, the following areas provide pertinent information:

- PSA
- Trace table
- Lockwords.

PSA

A dump for a program operating in AP or MP mode contains three PSAs:

- Absolute PSA
- One PSA (address is in PREFIXA at X'660') for the IPL processor
- One PSA (address is in PREFIXB at X'664') for the other processor.

In a formatted dump, the PSA for the IPL processor is displayed first and the PSA for the other processor is displayed second. The PSA contains important information about the status of each processor, such as:

- Normal low-core IPL
- Logout
- PSW information
- Processor model, type, and features
- BALR areas
- FREE areas
- Monitor and trace data
- Linkages to virtual machines, real devices, and spool files.

See *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP* for layout and explanation of the fields in the PSA.

CP's Page 0 (VM/SP HPO only)

CP's page 0 is relocated in several cases. A preferred machine assist quest gets real page 0. In these cases, CP's absolute page 0 is relocated.

Trace Table

In an AP/MP system, the trace table entries for both processors are intermixed. However, you can identify which processor made a particular entry by looking at the trace code in the first byte of the trace table entry. If bit 1 of the trace code contains a zero, the entry was made by the IPLed processor; but if bit 1 of the trace code contains a 1, the entry was made by the other processor. Processor identification information is implemented for an AP/MP system at system initialization when the system assigns each processor a trace identifier. The system assigns the IPLed processor a trace identifier of X'00' and the non-IPLed processor a trace identifier of X'40'. The identifier is ORed with the trace code when an entry is made in the trace table thus providing an easy way of determining which processor made a particular entry.

The following trace table entries appear in an AP/MP environment:

X'12'	indicates that the processor is spinning on a lock
X'13'	indicates that a processor issued a signal processor (SIGP) instruction
X'01'	may reflect multiprocessing-related external interruption codes (also appears in a uniprocessor environment)

When you are debugging an AP/MP system, you must relate the entries made by one processor to the entries made by the other processor in the same time period. For example, a signal processor (code X'13') entry by one processor should be followed closely by an external interruption (code X'01') for the other processor. See Figure 8 on page 73 for the layout of the CP internal trace table. The CP internal trace table pointers:

- Trace table start address
- Trace table end address
- Next available trace entry address

are in absolute page zero.

Lockwords

You can look in the DMKLOK module to find the status of the various VM/SP locks except the VMBLOK lock and the RDEVBLOK lock. Each of the locks in DMKLOK contains the following four fullwords of information:

- First fullword contains the logical processor address of the owning processor. This will be zero if the lock is not held.

- Second fullword contains the value in the lock owner's register 12.
- Third fullword contain the total amount of time spent spinning on this lock.
- Fourth fullword contains the total number of spins.

The VMBLOK lock is located in the VMBLOK at VMLOCK. When the VMBLOK lock is held, VMLOCK contains the logical processor address of the owning processor.

The RDEVBLOK lock is located in the RDEVBLOK at RDEVIOBL. When the lock is held, RDEVIOBL contains the logical processor address of the owning processor.

VMDUMP Records: Format and Content

When a user issues the VMDUMP command, CP dumps virtual storage of the user's virtual machine. CP stores this dump on the reader spool file of a virtual machine that the user specified as an operand on the VMDUMP command.

CP writes the storage dump to the spool file as a series of logical records. Each spool file record and each logical dump record is 4096-bytes long. However, because each spool file record contains a header, one logical dump record does not fit into one spool file record. For this reason, CP splits a logical dump record into two parts. CP writes one part to one spool file record and the other part to an adjacent spool file record. The size of each part varies depending upon the amount of space remaining in the spool file record that CP is currently using. Thus, each logical dump record spans two spool file records. Figure 11 on page 96 shows the format of spool file records, the format of logical dump records, and how logical dump records span spool file records.

The first spool file record contains a spool page buffer linkage block (SPLINK) followed by a TAG area followed by dump information. All other spool file records contain only a SPLINK followed by dump information.

A SPLINK, which contains data needed to locate information in the associated spool file record, has the following format:

Hexadecimal Offset	Length	Content
0	4 bytes	DASD location (DCHR) of the next page buffer
4	4 bytes	DASD location (DCHR) of the previous page buffer
8	4 bytes	Binary zeroes
C	4 bytes	Number of data records in the buffer

The TAG area contains either binary zeroes or user supplied data. If a virtual machine program or the user has issued the TAG command, the

TAG area contains the information provided via this command. Otherwise it contains binary zeroes.

The first logical dump record contains a dump file information record (DMPINREC). The second and third logical dump records each contain a dump file key storage record, DMPKYREC1 and DMPKYREC2 respectively. The dump file key storage records contain the value of the storage keys assigned to each page of virtual storage. The remaining logical dump records contain the virtual machine storage dump.

CP records the storage dump sequentially starting with the lowest address dumped and ending with the highest address dumped. CP records each byte as an untranslated 8-bit binary value.

For a description of the format and contents of DMPINREC, see *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP*. For a description of DMPKYREC1 and DMPKYREC2, see DMPKYREC also in *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP*.

The VMDUMP command dumps virtual storage that VM/SP created for the virtual machine user. VMDUMP creates a file that provides IPCS with header information to identify the owner of the dump. Once VMDUMP creates the file, IPCS may process it to debug errors, as well as to store and maintain error information about the virtual machine. For additional information, see Chapter 7, "Debugging Using IPCS" on page 197.

Locating Logical Dump Records

To locate a specific logical dump record, use the algorithm:

$$\text{loc} = \frac{240+16n+4096n}{4096}$$

where:

n

is a number that identifies the dump record. For example, to locate the first dump record, assign n a value of 1; to locate the second record, assign n a value of 2, and so forth.

loc

is the quotient and remainder of the algorithm.

Together these values specify a spool file record and an offset into that record where logical dump record *n* begins. The quotient specifies the spool file record, and the remainder specifies the offset into the spool file record.

The following example shows how to locate the third logical dump record:

$$\text{loc} = \frac{240+(16\text{x}3)+(4096\text{x}3)}{4096}$$

$$\text{loc} = \frac{12576}{4096}$$

```
quotient = 3

remainder = 288
```

Thus, the third dump record starts 288 bytes into the third spool file record.

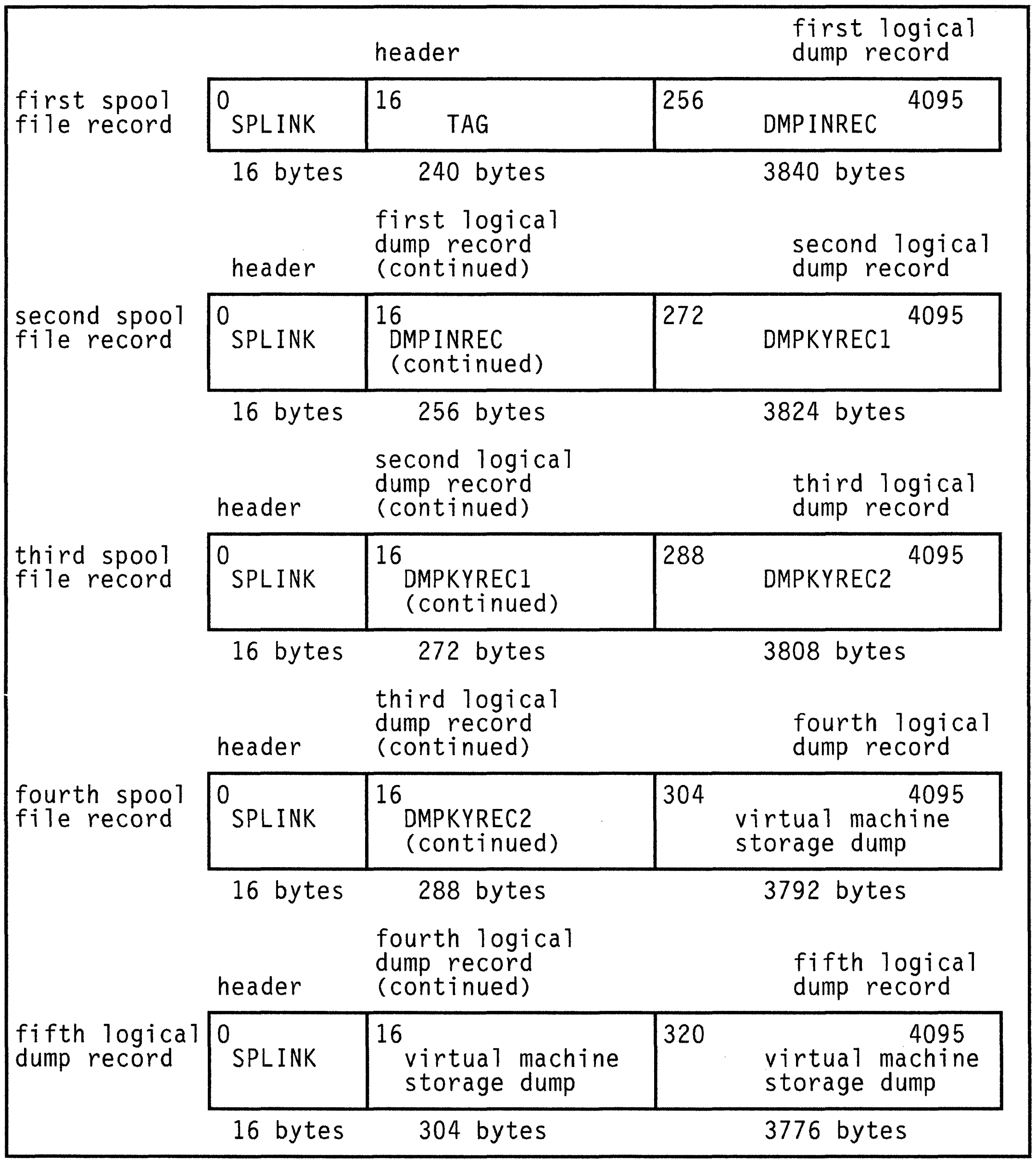


Figure 11. VMDUMP Record Format

Trapping Improper Use of CP Free Storage

Installations with UP and AP/MP generated systems can install the CP FRET Trap as an aid in solving problems caused by improper use of CP free storage and to solve many storage overlay problems. The design of the CP FRET Trap allows it to produce "tracks" in storage associated with each free storage request. The trap detects the release of areas of free storage that were not assigned, previously released, or outside the boundaries of the storage given.

The trap code is conditionally assembled in the DMKFRE, DMKFRT, and DMKCPI modules based on the value of the option &FRETRAP. &FRETRAP can be found in OPTIONS COPY and has a default value of 0 for normal operations without the trap.

The CP FRET Trap does the following:

- Disables CP Assist FREE, FRET, DSP1, DSP2, and UNTFR instructions.
- Expands each request for free storage by a three doubleword extension containing:
  - The status of the request. The status consists of the tag ALLO when the storage is allocated by DMKFRE or the tag FRET when the storage is released by DMKFRT.
  - The saved size (in doublewords) of the requested free storage area.
  - The address of the assigned free storage block.
  - The return address of the module requesting the storage.
  - The last three bytes of the calling module's name (if it is pageable).
  - The user's VMBLOK address.
  - The rest of the extension is cleared with zeroes and remains unused until the storage is released. The content at that time is as follows:
    - The return address of the module releasing the storage.
    - The last three bytes of the calling module's name (if it is pageable).

  Note: For the exact format of the extension, refer to the FREEXT control block in the *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP.*
- Checks each request to release free storage for the expected tag. Checks the size of the free storage area to be released against the saved size in the extension area, and abends in illegal situations. The ALLO

tag is replaced with the FRET tag if the trap detects no problems with the FRET request.

- For **VM/SP HPO** prime storage requests, it uses a doubleword trap extension containing the:
  - Last three bytes of the calling module's name (if it is pageable)
  - Low-order byte of the size requested
  - Return address of module requesting storage
  - Saved size (in doublewords and bytes).
- For **VM/SP HPO** prime storage FRET requests, it checks for the block marked prime. It checks the size of the area to be released against the size saved and checks whether it is on a cache boundary. The tag is replaced if no problems are detected.

When the CP FRET Trap is installed, performance for systems using CP Assists is degraded due to the disabling of the DSP1, DSP2, UNTFR, FREE, and FRET instructions. Also, performance for storage constrained systems having many users is degraded due to the expansion of each free storage request to include the trap extension area. The performance degradation is not likely to be a problem while suspected free storage problems are being trapped. The overall performance of the system remains the same when the trap is not installed.

The trap may be installed at system generation time. Refer to the *VM/SP Installation Guide* or *VM/SP HPO Installation Guide* for installation instructions and the *VM/SP System Logic and Problem Determination Guide Volume 1 (CP)* or *VM/SP HPO System Logic and Problem Determination Guide - CP* for specifics on the logic.

CP FRET Trap Examples

The following two examples demonstrate how the trap may be used to solve problems caused by improper use of CP free storage.

Example 1: Destruction of the free storage pointer.

Symptom:

Module X obtains a 36 doubleword block of storage from DMKFREE. The data in the storage block is being overlaid by data that has no resemblance to the data expected to be there.

The CP FRET Trap is installed and it abends with code FRT015.

Analyzing the Available Data:

The trap found the ALLO tag at FREER1 + the value of FREER0 in bytes. It abended with code FRT015 because the saved size of the original request (in the trap extension area) did not match the size of the block to be released in FREER0. Examination of FREER12, FREER14, FREER0, and FREER1 reveals that module Y called DMKFRET to release 40 doublewords of storage at the address contained in FREER1. Further examination of the trap extension area shows that module X made the original request for the free storage and that the requested size was 36 doublewords.

Conclusion:

If the free storage pointer in FREER1 and the size in FREER0 were correct, the size would have matched the saved size in the extension area. If the free storage pointer in FREER1 were correct and the size in FREER0 incorrect, then the trap would not have found the ALLO tag at FREER1 + FREER0. The trap would have abended with code FRT013 instead of FRT015. Therefore, the free storage pointer in FREER1 was incorrect when module Y tried to release the storage block.

The free storage block of module X could have been overlaid by module Y when its free storage pointer was destroyed. Or, when the trap was not installed, the storage block could have been released by module Y, recycled by DMKFREE and reissued to module Z, causing an overlay of the storage obtained by module X.

Example 2: Release of more storage than was given.

Symptom:

Module Y obtains a 9 doubleword block of storage from DMKFREE. The data in the storage block is being overlaid by data that has no resemblance to the data expected to be there.

The CP FRET Trap is installed and it abends with code FRT013.

Analyzing the Available Data:

The trap abended with code FRT013 because it could not find the ALLO tag at FREER1 + the value of FREER0 in bytes. Examination of FREER12, FREER14, FREER0, and FREER1 shows that module X called DMKFRET to release 15 doublewords of storage at the address contained in FREER1. Examining the storage at FREER1 reveals that an ALLO tag can be found at FREER1 + 9 doublewords, and that the saved size in the extension is 9 doublewords. The VMBLOK address in the extension matches that in FREER11. Further examination of the storage for the next ALLO tag shows that the storage block obtained by module Y overlaps the storage being released by module X by 3 doublewords.

Conclusion:

Module X tried to release more storage than was actually given. The free storage block of module Y could have been overlaid when the size for the storage block being released by module X was incorrect. Or, when the trap was not installed, the storage block could have been released by module X, recycled by DMKFREE and reissued to module Z, causing an overlay of the storage obtained by module Y.

Debugging with the CPTRAP Facility

The CPTRAP Facility provides field engineers and system programmers with problem determination capability. The facility is used to create a reader spool file of selected trace table entries, CP data, and virtual machine data in the order they happen. This data is collected in 4K blocks and placed in the CPTRAP spool file (CPTRAP FILE). Each 4K block, called a CPTRAP record, is time stamped. Each record has multiple entries.

The TRAPRED command can be used to access the CPTRAP reader file and the data collected in the file. TRAPRED output can be either a spooled print file or an interactive terminal display.

Activating CPTRAP

To activate CPTRAP, issue the CPTRAP START subcommand. The type of spool file produced depends on whether the WRAP option is specified with the CPTRAP START subcommand. See the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for more details on the CPTRAP command and its options. To obtain information about the CPTRAP function, such as current:

- Status of CPTRAP
- Selectivity for each type of CPTRAP record or for a specific CPTRAP record

you can use the QUERY CPTRAP command, as described in the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference.*

A *wrap* spool file reuses spool space. The user indicates the number of CPTRAP records (4K blocks of data) to be maintained in the spool file. After this number of 4K records has been collected, new CPTRAP records overlay the older records already in the file. This makes it possible to limit the total amount of spool space used by CPTRAP.

A *non-wrap* spool file does not reuse spool space. The spool space used by the file is limited only to the spool space available on the system. When the CPTRAP file has filled 15M of spool space, the file is closed automatically and a new file is opened. Since CPTRAP records are not overlaid in a non-wrap file, the available spool space is used very quickly, if the entries are not chosen selectively.

Recording CP Trace Table Entries in the CPTRAP File

The internal CP trace table is maintained in real storage. It is a 'wrap' table that continuously overlays previously stored information with new trace table entries. As a result, all the information needed to determine the cause of a problem may not be present in the trace table. CPTRAP allows selected CP trace table entries to be recorded in a spool file. Thus it is possible to save CP trace table entries that would be lost when the internal trace table wraps.

Collecting Entries in the CPTRAP file: To collect trace table entries, the system programmer must ensure that CPTRAP is started and the appropriate selectivity is specified. Once this is done, the selected CP trace table entries are moved from the internal trace table to the spool file without changing their format or length.

Specifying Selectivity: The CP trace table entries that are collected in the spool file are selected by trace type (typenum). For a list of the defined trace types see Figure 8 on page 73. CPTRAP allows further selection of input based on specific fields in the trace table entry. The three allowed fields are VMBLOK address (VMBLOK), real or virtual device address (DEVADDR), and various code fields (CODE). Please see the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for more details on the CPTRAP command and its option for selectivity.

Recording Virtual Machine Data in the CPTRAP File

The virtual machine interface lets a virtual machine send a data entry to CPTRAP to be added to the CPTRAP file. Any program running in VM/SP or VM/SP HPO (application program, CMS, GCS, TSAF etc.) can contain the interface to CPTRAP. The data gathered in the CPTRAP file by this interface could help determine the problem in your program.

Collecting Entries in the CPTRAP File To collect virtual machine entries in the CPTRAP file, the system programmer must ensure that:

- CPTRAP is activated.
- Selectivity is specified for the kind of virtual machine entry to be collected.
- The virtual machine is enabled to enter data into the CPTRAP file.
- The program running in the virtual machine contains the virtual machine interface to CPTRAP.

Once this is done, CPTRAP can construct the virtual machine entries and put them into the spool file.

Specifying Selectivity: Use the CPTRAP typenum subcommand to specify the type of entry to be collected in the CPTRAP file. The two types of entries for virtual machines are:

- X'3D' for group virtual machine data
- X'3E' for individual virtual machine data.

Enabling a Virtual Machine: A virtual machine must be enabled to enter data in a CPTRAP file. Any number of virtual machines can be enabled on the system. A virtual machine remains enabled until it is logged off, or until CPTRAP is terminated.

Two commands are available to enable virtual machines. CPTRAP ALLOWID enables only the virtual machine specified by the command. Reissue the command to enable other virtual machines. This machine must be logged on or disconnected when the command is issued. CPTRAP GROUPID enables all virtual machines that are currently in the group specified by the command. Later, any new members entering the group are automatically enabled. For more details on these CPTRAP subcommands, see the: *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference.*

Setting up the Interface: The virtual machine interface to CPTRAP is a parameter list and a class 10 monitor call instruction. There is no restriction on the number of interfaces that may be active at one time, or on the number of virtual machines that can use them.

You can insert the interface into a program in two ways:

1. Use the CP STORE command to store the interface into a program problem area.

2. Modify your program to include the interface, then reassemble the program.

Set up register 1 with the address of the parameter list. The parameter list identifies the data to be included in the CPTRAP file. It contains the length of the data, an individualizing code, and the address of the data to be added to the CPTRAP file.

The format of the 8-byte parameter list for monitor codes 0 and 1 is as follows:

Disp	Field	Length	Description
0	Length	2	Length of virtual machine data
2	Code	2	Individualizing code
4	Address	4	Virtual address of data in user's storage

The data must be 2048 bytes or less, and must reside in the virtual machine. If the length is greater than 2048 bytes, only the first 2048 are taken with no indication that the data has been truncated.

The format of the 12-byte parameter list for monitor code 2 is as follows:

Disp	Field	Length	Description
0	Dlength	2	Length of virtual machine data. The data length is variable up to a maximum of 2048 bytes.
2	Code	2	Individualizing code
4	Address	4	Virtual address of the data to be included in the CPTRAP file. The data must be within the virtual machine that issues the monitor call instruction.
8	Plength	2	Length of parameter list (in bytes). This must be at least X'C'.
A	Machtype	1	Code that identifies the various 3E type CPTRAP records. The values are: • X'01' or TSAF - TSAF virtual machine entries • X'FE' or FE - Field engineering entries • X'FF' or USER1 - User installation entries
B		1	Reserved.

The individualizing code is used to look selectively at the virtual machine entry in the CPTRAP file when you use TRAPRED. This individualizing code will be present in each entry. If the individualizing code is unique for each interface that is set up, it will be easy to review data selectively in the CPTRAP file that came from a particular virtual machine interface.

The interface also uses a class 10 monitor call instruction. A monitor call instruction can be executed in virtual supervisor or virtual problem state, BC mode or EC mode, and in multi-level environments. Multi-level is defined as VM/SP or VM/SP HPO running a Guest Virtual Machine (GVM) of a VM/SP or VM/SP HPO system.

The supported monitor codes are 0, 1 and 2. All other monitor codes are ignored and control returns to the invoker with no indication that the virtual machine data was ignored.

The format of the monitor call instruction is as follows:

MC x,10

x = 0	indicates that the data to be added to the CPTRAP file is general virtual machine data. Any virtual machine can use a monitor code 0.
x = 1	indicates that the data to be added to the CPTRAP file is virtual machine group data. Only a virtual machine that belongs to a group can use a monitor code 1.

x = 2 identifies an extended format for the parameter list that the virtual machine passes (general virtual machine data). Any virtual machine can use a monitor code 2.

The following chart shows the type of entry (if any) made in the CPTRAP file for the six possible situations that can arise:

	Virtual Machine is in a Group	**Virtual Machine is not in a Group**
Monitor Code 0 is issued	3E	3E
Monitor Code 1 is issued	3D	no entry is made
Monitor Code 2 is issued	3E	3E
Any other monitor code is issued	no entry is made	no entry is made

Virtual Machine Entries in the CPTRAP File

CPTRAP constructs the virtual machine entry that is placed in the CPTRAP file. An 8-byte header is appended to the front of the data that is passed to the virtual machine. The header, which identifies the virtual machine entry in the file, has the following format:

0	typenum	/////	code	length	machtype	//////

Disp	Field	Length	Description
0	Typenum	1	3E for individual virtual machine entries 3D for group virtual machine entries
1		1	Reserved byte
2	Code	2	Individualizing code
4	Length	2	Length of virtual machine entry
6	Machtype	1	Code that identifies the various 3E type CPTRAP records. The values are: ● X'00' - For entries not created via Monitor code 2 ● X'01' or TSAF - TSAF virtual machine entries ● X'FE' or FE - Field engineering entries ● X'FF' or USER1 - User installation entries For 3D entries, this field is reserved.
7		1	Reserved
8	DATA	0 to 800	Can be up to 2048 bytes of virtual machine data

The individualizing code that CP puts in the third and fourth bytes of the header is the same code the user specified in the parameter list. The individualizing code is necessary to look selectively at the virtual machine entry in the CPTRAP file when you use TRAPRED. The length of the virtual machine entry in the CPTRAP file is variable. It includes the length of the virtual machine data plus eight for the length of the header.

Recording CP Data in the CPTRAP File

A CP interface to CPTRAP lets CP send information to be recorded in the CPTRAP file. This data collected is used to solve a problem in CP code.

Collecting Entries in the CPTRAP file: To collect CP entries in the CPTRAP file, the system programmer must ensure that:

- CPTRAP is activated.
- Selectivity is specified for CP entries.
- The interface is contained in the CP code.

Once this is done, CPTRAP can construct the CP entries and put them into the spool file.

Specifying Selectivity: Use the CPTRAP typenum subcommand to specify that CP entries are to be collected in the CPTRAP file. CP entries have a typenum of X'3F'.

Setting up the CP Interface: The CP interface to CPTRAP is a parameter list and a BALR 14,15 instruction. There are no restrictions on the number of interfaces that may be active at the same time. You can insert the CP interface into the CP code in two ways:

1. Use the CP STCP command to store the interface into the problem area in the CP code.

2. Modify the CP module to include the interface, reassemble the particular source module, and regenerate the system.

Be careful where you insert the interface in the CP code. There may be a condition code setting that has not yet been interrogated by the CP code. Any code inserted as part of the interface must not change that condition code setting. If the inserted code changes the condition code, it also must save and restore this setting. CPTRAP preserves the condition code setting in effect at the time the BALR 14,15 instruction executes.

Set up register 1 with the address of the parameter list. The parameter list identifies the data to be included in the CPTRAP file. It contains the length of the data, an individualizing code, and the address of the data to be added to the CPTRAP file.

The format of the 8-byte parameter list is as follows:

Disp	Field	Length	Description
0	Length	2	Length of CP data
2	Code	2	Individualizing code
4	Address	4	Address of CP data

The data must be 280 bytes or less, and must reside in real storage. If the length is greater than 280 bytes, only the first 280 are taken with no indication that the data length has been truncated.

The individualizing code is used to look selectively at the CP entry in the CPTRAP file when you use TRAPRED. This individualizing code will be present in each entry. If the individualizing code is unique for each interface that is set up, it will be easy to review data selectively in the CPTRAP file that came from a particular CP interface.

The interface also uses a BALR 14,15 instruction. Register 15 must be set up with the address of TRAPOK in the PSA before the BALR instruction is issued. This is the address of the logic within module DMKPSA which determines if CPTRAP is active. When the BALR 14,15 instruction is issued, register 14 gets the return address to the caller. The status of the CPTRAP facility determines what happens next. If CPTRAP is active, control goes to CPTRAP. If CPTRAP is not active, control returns to the caller immediately.

CP Entries in the CPTRAP File

CPTRAP constructs the CP entry that is placed in the CPTRAP file. An 8-byte header is appended to the front of the data that is passed by CP. The header identifies the CP entry in the file with the following format:

0	typenum	/////	code	length	/////////////

<u>Disp</u>	<u>Field</u>	<u>Length</u>	<u>Description</u>
0	Typenum	1	3F for CP data
1		1	Reserved byte
2	Code	2	Individualizing code
4	Length	2	Length of CP entry
6		2	Reserved

The individualizing code that CP puts in the third and fourth bytes of the header is the same code the user specified in the parameter list. The individualizing code is necessary to look selectively at the CP entry in the CPTRAP file when you use TRAPRED. The length of the CP entry in the CPTRAP file is variable. It includes the length of the CP data plus eight for the length of the header.

Additional CPTRAP Considerations

Checkpointing

Closed CPTRAP reader files are checkpointed just like any other spool files. In addition, if the system abends and the CPTRAP file is still open, the file will be closed and checkpointed.

AP and MP Support

APs and MPs are supported by the CPTRAP facility. When the facility is actively processing the CP trace table, the AP/MP lock becomes active. The control method is a word within the module which is tested with a TS (TEST and SET) instruction. This word also holds the CPU identification of the latest CPU through the CPTRAP logic.

Running with Microcode Assist Active

The CPTRAP facility deactivates the dispatcher assists in ECPS:VM/370 to support the monitor call interface for virtual machines. Deactivation occurs only when CPTRAP is active.

LOGOFF Considerations

The CPTRAP facility is stopped if the user who invoked CPTRAP logs off.

Spool Space Considerations

The CPTRAP facility is stopped if there is no spool space available on the system. When the system is using 90% of its spool space, and again when it is using 100%, CP sends a message to the user. When no space is available CPTRAP closes the file, creates a READER file, and stops processing.

Lost Data

A data lost message is issued when the system creates output faster than it can be transferred to the spool. When this happens, the output file also indicates that data has been lost. The amount of data lost can be 4K of CP internal trace table and/or CP/virtual data records. The possibility of a data lost situation is:

- Directly proportional to the rate of transfer of trace table data to spool.
- Directly proportional to the frequency and size of interface data.
- Inversely proportional to speed of the spool DASD. This is a potential problem with the faster CPUs and/or with heavy use of the interface.

A reduced selection of trace types and CP or virtual machine data helps reduce lost data problems.

CP/Virtual Machine Interface Errors

Two specific problems can occur in the following situations:

- Any byte of the parameter list or any byte of the data field lies outside of the virtual machine or CP storage due to an invalid address.
- An I/O error occurs while trying to read a page.

When these errors occur:

1. The system sends an informational message to the virtual machine user who started CPTRAP.
2. CP puts a special indicator, ADDR BAD, into the file.
3. The system ignores the data sent by VM or CP.

Release Level Conflicts

Problems may arise if you are using different VM/SP or VM/SP HPO release levels of CPTRAP and TRAPRED.

Using TRAPRED at VM/SP Release 3 level to process a CPTRAP file from VM/SP Release 4 or 5 can provide unpredictable results. This level of TRAPRED cannot handle X'3D' entries, but can handle CP trace table entries.

Similarly, using TRAPRED at VM/SP HPO Release 3.2, 3.4, or 3.6 levels to process a CPTRAP file from VM/SP HPO Release 4.0, 4.2, or 5.0 can provide unpredictable results. The VM/SP HPO release 3.2, 3.4, and 3.6 levels cannot handle X'3D' entries, but can handle CP trace table entries.

Using TRAPRED at VM/SP Release 4 or 5 or VM/SP HPO Release 4.0, 4.2, or 5.0 level to process a CPTRAP file from VM/SP Release 3 or VM/SP HPO Release 3.2, 3.4, or 3.6 works correctly for hexadecimal output. Formatted output gives unpredictable results if the contents of VM/SP Release 3 or VM/SP HPO Release 3.2, 3.4, or 3.6 trace table entries differ from the contents of VM/SP Release 4 or 5 or VM/SP HPO Release 4.0, 4.2, or 5.0.

Using the TRAPRED Facility

Viewing Entries in the CPTRAP File

The spool file created by the CPTRAP facility has a filename of CPTRAP and a filetype of FILE. It is made up of noncontiguous 4K spool records which contain many individual entries. The records contain only data. No CCWs are within them. The file cannot be accessed as a normal spool file. The TRAPRED command must be used to access the file and review the entries.

The TRAPRED program is a CP module shipped with the CMS modules. It resides on the CMS S-disk with an S2 filemode. TRAPRED writes CP error messages to the terminal, and writes line mode data to the terminal and to the printer. Although TRAPRED issues CP messages, the program runs in the CMS user area. You must be in CMS to use TRAPRED.

Using the TRAPRED Command

Use the TRAPRED command to access the CPTRAP reader file and review the entries contained in that file. Once TRAPRED has been invoked, you may execute TRAPRED subcommands. You can also execute the CMS immediate commands HT and HX, and CP commands (if prefaced with 'CP'). Return control to the CMS environment by issuing the TRAPRED subcommand QUIT.

The format of the TRAPRED command is:

TRAPRED	*filenum*

where:

filenum
: is the *number* of the reader file that is the output of CPTRAP processing. The filename and filetype are 'CPTRAP FILE'.

  The spool file specified must belong to the user invoking the program. It also must be of a class which can be read and must not be held.

After you enter 'TRAPRED filenum', the CPTRAP reader file is accessed and interactive processing may begin. When the TRAPRED command is invoked, the message 'ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND' is issued.

TRAPRED Subcommands

The TRAPRED subcommands, as discussed below, have been grouped by the following tasks:

- Identify type of entry.
- Determine type of display.
- Manipulate files.

Note: The TRAPRED subcommands must be entered on separate command lines.

The TRAPRED subcommands that help you identify the type of entry to be selected are:

- Typenum
- ALL.

The typenum subcommand identifies the type of entry to be selected. Selectivity defaults to ALL ON when the TRAPRED command is invoked. If ALL ON is in effect when typenum is issued, all other typenums are reset to OFF. Possible input for typenum is:

01 - 21	for CP trace table entries
3D	for group virtual machine data
3E	for general virtual machine data
3F	for CP data

Typenum cannot be specified on the same command line with other TRAPRED subcommands. However, multiple typenum subcommands can be issued on the same command line. The format of the typenum subcommand is:

typenum	$\left[\begin{array}{ll}\textbf{Vmblok} & \textit{address} \\ \textbf{Devaddr} & \textit{cuu} \\ \textbf{COde} & \textit{code-value} \\ \textbf{MACHtype} & \left\{\begin{array}{l}\textit{nn} \\ \textit{machname}\end{array}\right\} \\ \textbf{OFF} & \end{array}\right]$

where:

Vmblok *address*
specifies additional selectivity based on a VMBLOK address. This option is only valid for entries in the file that have a VMBLOK field. See the discussion of CPTRAP in the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for a chart of the typenums that can specify the Vmblok option.

DEVaddr *cuu*
specifies additional selectivity based on a device address. This option is only valid for entries in the file that have a real or virtual device address field. See the discussion of CPTRAP in the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for a chart of the typenums that can specify the DEVaddr option.

COde *code-value*
specifies additional selectivity based on a code. This option is only valid for entries in the file that contain a code field. See the discussion of CPTRAP in the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for a chart of the typenums that can specify the COde option. This option is also valid for 3D, 3E, and 3F entries. For these typenums, the selectivity is based on the individualizing code in the entry.

MACHtype $\left\{\begin{array}{l}\textit{nn} \\ \textit{machname}\end{array}\right\}$

lets you restrict entries of type '3E' to a specific machine type, which you can enter as a hexadecimal code or as a mnemonic string.

If a formatting routine is listed in the internal TRAPRED data tables for the machine type being processed by TRAPRED, CP will invoke it to produce the output. This information is contained in OTABDATA COPY (O4TABLE). If the formatting routine does not exist or is not listed in the table, CP will display the information in the general hexadecimal and EBCDIC format for 3E records.

MACHtype
indicates the data is the machine-type information to further define the selectivity of the 3E record.

nn
is the machine-type found in the CP header of the trace entry. This is a one-to-two digit hexadecimal number.

machname
identifies the machine type. The values are:

- X'01' or TSAF - TSAF virtual machine entries
- X'FE' or FE - Field engineering entries
- X'FF' or User1 - User installation entries.

For example, if you enter:

3E MACHTYPE TSAF
TYPE 3

you may get output like the following:

```
TSAF       9AF3D043 1B225000 D4E3E8 60E8 0005    TRACE INPUT
TSAF       9AF3D043 1B234000 D4E3E8 60E6 0008    ATSMTY EXIT
TSAF       9AF3D043 1B298000 D3F1D7 A461 0004    MODULE ENTERED AT ATSL1P
```

If you want to see the formatted version, you could enter the following:

FORMAT
TYPE 1

In this case, you would get something like the following:

```
60E8 TRACE INPUT                                                   TSAF
       CLOCK = 9AF3D043 1B225000               MODULE = ATSMTY
       TIME = 08:43:08.627493 GMT 05/20/1986   LENGTH = 0005
       DELAY              0000000C                             *....*
```

OFF
deletes all selectivity set up for the specified typenum.

The ALL subcommand is used to turn the selection of all typenums on or off. The format of the ALL subcommand is:

ALL	[<u>**ON**</u> / **OFF**]

where:

<u>**ON**</u>
turns the selection of all typenums on. ON is the default setting.

OFF
turns the selection of all typenums off.

The TRAPRED subcommands that are used the determine the type of display that is presented are:

- HEX
- FORMAT.

The HEX subcommand indicates that hexadecimal output is requested for the entries that will be displayed. Hex is the default and is in effect until FORMAT is issued. When TSAF entries are displayed in the hex mode only "TSAF", trailer data in hex, and a descriptor are displayed. The format of the HEX subcommand is:

HEX	

The FORMAT subcommand indicates that formatted output is requested for the entries that will be displayed. FOrmat is in effect until HEX is issued.

Notes:

1. *3F entries are user-defined data and are not formatted.*

2. *A formatter may be provided by the installation for 3E entries (MC 2,10) in the same manner as TSAF provides one.*

The format of the FORMAT subcommand is:

FOrmat	

The TRAPRED subcommands that help you to manipulate the files are:

- TOP
- BOTTOM
- UP
- DOWN
- TYPE
- TYPEBACK
- PRINTER
- QUIT.

The TOP subcommand positions you at the top of the CPTRAP reader file. The null entry at the top of the file becomes the new current entry. It contains

```
* * * TOP OF FILE * * *
```

The format of the TOP subcommand is:

TOP	

The BOTTOM subcommand positions you at the bottom of the CPTRAP reader file. The last entry of the file becomes the current entry. The last entry displayed is

* * * END OF FILE * * *

The format of the BOTTOM subcommand is:

BOTtom	

The UP subcommand scrolls a specified number of entries toward the top of the file and displays the new current entry. Wrapping is not allowed, and the subcommand ends prematurely if the top of the file is reached. The format of the UP subcommand is:

Up	$\begin{bmatrix} \underline{\mathbf{1}} \\ n \end{bmatrix}$

where:

n
: indicates the number of entries to be skipped. It may be any positive decimal integer up to eight digits. One is the default value.

The DOWN subcommand scrolls a specified number of entries toward the end of the file and displays the new current entry. Wrapping is not allowed, and the subcommand ends prematurely if the end of the file is reached. The format of the DOWN subcommand is:

Down	$\begin{bmatrix} \underline{\mathbf{1}} \\ n \end{bmatrix}$

where:

n

indicates the number of entries to be skipped. It may be any positive decimal integer up to eight digits. One is the default value.

The TYPE subcommand displays the specified number of selected entries at the terminal starting with the current entry and moving toward the end of the file. The subcommand ends prematurely if the end of the file is reached. The format of the TYPE subcommand is:

Type	$\begin{bmatrix}\underline{1}\\ n\end{bmatrix}$

where:

n

is the number of selected entries to be displayed at the terminal. The number may be any positive decimal integer up to eight digits. One is the default value.

The TYPEBACK subcommand displays the specified number of selected entries at the terminal starting with the current entry and moving toward the top of the file. The subcommand ends prematurely if the top of the file is reached. The format of the TYPEBACK subcommand is:

TYPEBack	$\begin{bmatrix}\underline{1}\\ n\end{bmatrix}$

where:

n

is the number of selected entries to be displayed at the terminal. The number may be any positive decimal integer up to eight digits. One is the default value.

The PRINTER subcommand spools selected entries to the printer starting with the current entry and moving toward the end of the file. The subcommand ends prematurely if the end of the file is reached. The format of the PRINTER subcommand is:

Printer	$\begin{bmatrix}\underline{1}\\ n\end{bmatrix}$

where:

n

is the number of selected entries to be spooled to the printer. The number of entries may be any positive decimal integer up to eight digits. One is the default value.

The QUIT subcommand ends TRAPRED and returns you to the CMS environment. The accessed reader file is not purged. To purge the file, you must explicitly issue the CP PURGE command. The format of the QUIT subcommand is:

QUIT	

CPTRAP Examples

How to Collect CP Data in CPTRAP File

Generating the CP Data

Logic in CP Code: A CP interface to CPTRAP lets CP send information to be recorded in the CPTRAP file. This lets the system programmer collect problem determination data for solving problems that may be related to CP code. The CP interface to CPTRAP could be used to trace CP control blocks at various points in the CP code. For example, code in the module DMKQCN releases the storage used for CONTASKs. If you needed to record the information in a CONTASK for problem determination before it is released, you could include the following code at the appropriate location in DMKQCN:

```
         .
         .
         .
ALLDONE  DS     0H               HERE TO RELEASE THE CONTASK
         USING  CONTASK,R6       GET ADDRESSABILITY TO CONTASK
         LH     R2,CONTSKSZ      GET CONTASK LENGTH IN DWORDS
         SLL    R2,3             CONVERT THE LENGTH INTO BYTES
         STH    R2,DATALEN       PUT LENGTH OF CONTASK IN PLIST
         ST     R6,DATADDR       PUT ADDRESS OF CONTASK IN PLIST
         USING  PSA,R0           NEED ADDRESSABILITY TO PSA
         BAL    R1,AROUND        SET UP POINTER TO PLIST
*                                PARAMETER LIST:
DATALEN  DS     AL2                2 BYTES FOR LENGTH
         DC     AL2(3)             2 BYTES FOR CODE .. (THIS IS 3)
DATADDR  DS     AL4                4 BYTES FOR ADDRESS
AROUND   DS     0H
         LA     R15,TRAPOK       GET ADDRESS OF TRAPOK
         BALR   R14,R15          SEND DATA TO CPTRAP FILE
         .
         <existing code in DMKQCN to release the CONTASK>
         .
         .
```

Now, to use this trap in DMKQCN, the module has to be reassembled and the system programmer has to rebuild the system using the modified version of DMKQCN. Then, every time a CONTASK is released, a parameter list is set up and control goes to CPTRAP.

The system programmer can set-up any number of traps in CP code at the same time. By making the individualizing code unique in each case, the CP entries can be reviewed selectively in the CPTRAP file. In the example, the CP entries that are created have an individualizing code of 3.

Collecting the CP Data in the CPTRAP File

When CPTRAP is not active, control immediately returns to the caller, DMKQCN.

When CPTRAP is active, control is given to the CPTRAP module (DMKTRT). If X'3F' entries are being collected, the data identified by the parameter list is recorded in the CPTRAP file.

To activate CPTRAP and collect only records that CP sends, issue the following set of commands:

CPTRAP START
CPTRAP 3F

which will give the following message:

```
CPTRAP SELECTIVITY RESET
Ready;
```

Now, whenever any CONTASK in the system is released, this is recorded in the CPTRAP file.

If a user issued the following 2 messages:

M OP ARE YOU LOGGED ON TODAY
M OP DID I CATCH THIS CONTASK IN THE CPTRAP FILE?

this would have created 2 CONTASKS and when these are released, you would expect to find both of them in the CPTRAP file.

To stop the CPTRAP facility enter:

CPTRAP STOP

which creates a reader file such as this:

```
RDR FILE 0002  TO  WILL      COPY 001   NOHOLD
CPTRAP COMMAND COMPLETE
```

Displaying the CPTRAP Output

Look at the CPTRAP file using the TRAPRED facility. By spooling your reader to the same class as the CPTRAP file you can access the reader file that was created. If you enter:

Q RDR ALL

you could get the following response:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME      TYPE     DIST
WILL     0002 P DMP 00000001 001 NONE 02/10 14:23:06 CPTRAP    FILE     WILL
Ready;
```

If you enter:

TRAPRED 2

you would get the following response:

```
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

Since the parameter list set up in DMKQCN used "3" for an individualizing code, you can use the TRAPRED subcommand, 3F CODE 3, to indicate that only CP data entries that came from the trap in DMKQCN should be selected. If you entered:

3F CODE 3

you would get the following response:

```
TRAPRED SELECTIVITY RESET
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

Using the TRAPRED subcommand TOP, you can position yourselves at the top of the file. If you entered:

TOP

you would get the following response:

```
* * * TOP OF FILE * * *
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

Using the TRAPRED subcommand TYPE, you can display the entries you want to see at the terminal. If you entered:

TYPE 2

you could get the following response:

```
14:23:06
           3F000003 00800000 00000000 00000000         *................*
           8003000F 000411C0 00440000 00000000         *................*
           001E4128 00000037 00000000 00000000         *................*
           00000000 00000000 00000000 00000000         *................*
           F1F47AF2 F37AF2F2 151515D4 E2C740C6         *14:23:22...MSG F*
           D9D6D440 E6C9D3D3 40404040 7A40C1D9         *ROM WILL    : AR*
           C540E8D6 E440D3D6 C7C7C5C4 40D6D540         *E YOU LOGGED ON *
           E3D6C4C1 E8151500 00000000 00000000         *TODAY...........*
           3F000003 00980000 00000000 00000000         *................*
           80030012 000411C0 00440000 00000000         *................*
           001E3EB8 0000004B 00000000 00000000         *................*
           00000000 00000000 00000000 00000000         *................*
           F1F47AF2 F37AF3F5 151515D4 E2C740C6         *14:23:35...MSG F*
           D9D6D440 E6C9D3D3 40404040 7A40C4C9         *ROM WILL    : DI*
           C440C940 C3C1E3C3 C840E3C8 C9E240C3         *D I CATCH THIS C*
           D6D5E3C1 E2D240C9 D540E3C8 C540C3D7         *ONTASK IN THE CP*
           E3D9C1D7 40C6C9D3 C5151500 00000000         *TRAP FILE.......*
           00000000 00000000 00000000 00000000         *................*
```

Since you are positioned at the top of a CPTRAP record, the time stamp for that record is displayed. Next, notice that the two entries in the CPTRAP file are from CP (3F) and came from your trap in DMKQCN, that is, they have an individualizing code of 3. These entries are displayed in hex on the left with the EBCDIC translation on the right. The first 3F entry displayed has an individualizing code of 3. The entire entry is X'80' (128) bytes long. The first 8 bytes are the CP header. The 120 bytes following the CP header contain the CONTASK for the first message issued. In the next 3F entry displayed, notice that the 144 bytes following the CP header contain the CONTASK for the second message issued.

How to Collect Virtual Machine Data in a CPTRAP File

Generating the Virtual Machine Data

Logic in a Program: A virtual machine interface to CPTRAP lets a virtual machine send information to be recorded in the CPTRAP file. This lets the system programmer collect information for solving problems that may be related to a program running in the virtual machine.

Two types of virtual machine data can be recorded in the CPTRAP file.

- The data is general virtual machine data if it is sent by any virtual machine that is enabled and uses a monitor code 0 when passing the data to CPTRAP.
- The data is group virtual machine data if it is sent by a virtual machine that is enabled, uses a monitor code 1 when passing the data to CPTRAP, and belongs to a group.

One way that the virtual machine interface to CPTRAP might be used is to capture some data being changed incorrectly by the program. For example, you could include the following code at the appropriate location in the program:

```
         .
         .
         .
DOTHIS   CNOP  0,4            HERE TO RECORD ENTRY IN CPTRAP FILE
         STH   R2,DATALEN     PUT LENGTH OF DATA YOU WANT IN PLIST
         ST    R6,DATADDR     PUT ADDRESS OF DATA YOU WANT IN PLIST
         BAL   R1,AROUND      SET UP POINTER TO PLIST
*                             PARAMETER LIST:
DATALEN  DS    AL2              2 BYTES FOR LENGTH
         DC    AL2(5)           2 BYTES FOR CODE .. (THIS IS 5)
DATADDR  DS    AL4              4 BYTES FOR ADDRESS
AROUND   DS    0H
         MC    0,10           SEND GENERAL VM (3E) DATA TO CPTRAP
         .
         <existing code in program to continue doing something>
         .
         .
```

Now, to use this trap in the program, this program has to be reassembled. The system programmer has to do whatever is required to run this new version of the program. Then, every time something causes the code in the trap to execute, the parameter list would be set up and control would go to CPTRAP.

The system programmer can set up any number of these traps in the code at the same time. By making the individualizing code unique in each case, the virtual machine entries can be reviewed selectively in the CPTRAP file. In the example here, the virtual machine entry created is for general virtual machines (type 3E) and has an individualizing code of 5.

Collecting the Virtual Machine Data in the CPTRAP File

When CPTRAP is activated, the monitor code interface gives control to CPTRAP. If X'3E' entries are being collected then the data identified by the parameter list is recorded in the CPTRAP file.

To activate CPTRAP and collect only general virtual machine data, issue the following set of commands:

CPTRAP START
CPTRAP 3E

and you will get a response like this:

```
CPTRAP SELECTIVITY RESET
Ready;
```

Now, whenever the code in your trap executes, an entry should be made in the CPTRAP file. You can stop the CPTRAP facility and cause the reader file to be created by issuing:

CPTRAP STOP

and you will get a response like this:

```
RDR FILE 0003  TO  WILL     COPY 001   NOHOLD
CPTRAP COMMAND COMPLETE
Ready;
```

Displaying the CPTRAP Output

Look at the CPTRAP file using the TRAPRED facility. By spooling your reader to the same class as the CPTRAP file you can access the reader file that was created. If you enter:

Q RDR ALL

you will get a response like this:

```
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME     NAME     TYPE     DIST
WILL     0003 P DMP 00000001 001 NONE 02/10 15:30:06 CPTRAP   FILE     WILL
Ready;
```

Then if you enter

TRAPRED 3

you would get the following response:

```
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

Since the parameter list set up in the program used "5" for an individualizing code, you can use the TRAPRED subcommand, 3E CODE 5, to indicate that only virtual machine data entries that came from the trap in your program should be selected. If you enter:

3E CODE 5

you would get the following response:

```
TRAPRED SELECTIVITY RESET
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

Using the TRAPRED subcommand TOP, you can position yourself at the top of the file. If you enter:

TOP

you would get the following response:

```
* * * TOP OF FILE * * *
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

Using the TRAPRED subcommand TYPE, you can display the entries you want to see at the terminal. If you enter:

TYPE

you could get a response that looked something like this:

```
15:01:35
          3E000005 00B00000 E3C8C9E2 40C4C1E3           *........THIS DAT*
          C140C9E2 40C6D9D6 D440E3C8 C540E3D9           *A IS FROM THE TR*
          C1D740C3 D6C4C5C4 40C9D540 E3C8C540           *AP CODED IN THE *
          C1D7D7D3 C9C3C1E3 C9D6D540 D7D9D6C7           *APPLICATION PROG*
          D9C1D44B 40E3C8C5 40E2E8E2 E3C5D440           *RAM. THE SYSTEM *
          D7D9D6C7 D9C1D4D4 C5D940E2 C8D6E4D3           *PROGRAMMER SHOUL*
          C440E2C5 E340E3C8 C9E240E4 D740E3D6           *D SET THIS UP TO*
          40C3D6D5 E3C1C9D5 40E3C8C5 40D7D9D6           * CONTAIN THE PRO*
          C2D3C5D4 40C4C5E3 C5D9D4C9 D5C1E3C9           *BLEM DETERMINATI*
          D6D540C9 D5C6D6D9 D4C1E3C9 D6D540E3           *ON INFORMATION T*
          C8C1E340 C9E240D9 C5D8E4C9 D9C5C44B           *HAT IS REQUIRED.*
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

Since you are positioned at the top of a CPTRAP record, the time stamp for that record is displayed. Next, notice the entry (3E) in the CPTRAP file that would have come from the trap set in the program running in the virtual machine (a CODE of 5). The entry is displayed in hex on the left with the EBCDIC translation on the right.

Displaying Formatted CPTRAP Output

All CPTRAP entries can be displayed in hexadecimal. It is possible to obtain formatted output for CP trace table entries and X'3D' entries in the CPTRAP file. The X'3E' and X'3F' entries, which contain variable or user-defined data, cannot be formatted.

Following is an example of both hexadecimal output and formatted output for some CP trace table entries in a CPTRAP file. Use the TRAPRED subcommands TOP, HEX, and TYPE 5 to display the first five entries in the CPTRAP file in hexadecimal. If you enter:

TOP

you would get a response like this:

```
* * * TOP OF FILE * * *
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

If you enter:

HEX

you would get a response like this:

```
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

If you enter:

TYPE 5

you would get a response that could look like this:

```
10:37:18
          0A000000 00043820 070D0000 00195D04    RUN USER
          03000400 00040010 070D3000 00194264    PROGRAM INTERRUPT
          020216E0 00020008 000C0000 0001FB22    SVC INTERRUPT (CALL)
          0201FB22 0002000C 000C0000 00021AAE    SVC INTERRUPT (RETURN)
          02000001 000200CA 070D2000 00196000    SVC INTERRUPT (USER)
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

Now, to obtain formatted output of the same information, use the TRAPRED subcommands TOP, FORMAT, and TYPE 5 as follows. If you enter:

TOP

you would get a response like this:

```
* * * TOP OF FILE * * *
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

If you enter:

FORMAT

you would get a response like this:

```
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

If you enter:

TYPE 5

you could get a response like this:

```
10:37:18
0A RUN USER          **MP** 0A000000 00043820 070D0000 00195D04
       VMBLOK = 043820
       RUN PSW = 070D0000 00195D04
03 PROGRAM INTERRUPT   **MP** 03000400 00040010 070D3000 00194264
       ILC = 04, CODE = 0010 = SEG TRANS
       OLD PSW = 070D3000 00194264
02 SVC INTERRUPT       **MP** 020216E0 00020008 000C0000 0001FB22
       CALL TO   0216E0              FROM 01FB22
02 SVC INTERRUPT       **MP** 0201FB22 0002000C 000C0000 00021AAE
       RETURN TO 01FB22              FROM 021AAE
02 SVC INTERRUPT       **MP** 02000001 000200CA 070D2000 00196000
       USER SVC (DEC) =   202, OLD PSW = 070D2000 00196000
ENTER SELECTIVITY OPTION(S) OR SUBCOMMAND:
```

For examples of the hexadecimal and formatted CP trace table entries and a full explanation of the individual entries, see Chapter 7, "Debugging Using IPCS" on page 197. Please note that the formatted output produced by CPTRAP does not contain the following fields found in the formatted output produced by IPCS:

- Trace entry address
- Userid corresponding to a VMBLOK address
- Module names and displacement within the module.

For examples of the hexadecimal and formatted version of the X'3D' entries, see Chapter 5, "Debugging GCS" on page 153.

370X Dump Processing

The following sections discuss the Network Dump and the NCPDUMP. Use the NETWORK DUMP command to dump the 370x communications controller's storage. Use the NCPDUMP command to process CP spool reader files created by NETWORK DUMP command.

Network Dump Operations

This section only applies to 3704 or 3705 communication controllers that have been loaded by VM/SP or VM/SP HPO. If you want to dump the contents of a 3725 or a 3705 that has been loaded by ACF/SSP, refer to *ACF/NCP V4, ACF/SSP V3 Diagnosis Guide.*

If 3704/3705 operations are erratic, fatal hardware errors occur, or some other internal error appears, the Communications Controller's storage should be dumped. The NETWORK DUMP command dumps the contents of 3704/3705 storage for NCP, PEP, or EP 3704/3705 control programs, if unit check or IPL required conditions are detected.

The format of the NETWORK command with the DUMP operand is:

NETWORK	**DUMP** *raddr* [**IMMED** / **AUTO** / **OFF**]

where:

raddr
is the real hexadecimal address of the 3704/3705.

IMMED
is the default operand; it forces an immediate dump. The IMMED operand, if specified, does not reload the control program. Before 3704/3705 resources can be used again, the control program must be reloaded. To reload the control program after the NETWORK DUMP raddr IMMED command has executed, use the NETWORK LOAD raddr ncpname command.

If the IMMED operand is specified, a check is made to determine whether the "IPL required" sense status is present. If it is not, the following message occurs:

```
CTLR raddr IPL NOT REQUIRED; ENTER 'YES' TO CONTINUE
```

This pause in operations allows the operator an opportunity to check the NETWORK DUMP command line before engaging or terminating the operation.

AUTO
causes a dump if VM/SP subsequently detects a unit check condition or "IPL required" condition. If AUTO is specified, each time a dump is taken, the Communications Controller is reloaded with the 3704/3705 control program that was previously active.

OFF
resets a previously set AUTO (automatic dump) status.

Note: The dumps produced by the NETWORK command cannot be processed by the IPCSDUMP service program. NETWORK-initiated dumps are processed by the NCPDUMP (Network Control Program DUMP) service program created for this task.

NCPDUMP Service Program and How To Use It

NCPDUMP applies only to dump files that were dumped with the NETWORK DUMP command after the 3704/3705 was loaded by VM/SP.

NCPDUMP is a CMS command. It processes CP spool reader files created by 3705 dumping operations, that is, dump files that are produced as a result of the CP NETWORK command specified with the DUMP operand and either automatic or immediate mode.

The NCPDUMP file processing operation can include:

- Erasing a specific CMS NCPDUMP file after printing it
- Formatting the dump
- Printing the dump
- Assigning an identifier to the CMS NCPDUMP file
- Creating the CMS NCPDUMP file from the spool file.

Although NCPDUMP is a CMS command, its use is restricted to the user identified by the SYSDUMP operand of the SYSOPER macro in DMKSYS during VM/SP system generation. The operation of NCPDUMP is similar to IPCSDUMP operations. A general description of the NCPDUMP operation follows the command description.

The NCPDUMP command has the following format:

NCPDUMP	[**DUMP** *xx*][([**ERASE**][**NOFORM**][**NCPBUFF**][)]]

where:

DUMP*xx*
is the filename of a CMS file containing a 3704/3705 Communications Controller program dump. This dump was created by a previously invoked NCPDUMP command with the ERASE operand not specified.

ERASE
erases the current CP DUMP file or a specified DUMPxx (filename), saved CMS file.

NOFORM
specifies that a formatted control block is not desired.

NCPBUFF
specifies that a formatted listing of the NCP buffer pool is desired.

The NETWORK command invoked with the DUMPxx operand, as stated previously, produces CP files that contain the contents of a designated 3704/3705 Communications Controller unit buffer. These CP files reside as a spooled reader input assigned to a system designated user. The CMS NCPDUMP command invoked by this user formats (if requested) and prints the contents of these files.

The NCPDUMP program creates a CMS file with a filename DUMPxx and a filetype of NCPDUMP, and erases the original spooled NETWORK initiated dump reader file. The created CMS file is erased if you specify ERASE; otherwise it is kept.

A maximum of ten dumped spooled files can be processed and saved, and later recalled, if necessary, by the system assignment of an xx identifier suffix to the CMS DUMPxx filename. The "xx" is a decimal number from 00 to 09, depending on any existing files of a similar name. For example, if the files DUMP00 NCPDUMP and DUMP01 NCPDUMP already exist, the new file would be called DUMP02 NCPDUMP. The file thus created is retained for later use unless the ERASE option is specified, in which case the file is erased immediately after the dump is printed.

Stand-Alone Dump Facility

Overview

With the stand-alone dump facility, you can dump up to 16 Mb (64 Mb in VM/SP HPO) of real storage when VM/SP or VM/SP HPO cannot create a CP Abend dump. This facility dumps all resident pages, CP and non-CP. The stand-alone dump facility cannot dump virtual machine storage and non-resident pages from the paging device.

To use the stand-alone dump program to dump the real storage, you must have access to IPL the real machine. You can IPL the stand-alone dump program from tape or disk and direct the output to tape or printer. When using tape as the output device, reserve the complete tape for the stand-alone dump facility. Basic error recovery is available for DASD, tape, and printer devices used as IPL or output devices.

Typically, an installation can have several stand-alone dump programs generated and ready to run. It would be useful to have the following configurations available for the stand-alone dump facility:

- IPL from tape with output directed to printers
- IPL from tape with output directed to tapes
- IPL from DASD with output directed to printers
- IPL from DASD with output directed to tapes.

These configurations let you take a stand-alone dump with any of the supported possible environments.

The stand-alone dump program communicates with the user with PSW wait codes. Refer to *VM/SP System Messages and Codes* or *VM/SP HPO System Messages and Codes* in "Stand-Alone Dump Facility Wait State Codes." Once the CPU has gone into a wait state, the user can display the PSW, using conventional means, to find if the dump was successful.

The starting and ending addresses of the CP internal trace table are stored in the PSA at locations X'7B0' and X'7B4', respectively, in addition to the PSA locations X'0C' and X'10'.

Refer to Appendix E, "Stand-Alone Dump Formats" on page 353 for information on dump formats.

Devices that You Can Use to IPL Stand-Alone Dump

The following are the devices you can use to IPL the stand-alone dump program:

DASD	**Tape**
3330	2401
3333	2415
3340	2420
3344	3410
3350	3420
3375	3422
3380	3430.

Notes:

1. *If a disk is selected as the IPL device:*
   a. *It cannot be the resident system device*
   b. *It must be CP formatted*
   c. *Cylinder 0 must be allocated as permanent space*
   d. *Cylinder 0 will be used.*
2. *The stand-alone dump IPL tape can be the same as the tape you direct the dump output to.*
3. *Do not try to IPL from a device that is not in the above list.*

Devices to which You Can Send Dump Output

The following are the devices to which the dump output can be directed:

Tape	**Printer**
2401	1403
2415	1443
2420	3211
3410	3203 (Models 4 & 5)
3420	3289E (Model 4)
3422	3262 (Models 1, 5, & 11)
3430	4245
	4248

Notes:

1. *You can specify a maximum of eight real addresses for the dump output device.*

2. *Do not mix printers and tapes in the same list. If you use a printer as the output device, the FCB should match the forms loaded in the printer. If the FCB does not match the form, data may be lost when the printer runs out of paper.*

3. *When you configure the stand-alone dump facility, you can use any printer type or tape type from the list of supported devices.*

   For example, the SADUMP exec prompts you for the output device type with the following:

   ```
   PLEASE ENTER ONE OF THE FOLLOWING OUTPUT DEVICE TYPES:
   PRINTER:  (1403, 1443, 3203, 3211, 3262, 3289, 4245, 4248)
   TAPE:  (2401, 2415, 2420 3410, 3420, OR 3430)
   ```

   If you specify 3420, the system expects the output to be directed to a tape device.

   The system will then request the output device address with the following:

   ```
   PLEASE ENTER REAL OUTPUT DEVICE ADDRESS OR LIST ADDRESSES
   (MAXIMUM OF 8) FOR TAPE:
   ENTRIES IN A LIST MUST BE SEPARATED BY A MINIMUM OF ONE BLANK.
   ```

   You must then respond with the address or list of addresses of the tape device(s) which can receive the output. Be sure the output addresses match the device type (tape in this example); otherwise, results are unpredictable. Keep in mind that the generated stand-alone dump program does not check the address of the device for validity.

   For VM/SP, you can enter up to three digits and for VM/SP HPO, you can enter up to four digits for the real output device address.

4. *The stand-alone dump must have channel 1 defined in the FCB or carriage control tape, if the output device is a printer.*

5. *Do not send the stand-alone dump output to a device that is not included in the above list.*

When you specify tape as the output device, the stand-alone dump program selects, as the dump output device, the first available device in the list, excluding the IPL device (if it is in the list). If you want the stand-alone dump output to go to the IPL tape, make all other devices that are in the list not ready. If no other device within the output address list is available and the IPL tape address is in the list, the IPL device will receive the dump.

If you select a tape for the dump output device, other than the IPL tape, the stand-alone dump facility:

1. Rewinds the tape to ensure that the dump is at the beginning of the tape

2. Sets the density to the highest value for the tape device.

If the tape device selected is the one on which the stand-alone dump facility resides, the facility will write the dump at the same density as the stand-alone dump program was written.

When using tape, reserve the complete tape for the stand-alone dump program; do not put the stand-alone dump program on a tape with the other stand-alone utilities. If you do not want to use the IPL tape as the dump output tape, you may want to put the stand-alone dump program on a mini-reel to make better use of tape resources.

If you select tape to be the output device type, use a single-volume, non-labeled tape for the stand-alone dump program. Be sure that the tape is non-labeled, because the facility does not check to ensure that it is a non-labeled tape.

You can issue the SPTAPE command with the SADump option to move the data from the output tape to a class V reader spool file which is IPCS compatible. To use SPTAPE command when dumping to the IPL tape (that is, if the IPL device address is the same as the dump device address), remember that the tape must be moved to the first tape mark. This tape mark identifies the beginning of the dump. From that point on, you can invoke IPCS to handle the stand-alone dump.

Stand-Alone Dump Program Generation

Your installation can generate the stand-alone dump program to customize the facility to your system configuration. This gives you control over the device used to IPL the stand-alone dump program and the output device for the dump. Invoke the SADUMP EXEC in a CMS virtual machine to do the generation.

Do not call the SADUMP EXEC from within another EXEC. Also, do not queue up the answers ahead of time when running the SADUMP EXEC. To generate a stand-alone dump program, enter "sadump".

To use the SADUMP EXEC:

- You must have R/W access to the A-disk.
- The following files must exist on an accessed disk:
  - DMKSP CNTRL (this is the default if no control is entered)
  - DMKLD00E LOADER
  - LDT DMKSADWT
  - DMKSAD TEXT.

You are asked to answer a series of questions that describe the environment where the stand-alone dump program will run. The SADUMP

EXEC checks all input for validity, and returns messages if you enter invalid data. An example of the prompts and replies that appear on the virtual machine console during SADUMP EXEC execution is shown in "Example for Configuring the Stand-Alone Dump" on page 134.

Following the data that you provide, the SADUMP EXEC does one or more of the following:

- Creates a file with a name of SADGEN[1] ASSEMBLE, and places the file on the user's A-disk. The file has the SAD MACRO with the selected parameters.
- Assembles the SADGEN file to create the SADGEN TEXT file.
- Places the stand-alone dump program in the user's virtual card reader to be IPLed as desired. When the virtual reader is IPLed, the stand-alone dump program will be written on the IPL device.

Notes:

1. *The real device address from which the stand-alone dump program is IPLed is not necessarily the same device address used when it was created.*
2. *It is impossible to verify the dump output address(es) and type at stand-alone dump generation time.*

Using the Stand-Alone Dump Facility

To use the stand-alone dump facility:

1. Configure the stand-alone dump program.
2. Take the stand-alone dump.
3. Process the stand-alone dump from tape to a spool file.

Configuring the Stand-Alone Dump

Before you can use the stand-alone dump program, you must configure the facility. This lets you configure the IPL device and the dump output device(s) for the stand-alone dump facility to match the real I/O.

Use the SADUMP EXEC during configuration to create and assemble the SADGEN ASSEMBLE file. The SADUMP EXEC places an IPLable deck in the virtual card reader. If the system detects an incorrect response, the exec gives an error message to you and requests a new response. For assembly errors, the exec will exit. If the stand-alone dump IPL device is a DASD, it must be CP formatted and the facility will use cylinder 0. Allocate cylinder 0 as permanent space.

[1] SADGEN is the default. The filename will be the same as specified in the SADUMP command if you do not use the default.

The format of the SADUMP command line is:

SADUMP	{ **SP** / **HPO** }	[*filename* / <u>**SADGEN**</u>]

where:

SP
specifies that the stand-alone dump is going to be used on a VM/SP system, which will allow up to a 3-digit dump output real address.

HPO
specifies that the stand-alone dump is going to be used on a VM/SP HPO system, which will allow up to a 4-digit dump output real address.

filename
is the filename of the ASSEMBLE file that has the SAD MACRO. SADGEN ASSEMBLE is the default if you do not supply an operand. (Comply with the CMS guidelines for filenames if a filename is specified).

The following items apply to configuring the stand-alone dump:

1. If the system generated more than one stand-alone configuration, use unique names for each configuration. The default is SADGEN. If you answer "Y" to replace one that already exists (refer to "Example for Configuring the Stand-Alone Dump" on page 134), the original is erased.

2. If you respond with "N" to any of the questions (refer to "Example for Configuring the Stand-Alone Dump" on page 134), the exec will go directly to the next question without doing the indicated work.

3. When the exec asks you to enter the real output device address, you are limited to three digits for VM/SP and four digits for VM/SP HPO.

4. When the exec asks you to enter the control file, if you hit ENTER the exec uses the default of DMKSP.

5. The stand-alone dump configuration deck that the system puts in the virtual card reader is a class D file. You must place the deck in front of all the class D reader files before IPLing.

6. You must IPL the stand-alone dump reader file within a virtual machine. After the IPL'able deck is in your reader, perform the following instructions:

   a. **SET ECMODE ON**
   b. **SPOOL 00C CLASS D**

c. **System CLEAR**
d. **IPL 00C**

Note: Before you IPL the virtual reader, make sure that the IPL device is mounted and ready. If the IPL device is a tape, make sure the write ring is in.

Example for Configuring the Stand-Alone Dump

The following is an example of a stand-alone dump facility generation. In this example:

- You are placing the stand-alone dump program onto a 3330 device with an address of 150.
- The control file in this example is DMKHPO. You may enter any control file you would like to use or take the default of DMKSP.
- The system will send the dump output to the first available 3420 tape drive whose address is 570, 571, 572, 573, 574, 575, 576, or 577.
- The filename of the file that has the SAD MACRO defaults to SADGEN.

Note: In the following example, "= = = = >" indicates data that you enter.

```
   = = = = > SADUMP

The SADUMP EXEC:

   - OPTIONALLY CREATES A NEW SADGEN ASSEMBLE FILE CONTAINING
     A SAD MACRO WITH SELECTED PARAMETERS ON YOUR A-DISK.

   - OPTIONALLY ASSEMBLES THE SADGEN ASSEMBLE FILE.

   - OPTIONALLY PLACES A SADUMP CONFIGURATOR DECK INTO THE
     VIRTUAL CARD READER.

NOTE: YOU MAY EXIT FROM THIS EXEC BY ENTERING 'QUIT' FOR ANY
RESPONSE.

DO YOU WANT TO CREATE A NEW SADGEN MODULE? (Y|N)

   = = = = > Y

PLEASE ENTER THE VIRTUAL DEVICE ADDRESS TO WHICH THE SAD
PROGRAM WILL BE WRITTEN (IPL DEVICE):

   = = = = > 150

PLEASE ENTER ONE OF THE FOLLOWING IPL DEVICE TYPES:
DASD:  (3330, 3333, 3340, 3344, 3350, 3375, 3380)
TAPE:  (2401, 2415, 2420, 3410, 3420, 3422, or 3430)
```

```
    ====> 3330

PLEASE ENTER ONE OF THE FOLLOWING OUTPUT DEVICE TYPES:
PRINTER:  (1403, 1443, 3203, 3211, 3262, 3289, 4245, 4248)
TAPE:  (2401, 2415, 2420 3410, 3420, 3422, or 3430)

    ====> 3420

PLEASE ENTER REAL OUTPUT DEVICE ADDRESS OR LIST ADDRESSES
(MAXIMUM OF 8) FOR TAPE:
ENTRIES IN A LIST MUST BE SEPARATED BY A MINIMUM OF ONE BLANK.

    ====> 570 571 572 573 574 575 576 577

DO YOU WANT TO ASSEMBLE SADGEN NOW? (Y|N)

    ====> Y

ENTER THE CONTROL FILE YOU WANT TO USE.
THE DEFAULT IS DMKSP.

    ====> DMKHPO

THE SADGEN MODULE IS NOW BEING ASSEMBLED.
DO YOU WANT TO PLACE AN IPL'ABLE DECK IN YOUR VIRTUAL
CARD READER? (Y|N)

    ====> Y

AN IPL'ABLE DECK EXISTS IN YOUR VIRTUAL CARD READER
IN CLASS D.  IPL THE READER TO PLACE THE STAND-ALONE
DUMP PROGRAM ON THE IPL DEVICE.
Ready;
```

Note: After you IPL your reader, if no errors occurred, you will receive a wait state code of 912.

Taking a Stand-Alone Dump

If you plan to dump 16 Mb of storage, use a tape density of 1600 or 6250 BPI. A 16 Mb dump may not fit on a tape at 800 BPI. If you are using VM/SP HPO, you may dump up to 64 Mb and should use a tape density of 6250 BPI.

To invoke the stand-alone dump:

1. For multiple processor systems, stop all tightly-coupled processors. Do NOT clear storage.

2. For multiple processor systems, select the processor with the I/O configuration that has access to the resident volume address and the output device address(es).

3. Display locations X'0' to X'B' at the console. The stand-alone dump IPL sequence overlays these bytes, so they cannot be recovered.

4. Do a STORE STATUS operation on the CPU where you will IPL the stand-alone dump program. If you do not do the STORE STATUS, the following will not be saved in low storage:

   - CPU Timer
   - Clock comparator
   - Current PSW
   - Prefix
   - Model dependent features
   - Control registers
   - Floating point registers
   - General purpose registers.

   If the prefix value is not saved in low storage, the information from the prefix page is not available for the formatted section of the dump.

5. Mount and ready the volume that has the stand-alone dump program. If this is a tape, be sure to have the write ring in place.

6. Ready the output device, either tape or printer. If you want the system to place the stand-alone dump on the IPL tape, make all other tapes listed as output devices (at generation time) NOT ready. If you do not want the stand-alone dump on a device that is listed as a possible output device, the device must not be ready at the time you IPL the stand-alone dump program.

7. IPL the stand-alone dump program. The stand-alone dump program will initially write the first nine (eleven for VM/SP HPO) pages of storage to the IPL device. This will provide an area to load the stand-alone dump program and work space. (See "Tape Format" on page 353 and "DASD Format" on page 355 for information about the DASD or tape format.) This step causes the system to place the dump on the output tape or the printer. (See "Tape Format" on page 353 and "Printer Format" on page 356 for information about the format of the output.)

8. When the system enters the wait state, display the PSW. A wait state of 912 indicates successful completion of the stand-alone dump. If the stand-alone dump program is unsuccessful because of some error that you can fix (for example, an unrecoverable I/O error on the output tape):

   a. Correct the error.

   b. Invoke a hardware RESTART to restart the stand-alone dump. (For example, type in RESTART on the appropriate panel on a 4300 processor.)

If you re-IPL the stand-alone dump facility again, part or all the first nine (eleven for VM/SP HPO) pages of storage will be invalid. After the initial IPL, you cannot change the IPL address or IPL volume.

Processing the Stand-Alone Dump Data on Tape

If you directed the output to tape, re-IPL VM/SP or VM/SP HPO. Then issue the SPTAPE command with the LOAD raddr and SADUMP operands to create an IPCS compatible spool file. This is the only way to transfer the data. After the system has created the spool file, enter the IPCS command, IPCSDUMP, to process the stand-alone dump. For more information on CP Abend Dumps see "Reading CP Abend Dumps" on page 76.

Chapter 4. Debugging CMS

Debugging CMS

This section describes the debug tools that Conversational Monitor System (CMS) provides. These tools can be used to help you debug CMS or a problem program. In addition, a CMS user can use the Control Program (CP) commands to debug. Information that is often useful in debugging is also included.

Debugging Commands

There are commands that are useful in debugging such as:

- PER and ADSTOP, which set breakpoints (address stops) that stop program execution at specific locations.

- TRACE, which traces specific virtual machine activity and records the results on the terminal or printer.

- DISPLAY, which displays the contents of:

  - Channel Address Word (CAW)
  - Channel Status Word (CSW)
  - Old Program Status Word (PSW)
  - General purpose registers (GPR)
  - Virtual storage

  at the terminal.

- STORE, which:

  - Changes the contents of the control words (CAW, CSW, and PSW) and general purpose registers

  - Stores data in virtual storage locations.

- VMDUMP, which dumps virtual storage in a different format than the DUMP command; the output produced by VMDUMP can be processed by IPCS.

- DUMP, which dumps all or part of virtual storage at the printer.

The SVCTRACE command records information for all SVC calls. When the trace is terminated, the information recorded up to that point is printed at the system printer.

In addition, several CMS commands produce or print load maps. These load maps are often used to locate storage areas while debugging programs.

Using the SVCTRACE command

If your program issues many SVCs, you may not get all of the information you need using the CP TRACE command. The SVCTRACE command is a CMS command, which provides more detailed information about all SVCs in your program, including:

- Register contents before and after the SVC
- Name of the called routine and the location from which it was called
- Contents of the parameter list passed to the SVC.

See *VM/SP CMS Command Reference* for the format of the SVCTRACE command.

The SVCTRACE command has only two operands, ON and OFF, to begin and end tracing. SVCTRACE information can be directed only to the printer, so you do not receive trace information at the terminal.

Since the SVCTRACE command can only be entered from the CMS environment, you must use the Immediate commands SO (suspend tracing) or HO (halt tracing) if you want tracing to stop while a program is executing. Use the Immediate command RO to resume tracing.

Since the CMS system is "SVC-driven," this debugging technique can be useful, especially, when you are debugging CMS programs. For more information on writing programs to execute in CMS, see *VM/SP CMS for System Programming.*

Tracing Capabilities in EXECs

It may be very helpful to trace EXECs that are used to diagnose problems. By tracing the EXEC, you are able to follow the execution of the EXEC and see intermediate values that otherwise might not be obvious to the user. There are three EXEC processors:

- System Product Interpreter
- EXEC 2
- CMS EXEC.

The amount of information displayed during execution of an EXEC is controlled by a single instruction. The instruction depends upon which processor is being used as shown below:

Processor	**Instruction**
System Product Interpreter	TRACE
EXEC 2	&TRACE

CMS EXEC &CONTROL

Tracing can also be turned on for the System Product Interpreter or EXEC 2 by entering the following CMS command:

SET EXECTRAC ON

This causes the tracing bit in the EXECFLAG in NUCON to be turned on and allows tracing without program modification.

The TRACE instruction used by the System Product Interpreter has several options to control how much information is displayed to the user. The TRACE instruction even allows you to enter interactive debug. During interactive debug, the interpreter pauses after almost every instruction allowing the user to single-step through the program.

Assume that we have a Restructured Extended Executor (REXX) program called STATUS EXEC, which gives us some status information. The contents of STATUS EXEC follows:

```
/*This EXEC gives user some status information.*/
trace ?i
say 'Userid: ' userid()
say 'Time  : ' time()
say 'Date  : ' date('w')',' date()
exit
```

Notice the command **trace ?i**, which is the second line of the program. This command causes the program to go into interactive debug and to trace:

- All clauses before execution
- Intermediate results during evaluation of expressions
- Substituted names.

When the STATUS EXEC is executed *without* the trace command, you get a result that could look like this:

```
Userid:  GEORGEB
Time  :  09:50:54
Date  :  Thursday, 3 Apr 1986
```

When the STATUS EXEC is executed *with* the trace command, you get a result that could look like this:

```
     3 *-* say 'Userid: ' userid()
       >L>   "Userid: "
       >F>   "GEORGEB"
       >O>   "Userid:  GEORGEB"
Userid:  GEORGEB
       +++ Interactive trace. "Trace Off" to end debug, ENTER to continue. +++
```

At this point, you either type:

TRACE OFF

to end debug or hit the **ENTER** key to continue executing and you get a result that could look like this:

```
      4 *-* say 'Time   :  ' time()
        >L>    "Time   :  "
        >F>    "09:50:54"
        >O>    "Time   :   09:50:54"
Time   :  09:50:54
```

At this point, you either type:

TRACE OFF

to end debug or hit the **ENTER** key to continue executing and you get a result that could look like this:

```
      5 *-* say 'Date   :  ' date('w')',' date()
        >L>    "Date   :  "
        >L>    "w"
        >F>    "Thursday"
        >O>    "Date   :   Thursday"
        >L>    ", "
        >O>    "Date   :   Thursday,"
        >F>    "3 Apr 1986"
        >O>    "Date   :   Thursday, 3 Apr 1986"
Date   :  Thursday, 3 Apr 1986
```

At this point, you either type:

TRACE OFF

to end debug or hit the **ENTER** key to continue executing and you get a result that could look like this:

```
      6 *-* exit
```

As you can see in the previous example, the intermediate results of steps three through six of STATUS EXEC were traced and execution stopped at each step.

The System Product Interpreter also has a TRACE function. See *VM/SP System Product Interpreter Reference* for more information on using the TRACE instruction and TRACE function.

Nucleus Load Map

Each time the CMS resident nucleus is loaded on a DASD and an IPL can be performed on that DASD, a nucleus load map is produced as a printer spool file. Save this load map. It lists the virtual storage locations of nucleus-resident routines and work areas. Transient modules are not included in this load map. When debugging CMS, you can locate routines using this map. For information on obtaining a load map, see the *VM/SP Installation Guide* or *VM/SP HPO Installation Guide.*

Module Load Map

The module load map of a disk-resident command module contains the location of control sections and entry points loaded into storage. It may also contain certain messages and card images of any invalid cards or replace cards that exist in the loaded files. The load map is contained in the third record of the MODULE file.

This load map is useful in debugging. When using the Debug environment to analyze a program, use the program's load map to help in displaying information.

There are two ways to get a load map:

- When loading relocatable object code into storage, make sure that the MAP option is in effect when the LOAD command is issued. Since MAP is the default option, just be sure that NOMAP is not specified. A load map is then created on the primary disk each time a LOAD command is issued.

- When generating the absolute image form of files already loaded into storage, make sure that the MAP option is in effect when the GENMOD command is issued. Since MAP is the default option, just be sure that NOMAP is not specified. Issue the MODMAP command to type the load map associated with the specified MODULE file on the terminal. The format of the MODMAP command is:

MODmap	*filename*

where:

filename
is the module whose map is to be displayed. The filetype must be MODULE.

How to Print a CMS Dump File

Use the PRTDUMP command to print a previously created dump file under CMS. See "PRTDUMP Command" on page 251 for more information.

Reading CMS Abend Dumps

If an abend dump is desired when CMS abnormally terminates, the terminal operator may enter:

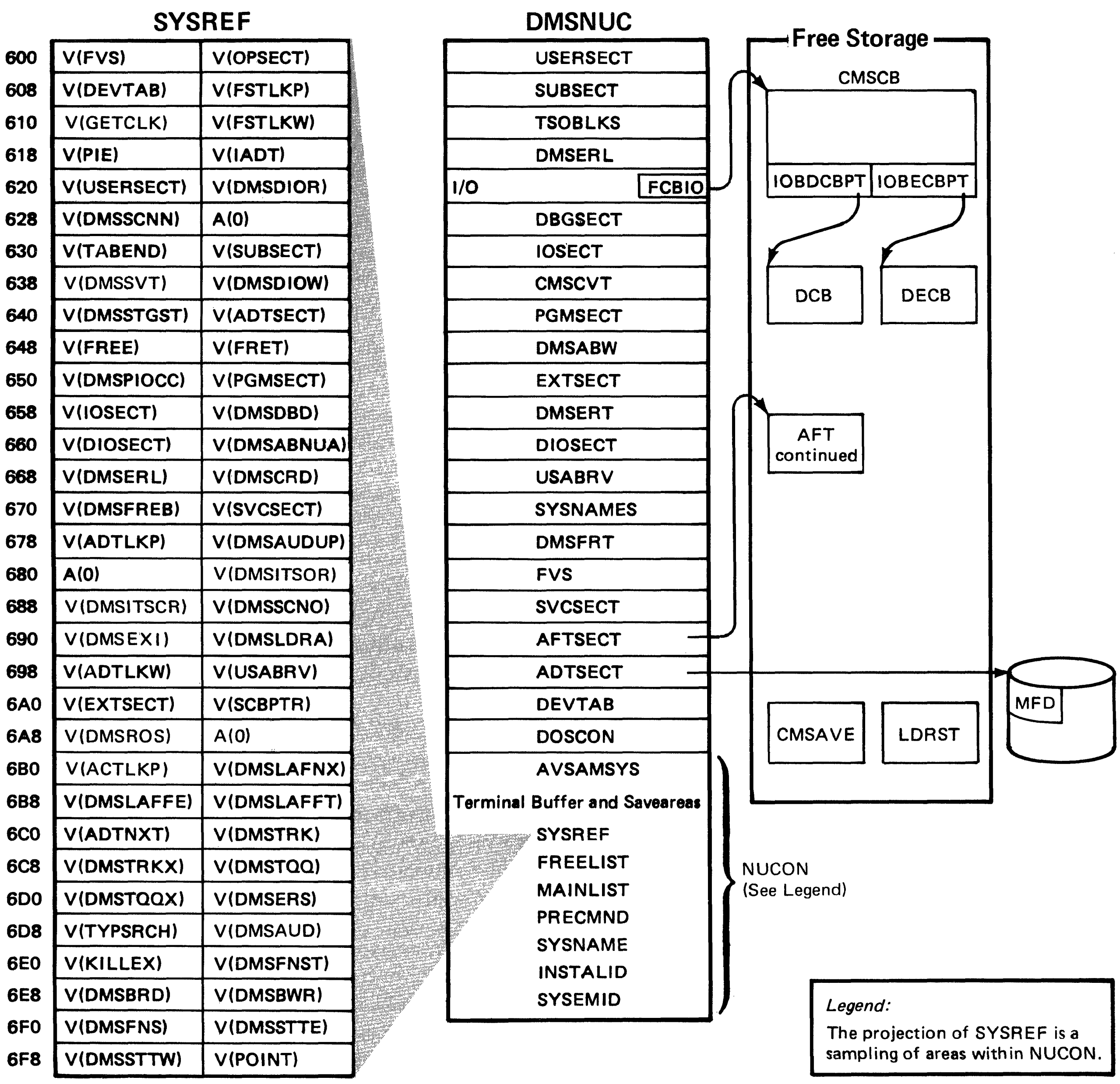
```
#CP VMDUMP 0-END FORMAT CMS DSS
```

By issuing these commands a dump spool file is created and sent to your reader. Now the system must be RE-IPLed and then issue the IPCSDUMP command which will format the dump into a usable form. The dump formats and prints:

- General Purpose Registers (GPRs)
- Extended control registers
- Floating-point registers
- Storage boundaries with their corresponding storage protect key
- Current PSW
- Selected storage.

Storage is printed in hexadecimal representation, eight words to the line, with EBCDIC translation at the right. The hexadecimal storage address corresponding to the first byte of each line is printed at the left.

When CMS can no longer continue, it abnormally terminates. To debug CMS, first determine the condition that caused the abend and then find why the condition occurred. To find the cause of a CMS problem, you must be familiar with the structure and functions of CMS. Refer to *VM/SP CMS for System Programming* for functional information on CMS. The following discussion on reading CMS dumps refers to several CMS control blocks and fields in the control blocks. Refer to the *VM/SP Data Areas and Control Block Logic Volume 2 (CMS)* for details on CMS control blocks. Figure 12 on page 146 shows the CMS control block relationships. You also need a current CMS nucleus load map to analyze the dump.



SYSREF

600	V(FVS)	V(OPSECT)
608	V(DEVTAB)	V(FSTLKP)
610	V(GETCLK)	V(FSTLKW)
618	V(PIE)	V(IADT)
620	V(USERSECT)	V(DMSDIOR)
628	V(DMSSCNN)	A(0)
630	V(TABEND)	V(SUBSECT)
638	V(DMSSVT)	V(DMSDIOW)
640	V(DMSSTGST)	V(ADTSECT)
648	V(FREE)	V(FRET)
650	V(DMSPIOCC)	V(PGMSECT)
658	V(IOSECT)	V(DMSDBD)
660	V(DIOSECT)	V(DMSABNUA)
668	V(DMSERL)	V(DMSCRD)
670	V(DMSFREB)	V(SVCSECT)
678	V(ADTLKP)	V(DMSAUDUP)
680	A(0)	V(DMSITSOR)
688	V(DMSITSCR)	V(DMSSCNO)
690	V(DMSEXI)	V(DMSLDRA)
698	V(ADTLKW)	V(USABRV)
6A0	V(EXTSECT)	V(SCBPTR)
6A8	V(DMSROS)	A(0)
6B0	V(ACTLKP)	V(DMSLAFNX)
6B8	V(DMSLAFFE)	V(DMSLAFFT)
6C0	V(ADTNXT)	V(DMSTRK)
6C8	V(DMSTRKX)	V(DMSTQQ)
6D0	V(DMSTQQX)	V(DMSERS)
6D8	V(TYPSRCH)	V(DMSAUD)
6E0	V(KILLEX)	V(DMSFNST)
6E8	V(DMSBRD)	V(DMSBWR)
6F0	V(DMSFNS)	V(DMSSTTE)
6F8	V(DMSSTTW)	V(POINT)

Legend:
The projection of SYSREF is a sampling of areas within NUCON.

Figure 12. CMS Control Blocks

Reason for the Abend

Determine the immediate reason for the abend and identify the failing module. The abend message DMSABN148T contains an abend code and failing address. The *VM/SP System Messages and Codes* or *VM/SP HPO System Messages and Codes* manual lists all the CMS abend codes, identifies the module that caused the abend, and describes the action that should be taken whenever CMS abnormally terminates.

You may have to examine several fields in the nucleus constant area (NUCON) of low storage.

1. Examine the program old PSW (PGMOPSW) at location X'28'. Using the PSW and current CMS nucleus load map, determine the failing address.

2. Examine the SVC old PSW (SVCOPSW) at location X'20'.

3. Examine the external old PSW (EXTOPSW) at location X'18'. If the virtual machine operator terminated CMS, this PSW points to the instruction executing when the termination request was recognized.

4. For a machine check, examine the machine check old PSW (MCKOPSW) at location X'30'. Refer to Figure 70 on page 352 for a description of the PSW.

Collect Information

Examine several other fields in NUCON to analyze the status of the CMS system. As you proceed with the dump, you may return to NUCON to pick up pointers to specific areas (such as pointers to file tables) or to examine other status fields. The complete contents of NUCON and the other CMS control blocks are described in the *VM/SP Data Areas and Control Block Logic Volume 2 (CMS).* The following areas of NUCON may contain useful debugging information.

- Save Area for Low Storage

  Before executing, DEBUG saves the first 160 bytes of low storage in a NUCON field called LOWSAVE. LOWSAVE begins at X'C0'.

- Register Save Area

  DMSABN, the abend routine, saves the user's floating-point and general purpose registers.

Field	Location	Contents
FPRLOG	X'160'	User floating-point registers
GPRLOG	X'180'	User general purpose registers
ECRLOG	X'1C0'	User extended control registers

- Device

  The name of the device causing the last I/O interrupt is in the DEVICE field at X'26C'.

- Last Two Commands or Procedures Executed

Field	Location	Contents
LASTCMND	X'2A0'	Last command issued from the CMS or XEDIT command line. If a command issued in a CMS EXEC abnormally terminates, this field contains the name of the command. When a CMS EXEC completes, this field contains the name 'EXEC.' EXEC 2 and System Product Interpreter do not update this field.
PREVCMND	X'2A8'	Next-to-last command issued from the CMS or XEDIT command line. If a command issued in a CMS EXEC abnormally terminates, this field contains the name 'EXEC'. When a CMS EXEC completes, this field contains the last command issued from the CMS EXEC. EXEC 2 and System Product Interpreter do not update this field.
LASTEXEC	X'2B0'	Last EXEC procedure invoked. EXEC 2 and System Product Interpreter do not update this field.
PREVEXEC	X'2B8'	Next to last EXEC procedure invoked. EXEC 2 and System Product Interpreter do not update this field.

- Last Module Loaded into Free Storage and the Transient Area

  The name of the last module loaded into free storage via a LOADMOD is in the field LASTLMOD (location X'2C0'). The name of the last module loaded into the transient area via a LOADMOD is in the field LASTTMOD (location X'2C8').

- Pointer to CMSCB

  The pointer to the CMSCB is in the FCBTAB field located at X'5C0'. CMSCB contains the simulated OS control blocks. These simulated OS control blocks are in free storage. The CMSCB contains a PLIST for

CMS I/O functions, a simulated Job File Control Block (JFCB), a simulated Data Event Block (DEB), and the first in a chain of I/O Blocks (IOBs).

The last command entered from the terminal is stored in an area called CMNDLINE (X'7A0'), and its corresponding PLIST is stored at CMNDLIST (X'848').

- External Interrupt Work Area

  EXTSECT is a work area for the external interrupt handler. It contains:

  – The PSW, EXTPSW.
  – Register save areas, EXSAVE1.
  – Separate area for timer interrupts, EXSAVE.

- I/O Interrupt Work Area

  IOSECT is a work area for the I/O interrupt handler. The oldest and newest PSW and CSW are saved. Also, there is a register save area.

- Program Check Interrupt Work Area

  PGMSECT is a work area for the program check interrupt handler. The old PSW and the address of register 13 save area are stored in PGMSECT.

- SVC Work Area

  SVCSECT is a work area for the SVC interrupt handler. It also contains the first four register save areas assigned. The SFLAG indicates the mode of the called routine. Also, the SVC abend code, SVCAB, is located in this CSECT.

- Simulated Communications Vector Table (CVT)

  The CVT, as supported by CMS, is CVTSECT. Only the fields supported by CMS are filled in.

- Active Disk Table and Active File Table

  For file system problems, examine the Active Disk Table (ADT), or Active File Table (AFT) in NUCON.

See a CMS nucleus map for the location of these CSECTs.

Debugging CMS

Register Use

To trace control blocks and modules, it is important to know the CMS GPR usage conventions.

Register	Contents
GPR1	Address of the PLIST
GPR12	Program's entry point
GPR13	Address of a 12-doubleword work area for an SVC call
GPR14	Return address
GPR15	Program entry point or the return code

The preceding information should help you to read a CMS dump. If it becomes necessary to trace file system control blocks, refer to Figure 12 on page 146 for more information. With a dump, the control block diagrams, and a CMS load map, you should be able to find the cause of the abend.

Tips for debugging after receiving a program check abend (e.g. DMSITP141) are as follows:

- DMSITP, the CMS program interrupt handler, issues error messages when a program check occurs. If a SPIE or a STAE has been issued, control is passed to the specified routine; otherwise control passes to DMSABN to try to recover from the error. If the message DMSITP144T is issued, the UFDBUSY byte is not zero and control is halted after the message is typed. If the wait state bit is turned off in the PSW, control continues as above. Also, if the error occurred during the execution of a system routine, control is halted until the wait state bit is turned off or CMS is re-IPLed.

- To determine the registers and PSW at the time of the abend, get the address of PGMSECT in the nucleus constant area (NUCON X'654'). The old PSW is stored 12 (X'C') bytes into the DSECT, immediately followed by registers 14, 15, 0, 1, and 2. The Program Interrupt Element (PIE), needed by SPIE, primarily uses these areas. Registers 0 through 15 are stored at location X'3C' into the DSECT. The SPIE/STAE routine or the DMSSAB routine uses the other areas within the DSECT.

- Another aid to debugging is the SVC save area (SVCSAVE) for the virtual machine. Location X'528' in NUCON points to these areas. The save areas are easily recognizable by the check words 'ABCD' and 'EFGH' contained within them. The address of the SVC caller is stored at location 4 and the name of the routine being called is saved at location 8. At location X'10', the old PSW is stored, followed by the addresses for the normal return and the error return. The registers 0 through 15 are stored at location X'20', followed by the floating point register at X'60'. After the first check word ('ABCD'), the address of the next SVCSAVE area is stored, followed by the address of the previous SVCSAVE area and the address of the user's area. If the address of the next or previous SVCSAVE area is zero, the chain is terminated.

Commands that Alter the Contents of Storage

You can use the STORE and STCP commands to alter the contents of virtual machine storage and real storage.

ZAP and ZAPTEXT commands are used to alter modules, OS LOADLIBS, TEXT libraries, or TEXT decks before the code is loaded and executed.

ZAP and ZAPTEXT are described in *VM/SP Installation Guide* or *VM/SP HPO Installation Guide*. See "Altering the Contents of Virtual Machine Storage (STORE command)" on page 65 and "Altering the Contents of Real Storage (STCP command)" on page 67 for information on STORE and STCP.

Chapter 5. Debugging GCS

Debugging GCS

While running programs on the Group Control System (GCS), you can encounter four types of problems:

- Loops
- Abends
- Incorrect results
- Endless wait states.[2]

To help you deal with these problems, GCS provides:

- Internal tracing facilities (See page 154.)
- External tracing facilities (See page 172.)
- Dumping facilities (See page 180.)
- Interactive Debugging Support (See page 182.).

Internal Tracing Facilities

In common storage, the GCS supervisor maintains a wraparound[3] trace table that serves all virtual machines in a group. When building your GCS configuration file, you specify how big you want this table to be. The minimum you can choose is 4K; the maximum depends upon how much common storage you have available to use. If you don't set a size limit, GCS gives you a default size of 16K. See the *VM/SP Installation Guide* or *VM/SP HPO Installation Guide* for more information about defining a GCS group configuration file.

This table contains information about the following supervisor events:

- Task dispatches
- External interrupts
- I/O interrupts
- Program interrupts
- SVC interrupts
- I/O requests (SIO, Diagnose, HDV, TIO) (invoked by supervisor)
- APPC/VM Synchronous event
- IUCV Signal System Service detail entries
- SVC GETMAIN storage requests
- SVC FREEMAIN storage requests
- Branch Entry FREEMAIN storage requests
- Branch Entry GETMAIN storage requests.

Besides tracing supervisor events, this table can record data from any of your GCS programs. The internal tracing of supervisor events is activated as soon as your virtual machine joins a group. Activating internal tracing of program data (GTRACE events) in your virtual machine involves the

[2] Outlined in Chapter 1, "Introduction to Debugging" on page 1.

[3] "Wraparound" means that, when the table fills, it goes back to the top and starts writing over itself.

following: issuing the ITRACE command and then issuing the GTRACE macro.

Using the ITRACE Command and GTRACE Macro

To begin tracing data in a virtual machine, you must issue, from the console, the ITRACE command with the GTRACE option. Then the GCS application program you want to trace must call the GTRACE macro (The GTRACE macro cannot begin tracing unless you first issue the ITRACE command.).

You can issue the ITRACE command for:

- Individual virtual machines, or
- Entire virtual machine group.

But any virtual machine operator who issues it on behalf of the whole group (ITRACE GROUP) must have an authorized userid.

For more information about the ITRACE command and the GTRACE macro, see the *VM/SP Group Control System Command and Macro Reference.*

Formats of Internal Trace Entries

Entries in the internal trace table come in two different formats:

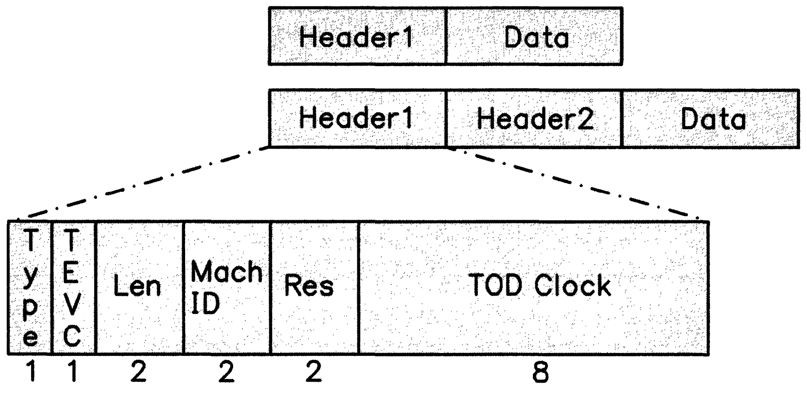
Supervisor entries:

Header1	Data

GTRACE entries:

Header1	Header2	Data

The Header1 that both entries share looks like this:



where:

Type

Shows the type of trace entry:

X'01'	=	Dispatcher
X'02'	=	External Interrupt
X'03'	=	I/O Interrupt
X'04'	=	Program Interrupt
X'05'	=	SVC Interrupt
X'06'	=	I/O Request
X'07'	=	IUCV Signal System Service details
X'08'	=	SVC GETMAIN Request
X'09'	=	SVC FREEMAIN Request
X'0A'	=	GETMAIN Request via branch entry
X'0B'	=	FREEMAIN Request via branch entry
X'0C'	=	APPC/VM Synchronous event entry
X'0E'	=	GTRACE macro data.

TEVC

(Trace Entry Verification Code) keeps track of every time the table "wraps around." The first set of entries will have a TEVC of zero (X'00'). Each time the table wraps, this number increases by one until it reaches X'FF'. After that, it recycles to X'00'.

By looking at this number, you'll be able to identify entries left over from the last "wrap" of the table. This could be important, for example, in a case where the GCS supervisor secures a trace table slot and then gets interrupted by CP before storing a new entry there. That slot would remain reserved, but unused, by the interrupted machine. Other machines in the group, when dispatched by CP, would create trace table entries in slots following it.

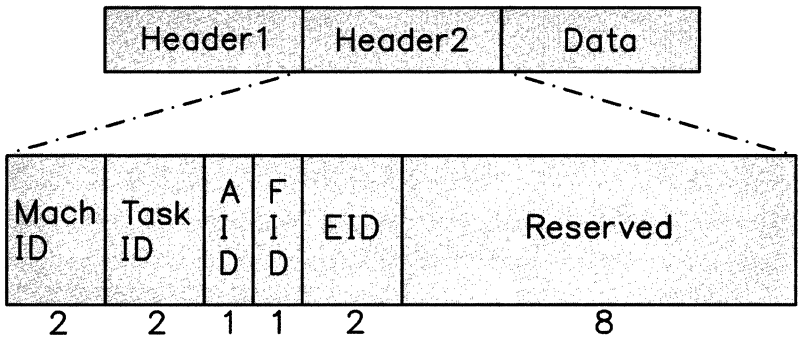
Len
Contains the length of the whole entry, including this header.

Mach ID
Identifies the virtual machine associated with this entry. There is a single trace table for the entire GCS group, it is important that you have the proper virtual machine identification (Mach ID).

Res
Represents a two-byte reserved field.

TOD Clock
Shows what time this entry was created in time-of-day format.

The Header2 used with GTRACE entries looks like this:



where:

Mach ID
Identifies the virtual machine associated with this entry. There is a single trace table for the entire GCS group, it is important that you have the proper virtual machine identification (Mach ID).

Task ID
Identifies the task being traced.

AID
Indicates this is a "data" record. It always contains X'FF'.

FID
(Format ID) identifies what formatting module handles this entry.

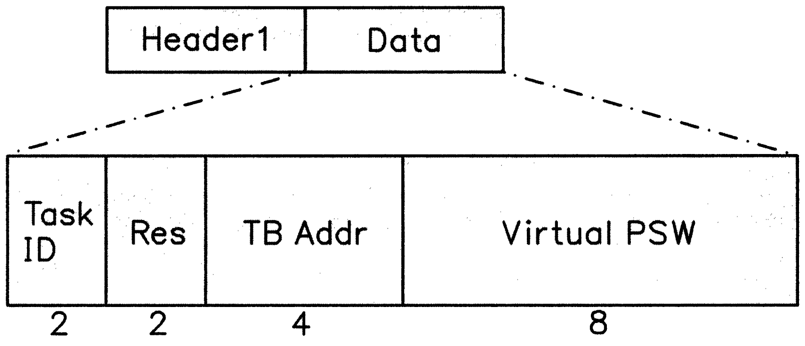
EID
Contains information from the GTRACE macro's ID parameter.

Reserved
Represents an eight-byte reserved field.

The data portion of supervisor entries can have any of twelve different formats:

- Dispatcher (type X'01'), see page 159
- External Interrupt (type X'02'), see page 160
- I/O Interrupt (type X'03'), see page 161
- Program Interrupt (type X'04'), see page 161
- SVC Interrupt (type X'05'), see page 162
- SIO (type X'06'), see page 163
- IUCV Signal System Service (type X'07'), see page 164
- Getmain via SVC (type X'08'), see page 165
- Freemain via SVC (type X'09'), see page 166
- Branch Entry Getmain (type X'0A'), see page 168
- Branch Entry Freemain (type X'0B'), see page 169
- APPC/VM Synchronous Event (type X'0C'), see page 170

- **Dispatcher** (type X'01')



where:

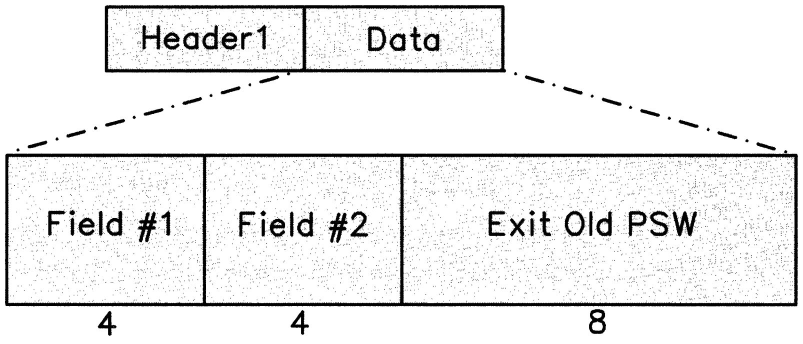
Task ID
Identifies the task being traced.

Res
Represents a reserved field.

TB Addr
Holds the address of a task control block for the task being dispatched.

Virtual PSW
Contains the virtual PSW being dispatched.

- **External Interrupt** (type X'02')



where:

Field #1
The value of this field depends on the type of external interrupt. The interrupt code is stored in bytes 2 and 3 of the EXT old PSW.

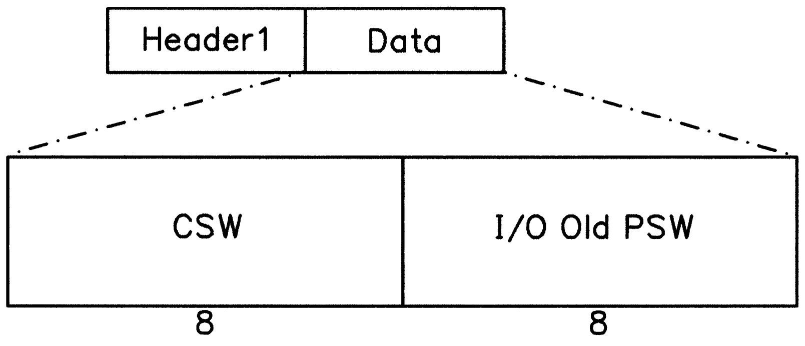
- For a Timer interrupt (code X'1004') this is a reserved field.
- For an IUCV interrupt (code X'4000'), it contains:
  - 2-byte IPPATHID
  - 1-byte IPFLAGS1
  - 1-byte IPTYPE.
- For all other types of external interrupts this is a reserved field.

Field #2
The value of this field depends on the type of external interrupt. The interrupt code is stored in bytes 2 and 3 of the EXT old PSW.

- For a Timer interrupt (code X'1004') it contains a pointer to the Timer Queue Element.
- For an APPC/VM interrupt (code X'4000' with an IPTYPE of X'81', X'82', X'83', X'87', X'88', or X'89'), it contains:
  - 2-byte IPCODE
  - 1-byte IPWHATRC - for a connect pending (type X'81') interrupt, this byte will contain the IPFLAGS2 field.
  - 1-byte IPSENDOP.
- For all other types of external interrupts this is a reserved field.

Ext Old PSW
Contains the external old PSW. If an IUCV poll (rather than an external interrupt) generates this entry, the external old PSW will contain zeros (except for the interrupt code).

- **I/O Interrupt** (type X'03')

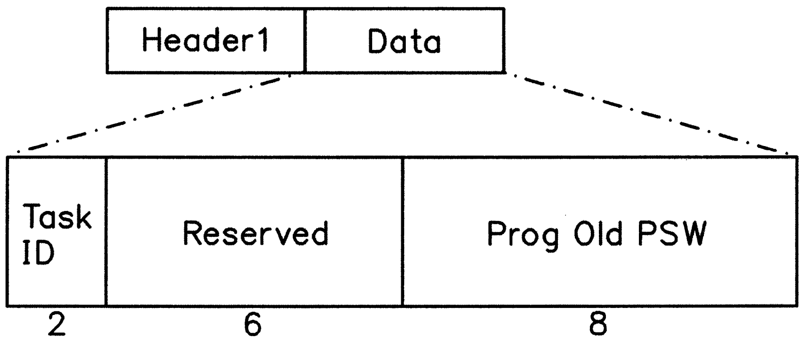


where:

CSW
Contains the channel status word.

I/O Old PSW
Contains the old I/O PSW.

- **Program Interrupt** (type X'04')



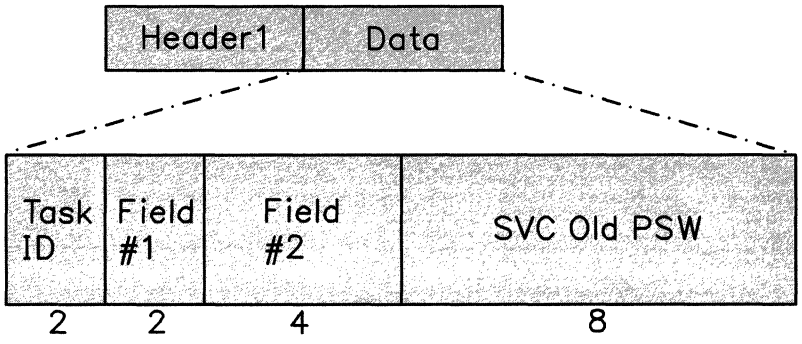
where:

Task ID
Identifies the task being traced.

Reserved
Represents a reserved field.

Prog Old PSW
Contains the program old PSW.

- **SVC Interrupt** (type X'05')



where:

Task ID
Identifies the task being traced.

Field #1
Represents a reserved field for all but three SVCs.

For SVC 202, it contains the first two bytes of the requested function's name. (For example, "GE" for the GENIO function.)

For SVC 203, it contains a two-byte flag and code parameter.

For a DOS SVC, the leftmost bit of this field is set to one, and the rest of the two bytes is zeros (reserved).

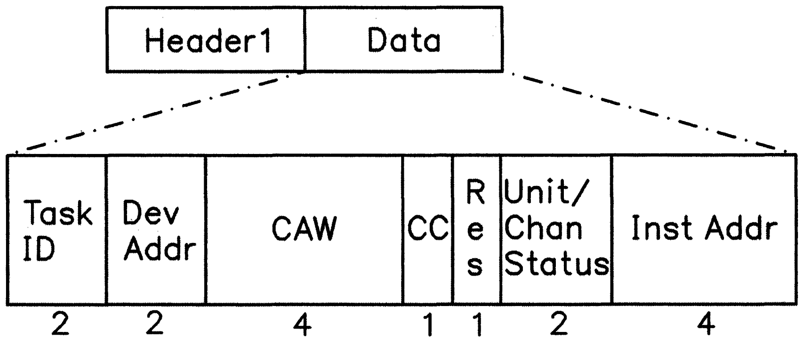
Field #2
Shows the contents of Register 1 for all SVCs except 202.

For SVC 202, it contains the middle four bytes of the requested function's name.

SVC Old PSW
Contains the entire eight bytes of the SVC old PSW for all SVCs but 202.

For SVC 202, it contains the last two bytes of the function name that's being invoked, followed by the last six bytes of the SVC old PSW.

•**SIO** (type X'06')



where:

Task ID
Identifies the task being traced.

Dev Addr
Holds the virtual address of a device where the operation is being directed. (For a Test Channel instruction, this is the virtual channel address.)

CAW
Contains the channel address word. However, for instructions like HDV and TCH that don't use CAWs, this contains zeros.

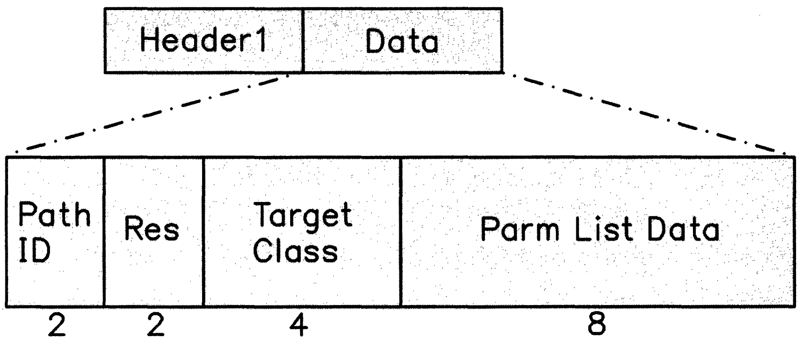
CC
Contains the condition code from the SIO event. The field is relevant only for SIOF and DIAG X'98' SIO entries.

Res
Represents a reserved field.

Unit/Chan Status
Contains the unit and channel status bytes from the stored CSW. This field is relevant only for SIOF and DIAG X'98' SIO events, and only when the CC field = X'01'.

Inst Addr
contains the address of the I/O instruction (SIO, SIOF, TIO, HIO, HDV, CLRIO, TCH, DIAG X'18', DIAG X'20', or DIAG X'98').

- **IUCV Signal System Service** (type X'07')



where:

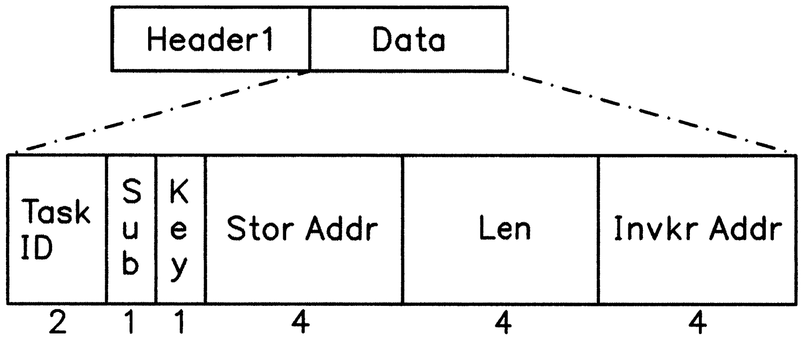
Path ID
: Identifies a two-byte IUCV path.

Res
: Represents a reserved field.

Target Class
: Identifies an IUCV target class containing the interrupt source's Signal ID and type of signal sent.

Parm List Data
: Contains IUCV parameter list data.

- **Getmain via SVC** (type X'08')



where:

Task ID
Identifies the task being traced.

Sub
Identifies the subpool of storage being requested. It contains zeros when:

- An SVC 4 fails because of an incorrect parameter list address.
- The GETMAIN fails because of an invalid mode byte.

Key
Contains the key of storage being obtained. It contains zeros when:

- An SVC 4 fails because of an incorrect parameter list address.
- The GETMAIN fails because of an invalid mode byte.
- If either the length or the subpool is incorrect, or both.

Note: The key data in this byte is right-justified. The rightmost bit of this field serves as a fetch-protection signal. If the subpool of storage you request is **not** fetch-protected, this bit will be 0 (zero).

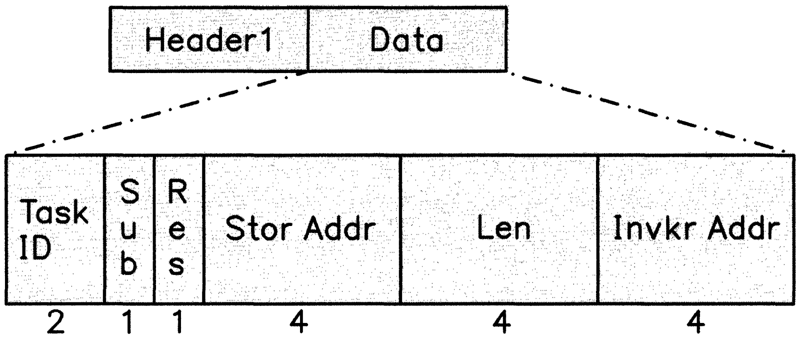
Stor Addr
Contains the address of storage obtained. If the GETMAIN failed, it contains zeros.

Len
Contains the length of the storage requested. It contains zeros when:

- An SVC 4 fails because of an incorrect parameter list address.
- The GETMAIN fails because of an invalid mode byte.

Invkr Addr
Identifies the invoker's address (the address that follows the SVC).

- **Freemain via SVC** (type X'09')



where:

Task ID
Identifies the task being traced.

Sub
Identifies the subpool of storage being released. If the FREEMAIN fails, it contains the subpool associated with the FREEMAIN.

It contains zeros for the following failures:

- SVC 5 is issued with an invalid parameter list address.
- An unsupported MVS parameter is specified on the FREEMAIN macro.
- An invalid mode byte is encountered.

Res
Represents a reserved field.

Stor Addr
Contains the address of storage being released. If the FREEMAIN fails, it contains the storage address passed to FREEMAIN.

It contains zeros for the following failures:

- SVC 5 is issued with an invalid parameter list address.
- An unsupported MVS parameter is specified on the FREEMAIN macro.
- An invalid mode byte is encountered.

Len
Contains the length of the storage released. If the FREEMAIN fails, it contains the length passed to FREEMAIN.

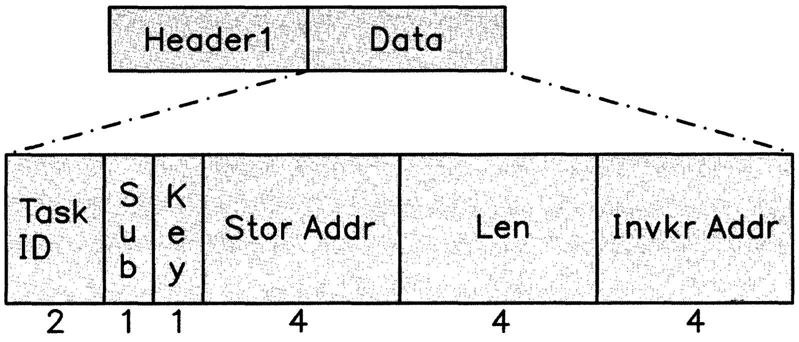
It contains zeros for the following failures:

- SVC 5 is issued with an invalid parameter list address.

- An unsupported MVS parameter is specified on the FREEMAIN macro.
- An invalid mode byte is encountered.

Invkr Addr
Identifies the invoker's address (the address that follows the SVC).

- **Branch Entry Getmain** (type X'0A')



where:

Task ID
Identifies the task being traced.

Sub
Contains the subpool specified in the GETMAIN request.

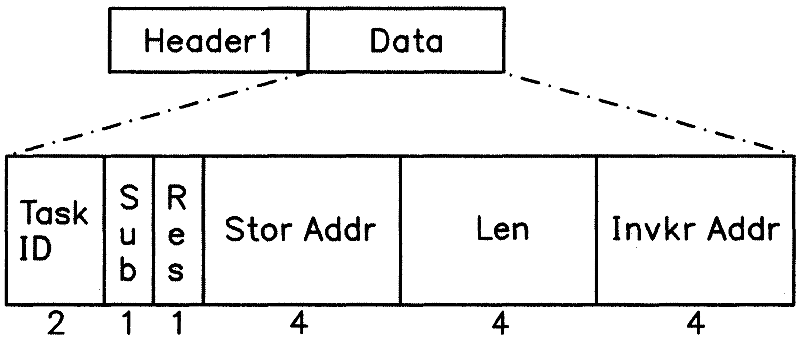
Key
The key data for a branch entry will always be filled in.

Stor Addr
Contains the address of storage obtained. If the GETMAIN failed, it contains zeros.

Len
Contains the length of the storage requested.

Invkr Addr
identifies the address following the GETMAIN call.

- **Branch Entry Freemain** (type X'0B')



where:

Task ID
Identifies the task being traced.

Sub
Contains the subpool specified in the FREEMAIN request.

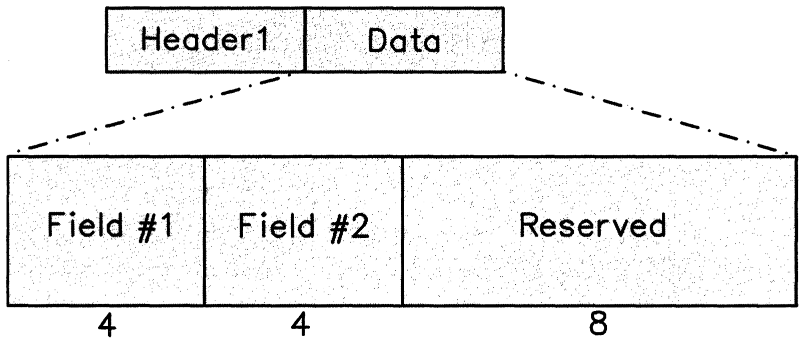
Res
Represents a reserved field.

Stor Addr
Contains the address of storage being released. If the FREEMAIN fails, it contains the storage address passed to FREEMAIN.

Len
Contains the length of the storage released. If the FREEMAIN fails, it contains the length passed to FREEMAIN.

Invkr Addr
identifies the address following the FREEMAIN call.

- **APPC/VM Synchronous Event** (type X'0C')



where:

Field #1
contains:

- 2-byte IPPATHID
- 1-byte IPFLAGS1
- 1-byte IPTYPE.

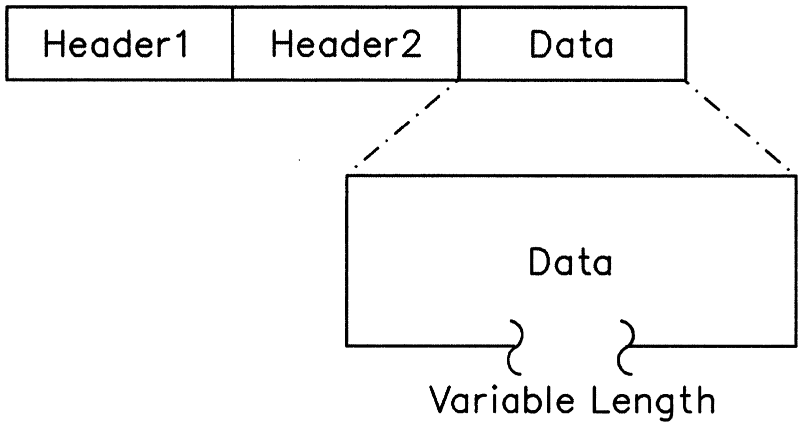
Field #2
contains:

- 2-byte IPCODE
- 1-byte IPWHATRC - for a connect pending (type X'81') interrupt, this byte will contain the IPFLAGS2 field.
- 1-byte IPSENDOP.

Reserved
is a reserved field.

The data portion of the GTRACE entries looks like this:

- **GTRACE** (type X'0E')



where:

Data
Represents from 1 up to 256 bytes of data passed from the application by the GTRACE macro.

How to Look at Internal Trace Table Entries

FLSCTB contains the common trace block address. The common trace block contains pointers in the following order:

1. Beginning of the table
2. End of the table
3. Next available entry slot in the table.

The next available entry immediately follows the last entry written. With that information, you can display whatever portion of the internal trace table that you need. For more information on FLSCTB refer to the *VM/SP Group Control System Command and Macro Reference.*

External Tracing Facilities

You can collect trace data in the CPTRAP spool file for later formatting and viewing. This requires a two-step process:

1. Issuing CPTRAP commands
2. Issuing the ETRACE command.

See the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* and "Debugging with the CPTRAP Facility" on page 100 for more information on the CPTRAP command, see the *VM/SP Group Control System Command and Macro Reference* for more information on the ETRACE command. You have to issue three CPTRAP commands, with a different parameter each time:

1. **CPTRAP 3D**

   The X'3D' parameter tells CPTRAP that it will be collecting "group" entries.

2. **CPTRAP GROUPID** *groupname*

   The GROUPID parameter tells CPTRAP what virtual machine group to get those entries from.

   Or, **CPTRAP ALLOWID** *userid*

   The ALLOWID parameter identifies what particular virtual machine to get the entries from.

CPTRAP START

The START parameter prompts a CP facility, Trace Table Recording, to start putting records in a spool file.[4] But before GCS will pass any records to CPTRAP, you have to enable external tracing with the ETRACE command.

The virtual machine that issues the CPTRAP command must have a Class C userid.[5] Once the CPTRAP commands have been issued, this machine can issue the ETRACE command to commence tracing for its own application or ETRACE GROUP for tracing the entire group (if it's an authorized machine). ETRACE allows you to specify which of the following events should be traced and recorded in the spool file:

- Task dispatches
- External interrupts
- I/O interrupts
- Program interrupts
- SVC interrupts
- I/O requests (SIO and Diagnose)
- IUCV Signal System Service details
- APPC/VM synchronous events
- GETMAIN requests
- FREEMAIN requests
- User trace data generated via GTRACE macro.

Note: Since CPTRAP is a CP function, data in this external trace file may be mixed with trace data from CP, from other GCS machines or machine groups, and from machines not part of any GCS group. You must be selective when deciding what machines and what events to trace with CPTRAP. If you trace too many events, CPTRAP might create spool data faster than you can write it to a DASD, and you might lose data.

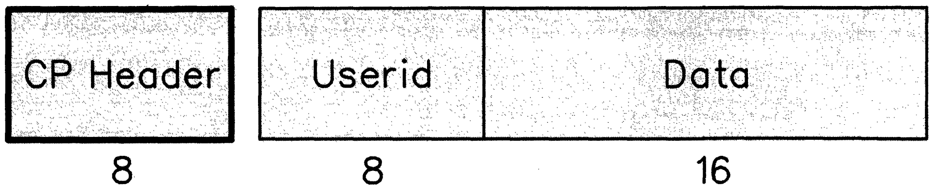
[4] This spool file optionally can be a wraparound file.

[5] Defined in the userid's VM/SP directory entry.

Formatting and Displaying External Trace Records

The external trace file contains two different entries produced by GCS virtual machines:

- An entry for GCS supervisor records:



where:

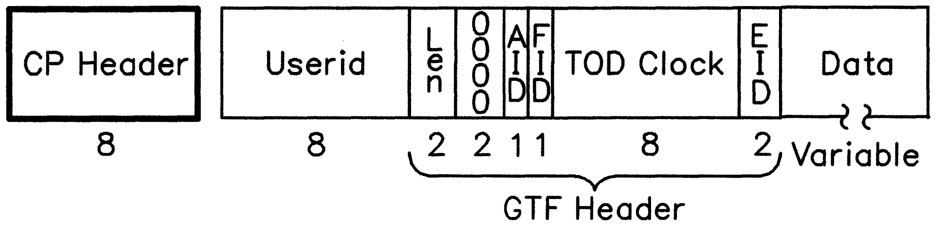
CP header
Contains an eight-byte header appended by CPTRAP when it gets the record.

Userid
Identifies the virtual machine that the entry belongs to.

Data
Contains the data portion of the event's internal trace entry.[6]

[6] Internal trace entry formats begin on page "Formats of Internal Trace Entries" on page 155.

- An entry for GTRACE records:



where:

CP header
: Contains an eight-byte header appended by CPTRAP when it gets the record.

Userid
: Identifies the virtual machine that the entry belongs to.

Len
: Contains the length of the entry, including the GTF header.

0000
: Reserved field of the GTF header.

AID
: Always contains X'FF', indicating that this is a data record.

FID
: (Format ID) identifies the formatting module used for this record.

TOD Clock
: Tells when the record was built, in time-of-day format.

EID
: Contains information from the GTRACE macro's ID parameter.

Data
: Contains the internal trace entry without the internal header (up to 256 bytes).

The main reason you create an external spool file with CPTRAP is to print out or interactively display your trace information. The TRAPRED program[7] lets you do that by providing:

- A routine that directs the formatting of trace entries in your external spool file

[7] TRAPRED is a data reduction program that works on the spool file created by CPTRAP. For more information on TRAPRED, see "Using the TRAPRED Facility" on page 109.

- A routine that prints out your external file or lets you examine it interactively under CMS.

The TRAPRED program handles the formatting of supervisor entries differently from GTRACE entries. The TRAPRED program formats supervisor records and GTF header information. See "Using the TRAPRED Facility" on page 109 for more information on TRAPRED. However, the applications being traced via the GTRACE facility have to supply their own GTRACE formatting modules. If they don't, their trace entries for the data portions of the records simply get printed unformatted, in hexadecimal.

As TRAPRED goes through the spool file, it examines each entry one by one. If the first byte of the CP header (from the format on page 174) is X'3D', then TRAPRED invokes a control program called CSIYTD. For supervisor records, CSIYTD will call a GCS-supplied formatting routine named CSIYTS to format it. However, for GTRACE records, CSIYTD uses GCS-supplied formatting routines to format the GTF header part of the record. CSIYTD also will look for another formatting routine, one supplied by the traced application, to finish the data portion of the record. (It uses the GTRACE record's one-byte FID field to locate this routine. The routine's name must be CSIYTXxx, with 'xx' being the two-digit FID, and it must have a filetype of TEXT.)

If the CSIYTD program cannot find a user-supplied formatting routine, it prints the entry information in hexadecimal. If the program does find a CSIYTXxx TEXT, it calls that routine. At that time, the registers will contain:

R15 The CSIYTXxx routine's entry point

R14 The return address

R13 A 72-byte save area

R1 A parameter list with the following format:

Bytes 0-3	Address of the trace record. (A standard VS1 GTF prefix followed by the trace data.)
Bytes 4-7	Address of a 132-byte output buffer. (Cleared to blanks before the call.) CSIYTXxx puts the formatted trace entry here.
Bytes 8-11	All zeros. (Not used with GCS, but put here to maintain compatibility with VS1. In VS1, this shows the address of GTF options in effect.)
Bytes 12-15	Address of the GTF Event Identifier (EID).
Bytes 16-19	Address of the trace record's data portion.

Bytes 20-23	Address of the end of the trace record's data portion + 1.
Byte 24	Flag for the following options and information: X'40' Format the record for display on a printer. X'80' Format the record for display on a terminal.
Byte 25	Flag byte for trace format processing. X'01' Do not reload the formatting module for the next entry. If this bit is off then the formatting module will be reloaded on each trace entry.
Bytes 26-30	Reserved for future use.
Byte 31	A byte containing the GCS type code.
Bytes 32-63	32 byte work area for use by the called formatting routine.

When formatting the GTRACE entries, user routine CSIYTXxx should fill the output buffer at the address found in bytes 4 through 7 in the input parameter list (the address of this input parameter list is in register 1). Then it should return to the calling routine with one of the following return codes in register 15:

RC	Description
0	The user has printed the buffer and continues processing on the same GTRACE record.
4	The user has printed the buffer and is finished processing the GTRACE record.
8	Do not print the buffer and the GTRACE record is done.
Other	Print the record in hexadecimal.

Examples of Formatted External Trace Table Entries

Here are several sample supervisor event entries, as you would see them in your external trace file.

- Entry type X'02' for an IUCV external interrupt:

SOURCE=IUCV (type X'4000')

```
3D 02 useridxx   VM/GCS EXTERNAL INTERRUPT  SOURCE=IUCV
       IPPATHID, IPFLAGS1, IPTYPE = xxxx xx xx
       OLD PSW = xx x x xxxx x x xxxxxx
```

SOURCE=APPC/VM (type X'4000')

```
3D 02 useridxx   VM/GCS EXTERNAL INTERRUPT  SOURCE=IUCV
       IPPATHID, IPFLAGS1, IPTYPE = xxxx xx xx
       IPCODE, IPWHATRC, IPSENDOP = xxxx xx xx
       OLD PSW = xx x x xxxx x x xxxxxx
```

- Entry type X'03' for an I/O interrupt:

```
3D 03  useridxx   GCS I/O INTERRUPT
        CHANNEL STATUS WORD = x x xxxxxx xx xx xxxx
        OLD PSW = xx x x xxxx x x xxxxxx
```

- Entry type X'05' for an SVC interrupt:

```
3D 05 useridxx   GCS SUPERVISOR CALL INTERRUPT
        SVC CODE = xx
        TASK ID = xxxx
        FUNCTION NAME = xxxxxxxx
        OLD PSW = xx x x xxxx x x xxxxxx
```

- Entry type X'08' External Trace Table Entry by SVC GETMAIN:

```
3D 08 useridxx   VM/GCS GETMAIN VIA SVC
       TASK ID = xxxx
       KEY = xx
       SUBPOOL = xx
       STORAGE ADDRESS = xxxxxxxx
       LENGTH = xxxxxxxx
       ISSUER ADDRESS = xxxxxxxx
```

- Entry type X'09' External Trace Table Entry by SVC FREEMAIN:

```
3D 09 useridxx   VM/GCS FREEMAIN VIA SVC
       TASK ID = xxxx
       KEY = xx
       SUBPOOL = xx
       STORAGE ADDRESS = xxxxxxxx
       LENGTH = xxxxxxxx
       ISSUER ADDRESS = xxxxxxxx
```

- Entry type X'0A' External Trace Entry by Branch Entry GETMAIN:

```
3D 0A useridxx   VM/GCS GETMAIN VIA BRANCH ENTRY
       TASK ID = xxxx
       KEY = xx
       SUBPOOL = xx
       STORAGE ADDRESS = xxxxxxxx
       LENGTH = xxxxxxxx
       ISSUER ADDRESS = xxxxxxxx
```

- Entry type X'0B' External Trace Entry by Branch Entry FREEMAIN:

```
3D 0B useridxx   VM/GCS FREEMAIN VIA BRANCH ENTRY
       TASK ID = xxxx
       KEY = xx
       SUBPOOL = xx
       STORAGE ADDRESS = xxxxxxxx
       LENGTH = xxxxxxxx
       ISSUER ADDRESS = xxxxxxxx
```

- Entry type X'0C' External Trace Entry by APPC/VM Synchronous Event:

```
3D 0C useridxx   VM/GCS APPC/VM SYNCHRONOUS EVENT
       IPPATHID, IPFLAGS1, IPTYPE = xxxx xx xx
       IPCODE, IPWHATRC, IPSENDOP = xxxx xx xx
```

Here is a sample GTRACE entry as you would see it in your external trace file.

- Entry type X'0E' for a GTRACE entry:

```
3D 0E useridxx   GCS USER REQUESTED GTRACE
        TIME OF DAY CLOCK = xxxxxxxxxxxxxxxx
        LENGTH OF GTF HEADER AND TRACE DATA = xxxx
        FORMAT ROUTINE ID = xx
        EVENT IDENTIFICATION = xxxx
       [formatted GTRACE data appears here. . . .]
```

Dumping Facilities

The Common Dump Receiver

To let you dump out the contents of virtual storage and see where problems have occurred, GCS must provide a way around its own safeguard mechanisms. Otherwise, your GCS dumps would be largely incomplete.

Rules of Authorization

If a dump is directed to an authorized user all of the requested storage will be dumped including the discontiguous shared segments. If the dump is directed to an unauthorized user only the storage with a key of 14 and non-fetch protected storage will be dumped.

If you direct the dump to yourself or another unauthorized userid, you cannot dump any fetch-protected areas or storage with a key other than 14. Unauthorized dump receivers can accept only key 14 and other non-fetch-protected storage.

You solve this problem by singling out one authorized virtual machine as your common dump receiver. At build time, when creating your GCS configuration file, you will be prompted to name this common dump receiver. Choose any authorized userid, perhaps the same userid that you specify as your recovery machine. But make sure you list it on the GROUP EXEC's screen of authorized GCS userids. If you name a common dump receiver, GCS's dump functions (listed in "How To Initiate Dumps") automatically will send their output to it.[8]

How To Initiate Dumps

There are four different functions for requesting GCS dumps:[9]

[8] Except for GDUMP, which optionally lets you choose another receiver.

[9] If you set it up beforehand, CP's VMSAVE function will also create copies of selected virtual machines either when CP itself terminates or when CP terminates them.

Macros applications can issue

ABEND (with DUMP operand) This dumps the entire virtual machine as well as any discontiguous shared segments (shared segments linked to your GCS system, but not within the bounds of your virtual machine). The dump automatically goes to the common dump receiver, if you have one, rather than the machine that issued ABEND.

SDUMP This can dump all or part of the virtual machine. If you don't have a common dump receiver, the data goes to the machine that issued it, according to the rules of authorization in "Rules of Authorization" on page 180.

Commands you issue

GDUMP This can dump all or part of a virtual machine's storage. You can send the dump to the common dump receiver, or the machine you issue GDUMP from, or another userid you chose. Whatever receiver you choose, the rules of authorization in "Rules of Authorization" on page 180 apply.

SYSTEM RESTART (a #CP command) This dumps all of a virtual machine's storage, plus any discontiguous shared segments, when you cannot use the GDUMP command. (Example: you have a GCS disabled loop and issue #CP SYSTEM RESTART). If you have no common dump receiver, the issuing machine gets the dump, according to the rules of authorization in "Rules of Authorization" on page 180.

To produce a dump requested by one of these functions, GCS calls CP and requests a dump. While it performs the dump, CP continues dispatching other machines in the virtual machine group. This poses a problem if those members go on to change common storage as it is being dumped.

To preserve common storage contents until the dump finishes, the GCS supervisor acquires the common storage lock. This prevents other machines from acquiring the lock during the dump. If all authorized machines test the common lock before trying to change common storage, they will be effectively suspended until the dump finishes. The only common storage that might change is that obtained by other machines before the dump began.

Note: The common storage lock gets set "on" only if your common dump receiver is an authorized GCS userid and you are using the SDUMP and GDUMP functions.

It is possible to receive two dumps. An example of this would be if a user ran out of storage while producing a dump. One dump would be produced as the user dump and the second dump would be the supervisor dump.

Interactive Debugging Support

Authorized CP Commands

Authorized userids can have access to six CP debugging commands:

- BEGIN
- PER
- DISPLAY
- STORE
- DUMP
- VMDUMP.

Initially, these are Class G commands, available to all userids. You may want to reclassify these commands to prevent unauthorized users from altering storage that may effect other members of the GCS group.

In addition, you should make two related CP commands, ADSTOP and TRACE, totally unavailable to both authorized and unauthorized GCS userids. For more information on controlling access to CP commands, see *VM/SP Planning Guide and Reference* or *VM/SP HPO Planning Guide and Reference.*

Analyzing Dumps

Once storage has been dumped, it can be:

- Read in and analyzed by the receiving virtual machine under CMS with the Interactive Problem Control System (IPCS), component of VM/SP Release 4

- Dumped to tape (using Spool-to-Tape) and sent to a Field Engineering service team for analysis.

IPCS has some specialized routines on the IPCSDUMP and MAP commands for processing GCS dumps.[10]

To use the GCS IPCS support necessary to process a virtual machine dump the appropriate IPCS minidisks (the minidisks containing GCS and IPCS support) must be accessed before processing the dump.

[10] For a discussion of what IPCS does and how to use it, see Chapter 7, "Debugging Using IPCS" on page 197.

A map compressing routine — compresses the GCS nucleus load map into a format IPCS can use with the IPCS MAP command. IPCS MAP will look for a nucleus load map with the default name GCSNUC MAP *. Once it's compressed, the default name is GCSIPCS MAP.

A data extraction routine — gathers information that IPCS can use when building its problem report, like:

- Reason for the dump (GDUMP, SDUMP, SYSTEM RESTART, ABEND, program check, or "other")
- Failure location (in case of task termination or machine termination)
- Appropriate keywords (APPLADDR, CKD, COMPID, DISP, ENTRY, FAILURE, MAINTLVL, MODULE, SCPLVL, or TASKID)
- Textual data.

If this extraction routine discovers a dump in RSCSV2 format, it invokes an RSCS extraction routine to find additional problem report information.

An expanded DMMTAB communication table — includes a dump type for GCS, like the ones for CP and CMS.

A way to print formatted VTAM or VSCS control blocks — adds an option to IPCS's PRTDUMP command that lets you specify whether you want formatted VTAM or VSCS control blocks printed in a dump. You'll have this option for any GCS- or VMDUMP-generated dump of type GCS or RSCSV2. First you will receive a prompt asking you if you want your dump printed using the VTAM option. If you do not pick the VTAM option you will receive a prompt asking you if you want your dump printed using the VSCS option. If you do not choose either of these options your dumps will be printed unformatted.

Subcommands for the DUMPSCAN command

For more information on DUMPSCAN subcommands see Appendix A, "Using DUMPSCAN Subcommands" on page 259.

Dumping VSAM Information

When VSAM detects certain internal logic errors, it produces a special dump, called an IDUMP, that can help you identify those problems. To look at information in the dump header, use the IPCS DUMPSCAN DUMPID subcommand. This dump header will contain the following information:

VSAM IDUMP	24—character symptom string	MM/DD/YY	HH:MM:SS	SAVEAREA ADDR

VSAM IDUMP
Is a dump identification message.

24-character symptom string
Identifies error codes, the location of the error, and the module that detected the error. For information on how to interpret this character string, see the *VSE/VSAM Programmer's Reference.*

MM/DD/YY
Shows the date when VSAM detected the error.

HH:MM:SS
Shows the time of day when VSAM detected the error.

SAVEAREA ADDR
Contains the address of the save area that shows what each register contained when VSAM discovered the error. Ignore the first 16 bytes of this save area, and look for the register contents beginning at the 17th byte. You'll find the contents of all 16 registers in the following order: registers 9-15, registers 0-8.

Chapter 6. Debugging TSAF

Debugging TSAF

The three ways that you can collect error information for problem diagnosis within Transparent Services Access Facility (TSAF) are described in this chapter. They are:

- Using console logs, described in "Using the Console Log" on page 187
- Using dumps, described in "Using TSAF Dumps to Diagnose Problems" on page 188
- Using system trace data, described in "Using System Trace Data to Diagnose Problems" on page 191

In addition, "Interactive Service Queries" on page 194 describes how the TSAF QUERY command can also provide you with problem diagnosis information.

Note: The TSAF operator does not necessarily diagnose problems, especially from the TSAF virtual machine. Dumps and system trace data are usually used by the system programmer or whoever is responsible for diagnosing system problems.

Summary of Steps to Follow When a TSAF Abend Occurs

When a TSAF abend occurs, you must do the following steps:

1. Collect information about the error.

   - Save the console sheet or spooled console output from the TSAF virtual machine.
   - Save and process any dumps that TSAF produces.

     When an abend occurs in TSAF, either because TSAF issued an abend or because a TSAF or CMS operation caused a program exception, TSAF produces a dump via the CP VMDUMP command (described in the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference*). CP sends the dump to TSAF's virtual reader.
   - Save any CPTRAP file that contains TSAF data.

2. Collect other types of information about system status, such as:

   - Status of real and virtual devices that TSAF is using
   - System load at the time of the failure on any systems using TSAF and the status of each system (for example, did another system abend?)
   - Types of applications that are using TSAF at the time and any information about them

- Physical connection configuration of the systems in use.

3. Recover from the abend to continue processing.

   After TSAF creates a dump, TSAF then issues a CP SYSTEM RESET command. If the CONCEAL option is on, as recommended, CP automatically IPLs CMS. Otherwise, you, the operator, must re-IPL CMS. Similarly, if TSAF is not invoked from the PROFILE EXEC, you must restart the TSAF virtual machine.

VM/SP System Messages and Codes or *VM/SP HPO System Messages and Codes* lists the TSAF abend codes and their causes.

Using the Console Log

TSAF provides informational messages, as well as error messages, that may help you with problem determination. To keep track of the console messages, enter:

SPOOL CONSOLE START TO *userid*

where *userid* can be the userid of the TSAF virtual machine or another virtual machine userid to whom you want TSAF to send the console log. You may want to add this to TSAF's PROFILE EXEC so a console log is always created.

To close the console log, enter:

SPOOL CONSOLE CLOSE

The log of messages received is sent to the specified userid. See the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* for more information on the SPOOL command.

TSAF provides additional information at the time of an abend to help you diagnose the problem. The console log contains information about the abend, such as:

- abend code
- program old PSW
- contents of the general purpose registers.

TSAF also attempts to determine the displacement of the module in which the abend occurred and the displacement of the calling module.

Figure 13 shows some of the messages that TSAF may issue in response to an abend condition:

```
ATSCAC999T TSAF system error
ATSCAB017I Abend code ATS999 at 022730
ATSCAB018I Program old PSW is FFE002FF 40022730
GPR0-7 00022FFC 000003E7 00022FDA 00052BC0 00208080 00020C58 0033E811 00000001
GPR8-F 7F3B78AF 603C0000 00020B64 00022D6F 50021D70 00022B48 40022718 00023FB0
ATSCAB019I Abend modifier is ATSCAC
ATSCAB021I Failure at offset 0A06 in module ATSCAC dated 86.020
ATSCAB022I Called from offset 04B4 in module ATSSCN dated 86.078
ATSCAB023I VMDUMP ATSCAB*ATSCAB1 05/28/86 16:02:06 taken
```

Figure 13. Sample TSAF Console Log

Using TSAF Dumps to Diagnose Problems

You can use IPCS to collect and diagnose problem data for the TSAF virtual machine. The console listing, as described in "Using the Console Log" on page 187, may help you diagnose problems without using dumps.

The steps involved in using dumps to diagnose problems are:

1. Create a TSAF IPCS map, if it does not already exist.
2. Create the TSAF dump.
3. Process the TSAF dump.
4. Diagnose the TSAF dump.
5. Print the TSAF dump.

Creating the TSAF IPCS Map

Note: You only need to do this step when a new CMS or TSAF nucleus is built.

When a new CMS or TSAF nucleus is built, enter the following IPCS command to compress the TSAF load map for IPCS:

MAP TSAF

The default name for the map source file is TSAF MAP; and the default name for the input CMS nucleus load map is CMSNUC MAP. The default name for the compressed map file is TSAFIPCS MAP, which you create using MAP TSAF.

Note: If you do not have the compressed map file, IPCS facilities, which allow for diagnosis with dumps, are greatly reduced. For instance, without the map you could not invoke the formatted display subcommand (FDISPLAY) and you would not receive any TSAF control block information.

Creating a TSAF Dump

The TSAF virtual machine creates its own dumps. The dump goes to the reader of the TSAF virtual machine. Because the TSAF virtual machine is not set up to process dumps, you need to transfer the dump file to the appropriate virtual machine.

If the TSAF virtual machine cannot create the dump, you can use the VMDUMP command. The VMDUMP command dumps virtual storage that VM/SP or VM/SP HPO creates for the virtual machine user; in this case, for TSAF. The dump goes to the virtual machine specified by the SYSDUMP parameter on the SYSOPR macro in the DMKSYS ASSEMBLE file, if you enter the following CP command:

```
VMDUMP 0-END SYSTEM FORMAT TSAF
```

Do not use the reserved names of ATSCAB1 or ATSCAB2 for the dump id of VMDUMP. The *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* has more information about the VMDUMP command.

Processing a TSAF Dump

After the TSAF virtual machine creates a dump, load the dump onto disk. To load the dump, enter the following IPCS command:

```
IPCSDUMP
```

The default map file is TSAFIPCS MAP.

When you issue IPCSDUMP, it invokes a TSAF routine to extract information from the dump and transmit it to IPCS for inclusion in the problem report and/or symptom summary. IPCSDUMP creates the following:

- Problem report
- Symptom summary
- Disk resident dump to which IPCS appends the map information.

See the "IPCSDUMP Command" on page 233 for more information about the IPCSDUMP command.

Diagnosing a TSAF Dump

The IPCSDUMP command generates a symptom record, which is based on problem report information. The symptom record helps you find out why TSAF created the dump. The symptom record includes:

- Information about the system environment at the time of the dump

- The symptom string that contains the following component-related symptoms:
  - Error code
  - ID of the failing component
  - ID of the failing module
  - Registers and PSW contents.

You can also use the DUMPSCAN command to examine the dump interactively. The DUMPSCAN command is described in "DUMPSCAN Command" on page 230. The following sections introduce those subcommands specifically for TSAF (FDISPLAY and TRACE).

Note: TRACE can also be used for CP dumps.

Displaying the TSAF Dump Information

The FDISPLAY subcommand of the IPCS DUMPSCAN command displays data control blocks, tables, and arrays important to the TSAF virtual machine. You can get information about the following by invoking different FDISPLAY parameters.

- Path array (PATH)
- Service table (SERVICE)
- Collection control block (COLLECT)
- Resource table (RESOURCE)
- Neighbor table (NEIGHBOR)
- Routing array (ROUTING)
- Link definition array (LINKDEF)
- Link control blocks (LINKCTL BSC or LINKCTL CTCA).

See "FDISPLAY Subcommand" on page 285 for a complete listing of the FDISPLAY parameters and for some example outputs of the FDISPLAY subcommand.

Formatting and Displaying Trace Records

TSAF maintains an internal trace table within the TSAF virtual machine. You can use the TRACE subcommand of IPCS DUMPSCAN to format and display trace records from the TSAF internal trace table. By using the HEX or FORMAT parameters, you can display the trace table entries in a hexadecimal display or a formatted display. See "TRACE Subcommand" on page 324 for examples of using the TRACE subcommand and the sample outputs.

You can also scroll through the formatted or hexadecimal output with either of the following IPCS DUMPSCAN subcommands:

- TRACE SCROLL or SCROLLU
- SCROLL or SCROLLU.

See "TRACE Subcommand" on page 324 and "SCROLL Subcommand" on page 314 for more information about the DUMPSCAN TRACE and SCROLL subcommands.

Printing a TSAF Dump

The IPCS PRTDUMP command prints the dump and symptom record that IPCSDUMP processed. The output you get consists of the following:

- Symptom record
- Dump in hexadecimal (no special formatting)
- Appended load maps
- Contents of the registers and the PSW.

See "PRTDUMP Command" on page 251 for more information on the IPCS PRTDUMP command.

Using System Trace Data to Diagnose Problems

TSAF maintains an internal trace table within the TSAF virtual machine. You can use the IPCS DUMPSCAN TRACE subcommand to display the internal trace table entries. TSAF also writes trace entries to the system CPTRAP file. You can then use TRAPRED to view TSAF entries.

Setting External Tracing

The TSAF SET ETRACE command lets you enable or disable external tracing for the TSAF virtual machine. If you want to collect TSAF trace records, issue the following from the TSAF virtual machine before CPTRAP is started:

SET ETRACE ON

When you set external tracing on, certain internal TSAF trace records are written externally to a CPTRAP spool file. A complete description of the SET ETRACE command is in the *VM/SP Transparent Services Access Facility Reference.*

Using CPTRAP to Trap Trace Table Entries

The CPTRAP command collects TSAF information in a reader file. This information helps with problem determination.

Note: Because the TSAF virtual machine is not set up to diagnose problems, the virtual machine that has the authority to issue the CPTRAP command must do so.

The following commands activate CPTRAP for TSAF records only:

1. **CPTRAP ALLOWID** *userid*
   (*userid* is the TSAF virtual machine userid.)

2. **CPTRAP 3E**
   This activates CPTRAP for 3E entries that the TSAF virtual machine produces.

3. **CPTRAP START**

Enter:

CPTRAP STOP

to end CPTRAP processing. When you issue this command, the CPTRAP SPOOL file goes to your reader.

For more specific information about the CPTRAP command, see the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference* and "Debugging with the CPTRAP Facility" on page 100.

Getting Information about CPTRAP with QUERY

The privilege class C, QUERY CPTRAP command gets information about the CPTRAP. For example:

- QUERY CPTRAP STATUS tells you if CPTRAP is started and by whom.

- QUERY CPTRAP SELECT returns a table that shows the current selectivity (on, off, or extra) for each type of CPTRAP record.

For example, if you enter the following sequence of commands:

1. **CPTRAP 5**
2. **CPTRAP C VMBLOK FF3AE8**
3. **CPTRAP C DEVADDR E**
4. **CPTRAP 6 OFF**
5. **QUERY CPTRAP SELECT 05 06 0C**

you will receive the following response:

```
05:  ON
06:  OFF
0C:  DEVADDR 000E, VMBLOK FF3AE8
```

Entering CPTRAP STOP ends CPTRAP processing. For more specific information about the class C QUERY command, see the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference*.

Viewing CPTRAP Data with TRAPRED

To access the CPTRAP reader file and review the entries contained in that file, enter the following:

TRAPRED *filenum*

where *filenum* is the number of the CPTRAP reader file. The filename and filetype are 'CPTRAP FILE'.

Then when TRAPRED prompts for selectivity, enter:

1. **3E MACHTYPE TSAF**

   This specifies you want to collect specific machine type entries for TSAF of record type 3E.

2. One of the following:

   - **HEX**
     to cause an entry to be displayed in hexadecimal form

   - **FORMAT**
     to cause an entry to be displayed in formatted form.

3. One of the following:

   - **TYPE** *n*
     to display entries starting with the current line and moving toward the end of the file

   - **TYPEBACK** *n*
     to display entries starting with the current line and moving toward the top of the file

   where *n* is the number of entries you want displayed.

The output for TRAPRED has the same format as the internal trace entries, with one exception; DUMPSCAN displays the internal trace entries with an address. Because CPTRAP entries do not have an address, the identifying string is simply "TSAF". For more information about the TRAPRED command, see "Using the TRAPRED Facility" on page 109.

Trace Table Entry Format for TSAF

The trace table entries vary in length and follow the format described below. The length fields are one-byte long and may be any number from 0 to 255. The length and data fields are optional data fields.

A trace table entry looks like the following:

length(1)	data(1)	. . .	length(n)	data(n)	TRAILER RECORD

Figure 14. TSAF Trace Table Entry

The trailer record format looks like the following:

Clock (STCK format)	Characters 4 through 6 of module name	Trace id code	Data area length	'E00E'x

Figure 15. TSAF Trace Table Trailer Record

The lengths associated with each field are:

- Clock (STCK format) - 8 bytes
- Characters 4 through 6 of module name - 3 bytes
- Trace id code - 2 bytes
- Data area length - 2 bytes
- 'E00E'x - 2 bytes.

Note: Module entries and module exits do not have length fields associated with each data field. Module entries and exits do, however, have the data area length in the trailer record.

Module entry trace records appear only in the internal trace table. TSAF identifies these records by setting bit 15 of the trace identifier code to 1. The data for a module entry is in the parameter list used during the module call.

Module exit trace records also appear only in the internal trace table. TSAF identifies these records by setting bit 14 of the trace identifier code to 1. The data for a module exit is in registers 14 and 15 at the time of the module exit.

Interactive Service Queries

The TSAF QUERY command, issued from the TSAF virtual machine, can give you more information to help you diagnose problems. The TSAF QUERY command gives you data about the TSAF configuration when the TSAF virtual machine is running:

- QUERY COLLECT displays the processor names that are currently in the TSAF collection.
- QUERY ETRACE displays the current setting of the external tracing.

- QUERY LINK displays information about the links that TSAF currently has.

- QUERY RESOURCE displays the current list of global resources in the collection.

See the *VM/SP Transparent Services Access Facility Reference* for more specific information about the TSAF QUERY command.

Chapter 7. Debugging Using IPCS

Debugging Using IPCS

What IPCS Does

The VM/SP Interactive Problem Control System (VM/SP IPCS) provides VM/SP or VM/SP HPO installations with an interactive, on-line facility for reporting and diagnosing software failures and for managing problem information and status. VM/SP IPCS can perform these functions for:

- CP ABEND dumps
- CMS dumps
- GCS dumps
- TSAF dumps
- PVM dumps
- RSCS dumps
- Any other dumps created by the CP command VMDUMP, or Stand-Alone Dump, or CP DIAGNOSE X'94'.

For more information on Stand-Alone Dumps see "Stand-Alone Dump Facility" on page 128. By using VMDUMP, the operator of a Guest Virtual Machine (GVM) can create a dump of any user-detected software problem and make it available to IPCS.

Note: IPCS and the component related format routines use the information (for example, control blocks and pointers), contained in the storage actually dumped to produce the output. Therefore, if the information has been corrupted either as a direct result of the error, or during the process of preparing for and taking the dump, unpredictable results may occur. When using IPCS, one must be aware that the various routines may not be able to extract the information for which they were invoked. This is not an error in the routines, but a direct result of the corrupted storage.

Major Functions

VM/SP IPCS provides facilities to assist in these major functions (as pictured in Figure 16 on page 200):

- Reporting Problems

  Facilities for reporting problems standardize the reporting process, identify previous occurrences of the same problem on the system, and should allow faster and more specific identification of similar problems previously experienced by the VM/SP customer base.

  Recognition of duplicate problems should:

  - Reduce the amount and expense of unnecessary hard-copy documentation

  - Allow faster identification of available fixes that can be applied to the system.

- Diagnosing Problems

  Facilities for diagnosing problems let you interactively view disk-resident problem data (i.e., CP ABEND dumps and all other dumps processed by the IPCSDUMP command). This capability lets the customer or Program Support Representative (PSR) interactively diagnose a dump-related problem from any VM/SP supported terminal, without the need for hard-copy problem data. A Symptom Record (SR) included in the dump helps you be more self-sufficient in determining problems and in identifying the sources of problems.

- Managing Problems and Data

  Facilities for managing problems and data let you:

  - Update the individual disk-resident problem reports and summary status reports
  - Display and print problem reports and status reports
  - Print a hard-copy Authorized Program Analysis Report (APAR) and off-load problem data.

  (An APAR is used to report program problems to IBM personnel responsible for resolving the problems.)

  The symptom record is a depository for Structured Data Base (SDB) keywords which are used in searches for duplicate problems. Thus, the symptom record enhances the APAR processing function.

  These facilities let the customer or PSR track and manage problems from their first occurrence through their resolutions.

Figure 16 on page 200 and Figure 17 on page 201 illustrate the IPCS functions and relate these functions to the IPCS commands. These functions will be expanded upon in the next few pages.

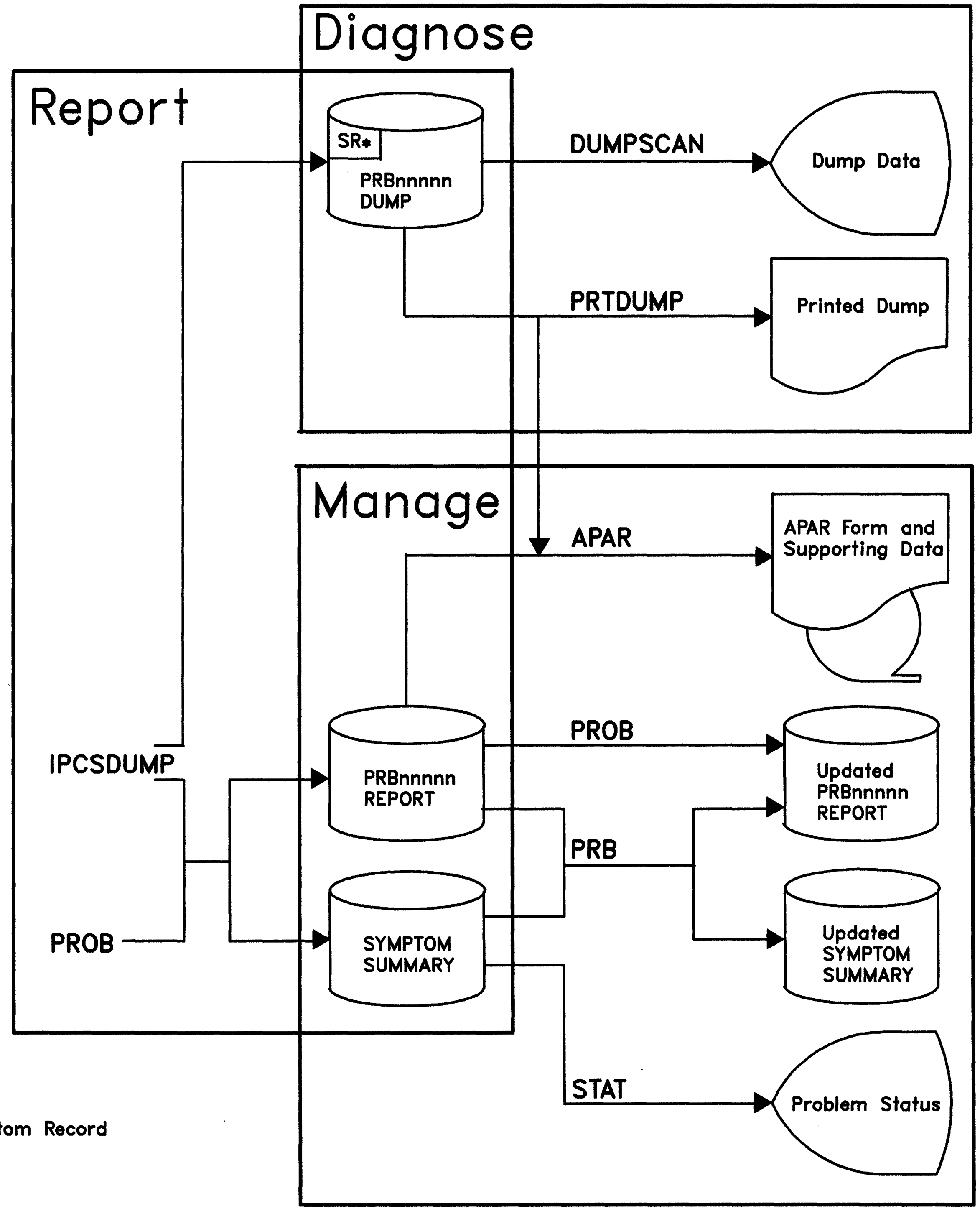


Figure 16. Functional Overview

Data Flow

Figure 17 shows the overall data flow and relationship between the various IPCS files and commands.

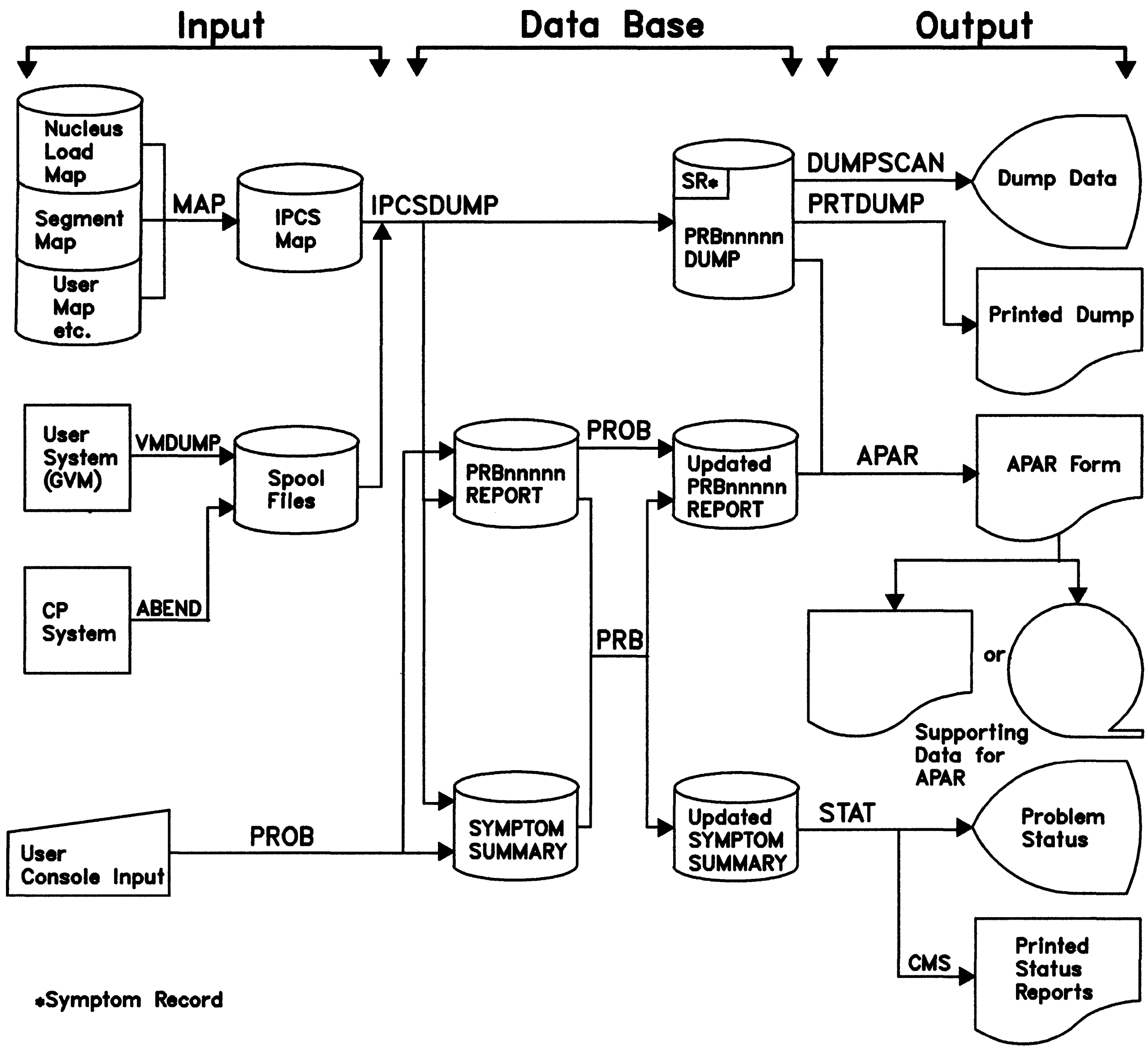


Figure 17. Data Flow

Problem Reporting (IPCSDUMP, PROB)

IPCSDUMP and PROB generate a disk-resident problem report for a software problem. The report contains specific symptoms of the problem as well as identification information (time and date stamps of problem occurrence and creation, problem severity, CPU type and CPU serial). Optionally, the report can contain free-form descriptive information and identification of CMS files containing data related to the problem.

IPCSDUMP provides the appropriate problem symptoms through automatic dump data extraction, user prompting, or a combination of both. When no dump is available, PROB prompts you for the data required for the problem report. IPCS also creates a symptom record as the first physical record of the dump.

IPCSDUMP and PROB automatically compare symptoms of the current problem with those of problems previously processed by IPCS at the installation and notify you of possible duplicates.

Problem Diagnosis (DUMPSCAN, PRTDUMP)

DUMPSCAN provides a variety of dump-viewing subcommands which allow you to interactively locate, display, and/or print data, or control the display screen. This is possible for all CP ABEND dumps, Stand-Alone Dumps, TSAF dumps, and those taken through the VMDUMP facility or CP DIAGNOSE X'94' or facilities provided by GCS (GDUMP), and processed by the IPCSDUMP command.

PRTDUMP provides a printing and formatting facility which allows you to obtain a hard-copy listing from a dump.

Problem and Data Management (PRB, STAT, APAR)

PRB provides facilities for updating the status of open problems from initial reporting through closing, and for viewing problem reports from a display.

STAT provides facilities for displaying or printing the status of any, all, or combinations of problems on file at the installation.

APAR lets you print a hard-copy APAR and off-load problem data to tape, or print hard copy for later submission of the problem to IBM. Only problems arising from IBM system control programs, licensed programs, and their respective documentation are covered by the APAR process. (For information about the type of problems for which APARs may be submitted and the preparation and submission of APARs, see *IBM Field Engineering Programming System: General Information.*)

Symptom Record

The symptom record is a collection of software-error related information formatted in Structured Data Base (SDB) keywords. This format aids the National Service Division in searching for duplicate problems for APAR processing.

The symptom record is located within the first 2K-byte record inserted into the first physical record of the dump file. The symptom record consists of these required sections:

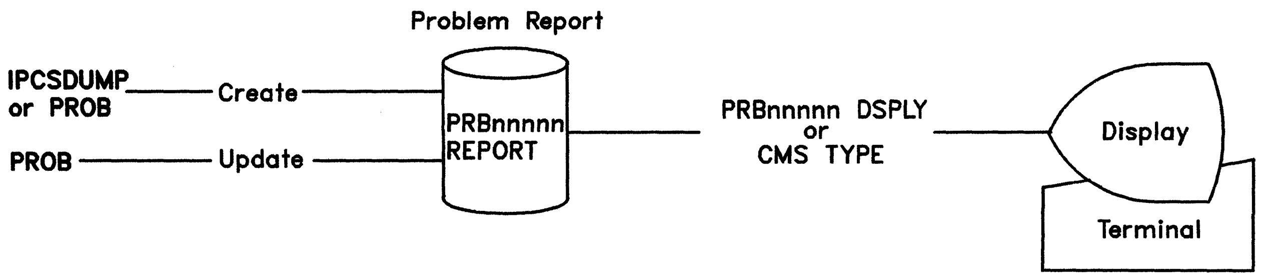
- The environment information describing the system environment at the time of the dump:
  - Component/Release/Service level of the system
  - CPU model and serial number
  - Date and time stamp
  - Type of dump.
- The symptom string containing the following component-related symptoms:
  - Error code
  - ID of the failing component
  - ID of the failing module
  - Register/PSW difference.

You can format and display a symptom record by using the DUMPSCAN subcommand SYMP. A symptom record can be printed by using either the DUMPSCAN subcommand PRINT or the PRTDUMP command. If PRTDUMP is used, the symptom record will be printed on the first page of the dump.

IPCS Files

The following are the filenames, filetypes, and descriptions of the data files used by IPCS. All files associated with a given problem (for example, Program Temporary Fix (PTF) files or supplementary files) are of the form PRBnnnnn filetype, where PRBnnnnn is the filename (nnnnn is the problem number).

PRBnnnnn REPORT
is generated by either the PROB command or the IPCSDUMP command. One file exists for each problem known to the system. You may display these files by using the CMS TYPE or the PRB DSPLY command. Figure 18 on page 205 is a sample problem report with an explanation of its fields.

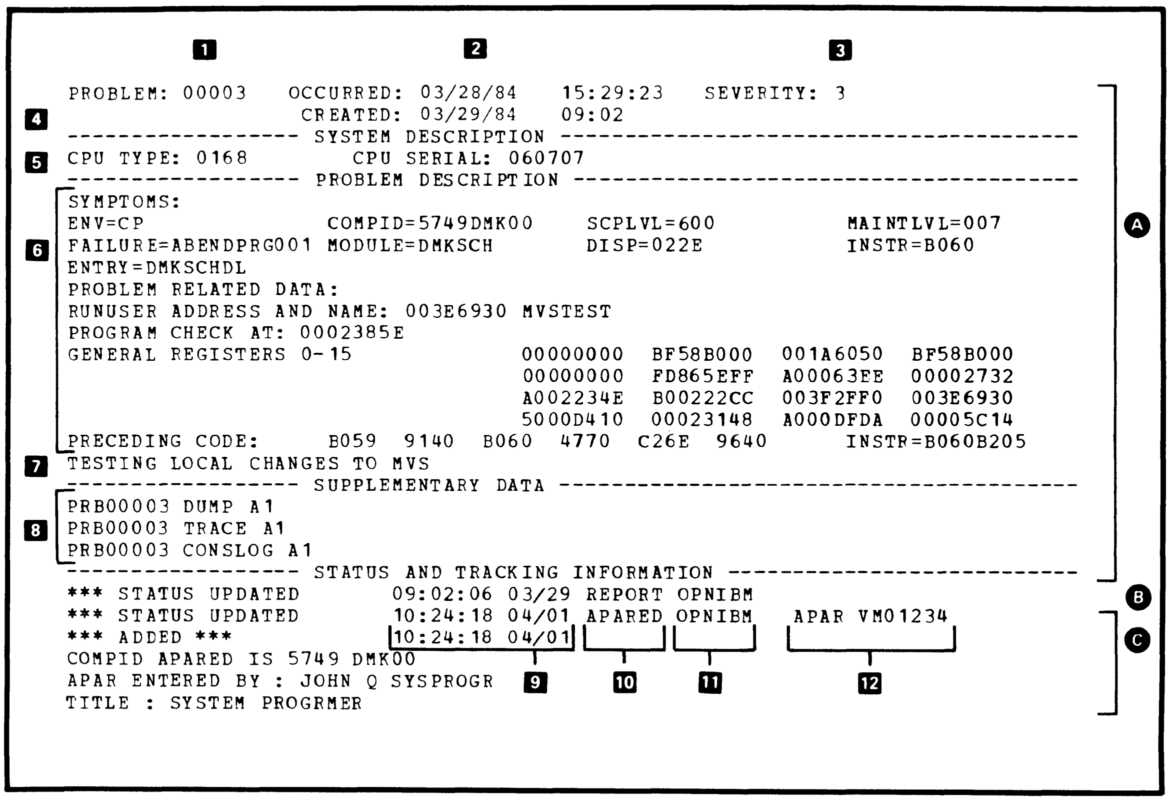


SYMPTOM SUMMARY
contains a control record for each problem. You may use the STAT command to display this record. The symptom summary file serves these functions:

- It provides information about all the problems known to the system and can, therefore, be used for problem control.

- It keeps the symptoms of each problem with their summary control record. These symptoms are used to screen out possible duplicate problems as they are entered into the system via the PROB command or the IPCSDUMP command.

Note: Do not edit the symptom summary file because it contains binary data not visible in a normal editing session. Alteration of this data could cause the file to become unusable by IPCS.



```
PROBLEM: 00003    OCCURRED: 03/28/84    15:29:23    SEVERITY: 3
                   CREATED: 03/29/84    09:02
------------------ SYSTEM DESCRIPTION ----------------------------------------
CPU TYPE: 0168         CPU SERIAL: 060707
------------------ PROBLEM DESCRIPTION ---------------------------------------
SYMPTOMS:
ENV=CP              COMPID=5749DMK00     SCPLVL=600             MAINTLVL=007
FAILURE=ABENDPRG001 MODULE=DMKSCH        DISP=022E              INSTR=B060
ENTRY=DMKSCHDL
PROBLEM RELATED DATA:
RUNUSER ADDRESS AND NAME: 003E6930 MVSTEST
PROGRAM CHECK AT: 0002385E
GENERAL REGISTERS 0-15                00000000  BF58B000  001A6050  BF58B000
                                      00000000  FD865EFF  A00063EE  00002732
                                      A002234E  B00222CC  003F2FF0  003E6930
                                      5000D410  00023148  A000DFDA  00005C14
PRECEDING CODE:     B059  9140  B060  4770  C26E  9640      INSTR=B060B205
TESTING LOCAL CHANGES TO MVS
------------------ SUPPLEMENTARY DATA ----------------------------------------
PRB00003 DUMP A1
PRB00003 TRACE A1
PRB00003 CONSLOG A1
------------------ STATUS AND TRACKING INFORMATION ---------------------------
*** STATUS UPDATED        09:02:06 03/29 REPORT OPNIBM
*** STATUS UPDATED        10:24:18 04/01 APARED OPNIBM    APAR VM01234
*** ADDED ***             10:24:18 04/01
COMPID APARED IS 5749 DMK00
APAR ENTERED BY : JOHN Q SYSPROGR
TITLE : SYSTEM PROGRMER
```

A Data in original problem report created by IPCSDUMP or PROB.

B Data added by execution of the PRB command.

C Data added by execution of the APAR command.

1 Number assigned to the problem.

2 Date on which the problem was initially detected.

3 Severity level of the problem.

4 Date and time the problem report file was generated.

5 Machine on which the problem occurred.

6 Data recorded with the problem.

7 User entered description of the problem.

8 Other pertinent data files that can be used to help solve the problem.

9 Time and date when user updated the problem status via the PRB command.

10 Last function executed by the PRB command.

11 New status after completion of the execution of the PRB command.

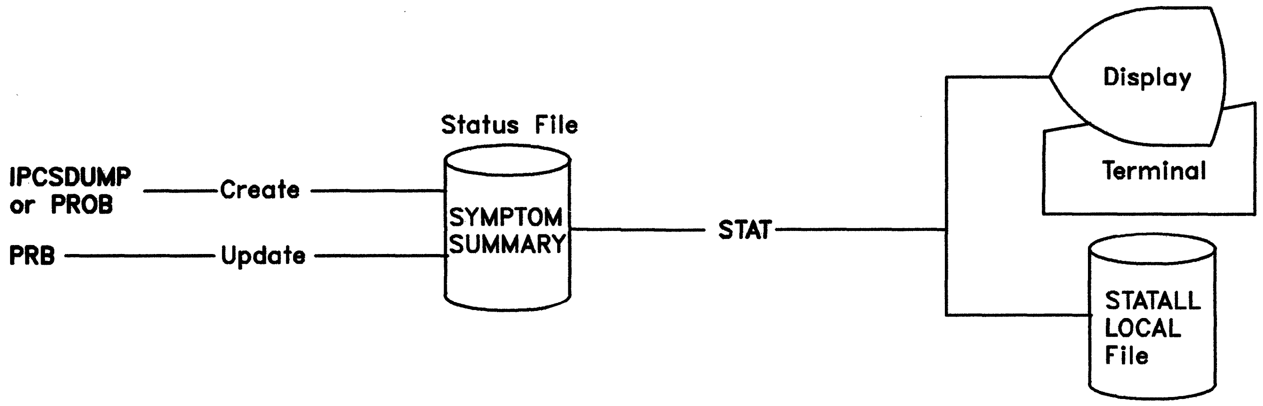
12 Additional data and/or information dependent on status update:

where:

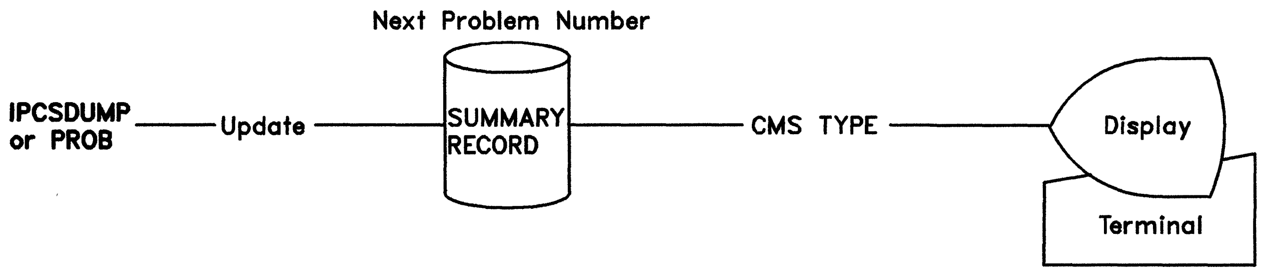
If FUNCTN is:	Then:
APARED	APAR nnnnnnn is the APAR number.
PTFRCV or PTFON	fn ft is the filename of the PTF.
If STATUS is:	**Then:**
CLOSED	If the problem was not closed by application of a PTF/PLC, the closing code is listed. This may be one of the following: DOCUMENTATION, HARDWARE, UNREPRODUCIBLE, USER, DUP OF mmmmmmm (where mmmmmmm is a previous problem number or APAR number.)
OPNUSR or OPNIBM	DUP OF nnnnn is the duplicate problem number.

Figure 18. Sample PRB00003 REPORT File with Status Updates Added

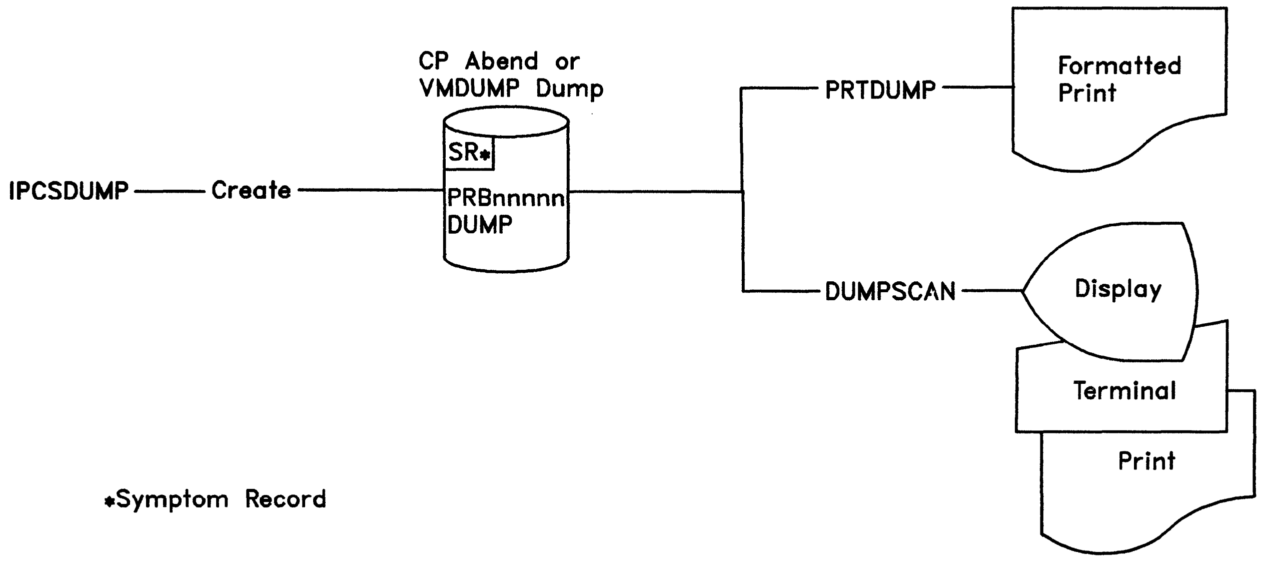
STATALL LOCAL
contains the status of all problems known to the system. It is created by the STAT ALL command. You may print this file for a summary of all known problems.



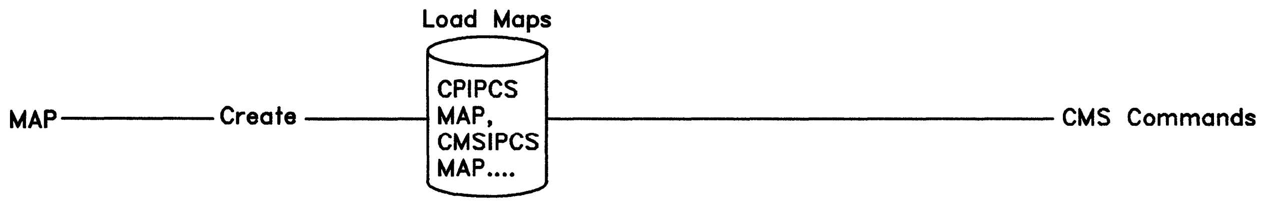
SUMMARY RECORD
consists of one 80-character record that contains the five digits of the next available problem number. This allows you to see what the next problem number is so that you can control information. If necessary, you may use the System Product editor, invoked by the XEDIT command, to create or change this file. You can display this file (using the TYPE command) to find the next problem number. This number allows you to name the files associated with a problem (PRBnnnnn). All data associated with a given problem should be controlled in this manner.



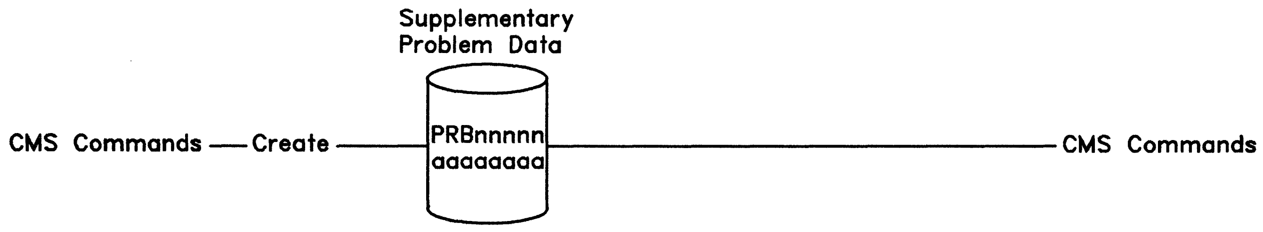
PRBnnnnn DUMP
contains the output of IPCSDUMP, where nnnnn is the problem number assigned by IPCSDUMP. The first record of the dump is the symptom record, a collection of software-error related information formatted in Structured Data Base (SDB) keywords.



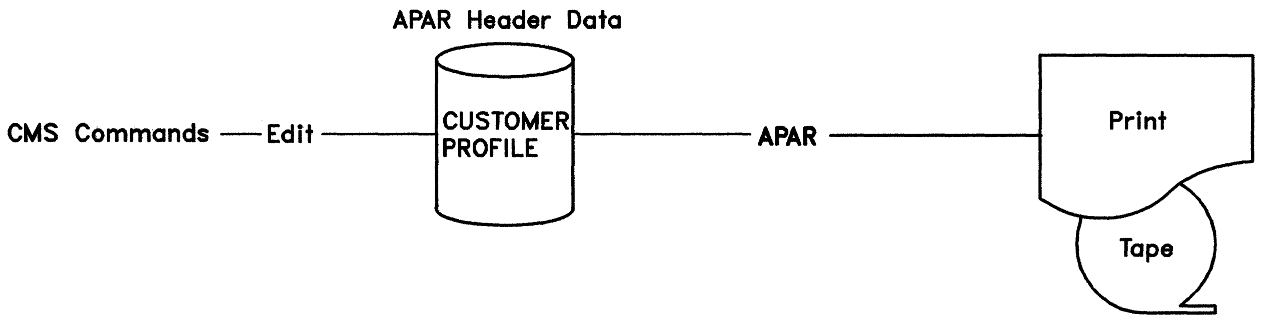
IPCS Load Map
contains the current nucleus load map and, where applicable, other secondary maps for the VM/SP system. This file is required for failure analysis by the IPCSDUMP program, and for the proper operation of the DUMPSCAN "MAPA" and "MAPN" subcommands.



PRBnnnnn aaaaaaaa
are any other files that you wish to associate with this problem. They may be created using any CMS facility or program, and should be given the filename associated with the problem and a filetype of your choice.



CUSTOMER PROFILE
is the file used by the APAR command to supply the header information for the hard-copy APAR form.



Commands and Subcommands

The following tables show the functions provided by the IPCS commands and the DUMPSCAN subcommands. Detailed information on each IPCS command is found in "Using IPCS Commands" on page 220 and on each DUMPSCAN subcommand in Appendix A, "Using DUMPSCAN Subcommands" on page 259.

IPCS Commands

IPCS Command	Function
APAR	Generates hard-copy APAR form to submit to IBM.
DUMPSCAN	Examine dump interactively.
IPCSDUMP	Process dump and create problem report.
MAP	Process nucleus load maps for use by the IPCSDUMP process.
PRB	Update problem status.
PROB	Create or add to a problem report.
PRTDUMP	Formats and/or prints, under CMS, a previously created dump file.
STAT	Lists status of individual problem or group of problems.

DUMPSCAN Subcommands

DUMPSCAN Subcommand	Dump Type	Function
reuse	Common	Reissue the previous CHAIN, LOCATE, or SCROLL subcommand.
?	Common	Display last subcommand entered.
+ or -	Common	Adjusts address pointer and issue DISPLAY.
&name	Common	Create synonyms for frequently used subcommands.
AREGS	CP	Display registers, CSW, and CAW for attached (non-IPL) processor.
ARIOBLOK	CP	Display RCHBLOK, RCUBLOK, and RDEVBLOK for the non-IPL processor.
C	CP	Display control registers for failing processor.

DUMPSCAN Subcommand	Dump Type	Function
CHAIN	Common	Verify control block chain.
CMS	Common	Enter CMS subset.
CMSPOINT	CMS	Display formatted contents of 17 pointers from CMS NUCON.
CORTABLE	CP	Display formatted contents of CORTABLE entry.
DISPLAY	Common	Display area in dump.
DOSPOINT	CMS	Display formatted contents of five pointers used by DOS simulation.
DUMPID	Common	Display dump identification and comment information from VMDUMP command line.
END	Common	End and return to CMS.
FDISPLAY	TSAF	Display the data control blocks, tables, and arrays important to the TSAF virtual machine.
G	Common	Display general register set for the failing processor or virtual machine.
HELP	Common	Display subcommand summary.
HX	Common	End and return to CMS.
IPCSMAP	Common	Add IPCS map to dump being viewed.
IUCV	GCS	Display all entries in the IUCV path table.
LOCATE	Common	Search for data in dump.
MAPA	Common	Locate module origin and displacement from that origin for address supplied.
MAPN	Common	Locate module name or entry point in load map.
MREGS	CP	Display registers, clocks, PSWs, CSW, and CAW for the main or the IPL processor.
MRIOBLOK	CP	Display RCHBLOK, RCUBLOK, and RDEVBLOK for the IPL processor.
OSPOINT	CMS	Display formatted contents of three pointers used by OS simulation.
PRINT	Common	Print data.
QUIT	Common	End and return to CMS.
REGS	Common	Display registers, clocks, PSWs, CSW, and CAW for the failing processor or virtual machine.
RIOBLOK	CP	Display RCHBLOK, RCUBLOK, and RDEVBLOK for the failing processor.

DUMPSCAN Subcommand	Dump Type	Function
SCROLL	Common	Display next or previous X'130' byte area from dump.
SYMP	Common	Format and display a symptom record.
TACTIVE	GCS	Display the task's active program list.
TLOADL	GCS	Display the task load list.
TRACE	CP TSAF	Displays trace table entries in hexadecimal or in a formatted display.
TSAB	GCS	Display the subpool map and chain header of a task.
USERMAP	CMS	Add user load map to dump being viewed.
VIOBLOK	CP	Display VCHBLOK, VCUBLOK, and VDEVBLOK.
VMBLOK	CP	Display userids from VMBLOK chain, or formatted data from selected userid VMBLOK.
VMLOADL	GCS	Display information about all NUCCBLKs on the VM load list.

Using IPCS

VM/SP IPCS usage covers these subjects:

- **Preliminary considerations**, which are those operations generally performed before starting to use the IPCS facilities for reporting problems, diagnosing problems, and managing problems and data.
- **Operating procedures**, which are those repetitive types of operations performed for each problem processed by IPCS.

Preliminary Considerations

Before any attempt is made to use IPCS to process dumps, several things must be considered:

- If any data files exist from the IPCS supplied with the VM/370 system, they must be made unavailable to VM/SP IPCS. There are two conversion functions that should be considered.
- The skeleton CUSTOMER PROFILE file supplied with the system must be edited to correctly describe the IPCS user.
- Adequate disk space must be made available to contain the various files which are used as input by IPCS, and files to be generated by IPCS.

- IPCS needs information about the system organization that produces each dump. This information is obtained from various load maps, which must be formatted for IPCS usage.
- If any updates are received for the IPCS modules, these modules must be reassembled and new load modules generated.

Conversion Considerations

The formats of various internal files and data areas, as well as internal interfaces for reading and writing dumps, creating problem reports, etc., have been changed for VM/SP IPCS. Therefore, VM/SP IPCS cannot view dumps or update symptom summary status for dumps taken by the IPCS supplied with the VM/370 system, and vice-versa. If your installation is already using the VM/370 IPCS and plans to install VM/SP IPCS, a method of converting the system IPCS symptom summary file and PRBnnnnn dumps to the format required by VM/SP IPCS has been provided. This conversion is accomplished by the use of the CONVERT command. For more information on converting symptom summary and dump files see Appendix F, "Converting Symptom Summary and Dump Files" on page 359.

Note: The trace table formatter function is valid for VM/SP and VM/SP HPO Release 4 dumps only. Unpredictable results will occur in the formatted output if this function is used against a pre-Release 4 dump.

If your installation has existing PVM or RSCS help files for use with IPCS (HELP PVM and HELP RSCSNET), these help files must also be converted to the proper format for IPCS usage. A utility, CONVIPCS EXEC, is provided to aid in this conversion.

See the *VM/SP Installation Guide* or *VM/SP HPO Installation Guide* for additional information on converting help files (CONVIPCS EXEC). See Appendix F, "Converting Symptom Summary and Dump Files" on page 359 for additional information on converting symptom summary and dump files (CONVERT command).

The existing VM/370 IPCS problem report files need not be converted for use with the PRB, PROB, or APAR commands of the VM/SP IPCS. The format of the problem report files will vary slightly from the VM/370 IPCS to the VM/SP IPCS, but the data will be accurate.

Completing the Customer Profile

One of the files provided with VM/SP IPCS is a CMS file named CUSTOMER PROFILE. As supplied, it contains all of the standard fields required for APAR submission. These fields, however, contain dummy data to illustrate the kind of information necessary. You should edit this file immediately after installing VM/SP IPCS and replace the dummy data with the correct information. If necessary, contact the local IBM Branch Office for information such as branch office address, telephone number, branch office number, etc., although this information is not essential. This

CUSTOMER PROFILE should be available so that the APAR submission routine will function properly.

Managing A-Disk Space

The amount of available IPCS virtual machine A-disk space can affect the operation of IPCS:

- Where adequate disk space is available, all problem reports, the IPCS load map, dumps, supplementary data files, and the symptom summary file are stored on disk. As a problem is resolved, the supplementary and dump data files for that problem can be erased. If the number of dumps resident on disk at a given time creates disk space problems, dumps can be temporarily stored on tape.

- Where disk space is limited, only the IPCS load map, the problem reports and symptom summary file must be kept on disk. All dumps created by IPCSDUMP and all supplementary data files associated with a problem can be temporarily stored on tape or printed, and retained for future reference to the problem. When processing a dump be sure you transfer it back to your A-disk.

For information about estimating disk storage and DASD space, see the *VM/SP Planning Guide and Reference* or *VM/SP HPO Planning Guide and Reference.*

Generating Load Maps

For proper operation of IPCSDUMP and some of the DUMPSCAN subcommands, certain mapping information is required. In this section you are shown how to generate a disk-resident IPCS load map for use by the IPCSDUMP command.

CP Load Map (CPNUC MAP) for IPCS:

Refer to the *VM/SP Installation Guide* or *VM/SP HPO Installation Guide* for installation procedures. Follow the example on how to build a new CP nucleus. When that process is complete, follow these steps to create a CP load map:

1. **IPL CMS**.

2. Read the CP load map onto the IPCS virtual machine A-disk and name the CP load map CPNUC MAP A1.

3. Use the MAP command to create CPIPCS MAP A1 which is recognizable by IPCS. Enter:

   MAP CP PROMPT

CMS Load Map (CMSNUC MAP) for IPCS: To create a CMS load map (CMSNUC MAP) for IPCS, follow the same procedure just described for the CP load map (CPNUC MAP) substituting the CMS map names found in Figure 19.

GCS Load Map (GCSNUC MAP) for IPCS: To create a GCS load map (GCSNUC MAP) for IPCS, follow the same procedure just described for the CP load map (CPNUC MAP) substituting the GCS map names found in Figure 19.

TSAF Load Map (TSAF MAP) for IPCS:

To create a TSAF load map for IPCS, make sure you have the following on an accessed disk:

- TSAF map source file with the default name of TSAF MAP
- CMS nucleus load map with the default name of CMSNUC MAP.

Issue:

MAP TSAF

to create the TSAFIPCS MAP.

Other Load Maps for IPCS:

To create load maps for PVM, RSCSNET, and RSCSV2 refer to their respective installation guides:

- *VM/SP Pass-Through Guide and Reference*
- *VM/SP RSCS Networking Program Reference and Operations Manual*
- *VM/SP RSCS Networking Planning and Installation.*

Then follow the CP load map example just described, substituting the component name with its corresponding map names. Select the correct map names from the following chart which lists each component with its default source map name, followed by the map name that IPCS creates. Repeat the load map procedure for any of the listed components.

Component	Default Source Map Name	Created IPCS Map Name
CP	CPNUC MAP	CPIPCS MAP
CMS	CMSNUC MAP	CMSIPCS MAP
GCS	GCSNUC MAP	GCSIPCS MAP

Figure 19 (Part 1 of 2). Generating Load Maps

Component	Default Source Map Name	Created IPCS Map Name
TSAF	CMSNUC MAP and TSAF MAP	TSAFIPCS MAP
PVM	CMSNUC MAP and PVM MAP	PVMIPCS MAP
RSCSNET (RSCS Vers.1, Rel.3)	RSCSNET MAP	RSCSIPCS MAP
RSCSV2 (RSCS Vers.2)	GCSNUC MAP	GCSIPCS MAP

Figure 19 (Part 2 of 2). Generating Load Maps

If there are any questions or problems, see your installation system programmer.

IPCS Load Map: After the nucleus load map (and secondary maps such as segment maps) have been created (the procedures were previously described for CP), the MAP command should be executed to provide an IPCS load map for subsequent appending to a dump. The MAP command should be executed for each load map. This IPCS map is normally created at system generation time, and should be updated when maintenance is applied and a new nucleus load map (or secondary map, if applicable) is created.

Operating Procedures

VM/SP IPCS is designed to run in a CMS virtual machine with user Class G (except for the case of IPCSDUMP command processing a CP system dump where only someone authorized to issue CP DIAGNOSE X'34' can do it) and a minimum storage size of 1M. There are no special requirements other than those documented in the Preliminary Considerations section. Installation procedures are documented in the *VM/SP Installation Guide* or *VM/SP HPO Installation Guide.*

The exact operating procedure for IPCS will vary depending on the type of problem encountered. A typical operating sequence is presented, along with additional information about the sources of data that might be used in diagnosing a problem. From this material, a specific sequence can be tailored to suit a particular problem.

Debugging Using IPCS

Typical Operating Sequence

The steps for a typical problem are as follows:

1. At system generation time, when the nucleus load map is created, run MAP to create the IPCS format load map (e.g., CPIPCS MAP, CMSIPCS MAP, etc.).

2. When a dump from any of the following dump types (CP ABEND, VMDUMP, Stand-alone DUMP, GCS GDUMP, TSAF, RSCS, or PVM) is placed in the virtual reader of the userid designated for IPCS, you should:

   a. Log on.
   b. IPL CMS with the IPCS disk accessed.
   c. Run IPCSDUMP to create the problem report, symptom summary, and disk resident dump.

3. Contact the IBM Support Center by phone for possible fix or duplicate problem recognition in the data base. Be prepared to give them problem information such as the symptom record, the problem report, and anything else pertinent.

4. Use PRB to change problem status to "reported to IBM."

5. Use DUMPSCAN to gather further information from the dump or to diagnose user problems.

6. Use PROB to add additional problem description information or references to supplemental data.

7. Use APAR to create a hard-copy APAR form and/or tape, or a hard-copy problem documentation.

8. Use STAT to find out the status of a problem or group of problems.

9. Use PRTDUMP to obtain a hard copy of the dump if it is required.

10. Use PRB to update the problem status.

Handling Supplementary Data

Identification of problem sources and resolutions of problems often require analysis of other data in addition to dumps. This data is termed "supplementary data" in IPCS. Procedures must be established to collect this data before problems occur.

Supplementary data includes console logs and other virtual machine information. The system operator should take the following steps to ensure this information is collected:

1. Spool the system console log or the virtual machine log to the IPCS userid so that you can use the system log file or virtual machine log file as supplementary data when necessary.

2. Use the CP privilege class B SET command as follows:

   SET DUMP AUTO ALL

   This command causes the CP abend dump to be placed in a spool file where it can be processed by the IPCSDUMP command.

Make available as supplementary data any other information that the:

- *VM/SP System Logic and Problem Determination Guide Volume 1 (CP)* or *VM/SP HPO System Logic and Problem Determination Guide - CP*
- *VM/SP System Logic and Problem Determination Guide Volume 2 (CMS)*

suggest for a given type of problem.

Console Logs: The system operator's console log may be useful in the event of a CP abend. At regular intervals during the day, the operator should close the system console, thus creating files in the IPCS userid's reader. Those files not associated with a problem may be reclaimed and printed (using the CP TRANSFER command), or purged.

To use the operator's console log when a problem occurs:

1. Display SUMMARY RECORD to see the next available problem number.

2. Read the console log file that corresponds to the time of the failure onto the A-disk and name the file PRBnnnnn CONSLOG, where nnnnn is the next problem number. This number will be used when you create the dump, therefore use it to create the file.

When the IPCSDUMP or PROB command creates the problem report for this problem, you are prompted for the names of any supplementary data files (as detailed in "IPCSDUMP Command" on page 233 and "PROB Command" on page 247).

Virtual Machine Dumps: When an abend occurs in a virtual machine (CMS, for example), you may use the CP VMDUMP command to create a dump for analysis:

1. Spool the virtual console log to the IPCS virtual machine.

2. Create the dump using the VMDUMP command.

3. Process the dump using the IPCSDUMP command in the same manner as a CP abend dump.

Trace Output: Some types of problems (loops in particular) are best analyzed using the CP TRACE command:

1. Spool the virtual printer or console (depending upon the TRACE command options used) to the IPCS virtual machine.

2. Display SUMMARY RECORD to see the next problem number to be assigned. This is the number that will be used when you create the dump.

3. Read the file onto the IPCS A-disk and name it PRBnnnnn TRACE, where nnnnn is the next problem number.

The CMS XEDIT or TYPE commands can be used to analyze this file.

Generating the Problem Report

IPCS can be used for reporting the following types of problem situations:

- CP system-detected failures
- CMS system-detected failures
- All user-detected failures.

To generate a problem report for these failures, use the IPCSDUMP command or the PROB command.

The IPCSDUMP command automatically gathers the failure analysis information from the dump. It then prompts you for additional problem description information.

The PROB command prompts you for a problem description *and* specific failure information. In both cases, you are prompted for the fileids of additional files of supplementary or diagnostic data associated with the problem (console logs, trace output, etc.), so that these fileids can be retained in the problem report file (PRBnnnnn REPORT). These files are referred to as supplementary data files.

IPCSDUMP or PROB assigns a 5-digit problem number when the problem report file is first generated. This number is unique for each problem and is used to refer to the problem for all future update information. This number is also appended to the prefix PRB to create the filename for the problem report file (for example, PRB00001 REPORT). The filename, PRBnnnnn, is used for all supplementary files associated with the problem so that all data for a given problem may be readily recognized and managed.

After the problem report is generated, the system checks the symptom summary file for possible duplicate problems already reported. One to ten possible duplicate problems are displayed with associated status information. For example: any PTFs (Program Temporary Fixes) associated with the problem are listed, closed or open status is noted, and the date the problem was detected is recorded. The current problem is entered in the symptom summary file whether or not a duplicate problem was found.

Recognizing Problems that are Duplicates

When a problem is reported (during PROB or IPCSDUMP command processing), its keyword information is compared with keyword information for all previously recorded problems, looking for possible duplicates.

It may happen that information about a problem entered via the PROB command or gathered by the IPCSDUMP command is minimal, for example, SCP (System Control Program) level, maintenance level, and abend type. If this is the case, possible duplicate status might be reflected against a problem that is not, in fact, a duplicate. Your analysis of the problem reports should reveal whether or not the problems are likely to be duplicates.

Use the STAT command to inquire about the current status of one or more problems.

Note: Since the maintenance level and SCP level of the system are keyword data, duplicates are not recognized across maintenance level or SCP level changes.

Updating the Problem Report

The PROB command puts additional information concerning a previously reported problem into the problem report file generated for that problem. When you issue the PROB command, the system prompts you for additional information, and then appends it to the existing problem report.

Using DUMPSCAN as a Debugging Tool for Dumps

DUMPSCAN allows you to interactively inspect the dump created by IPCSDUMP. DUMPSCAN is a problem source identification tool to aid you in analyzing the dump.

You can use the DUMPSCAN subcommands to view the code, the various control blocks, registers, and data areas as they appeared when the error was recorded. For CP dumps, the TRACE subcommand displays the CP trace table entries in either hexadecimal or in a formatted display.

Obtaining a Hard-Copy Listing from a Dump

Use the PRTDUMP command to obtain a hard-copy listing from a dump. Depending on the type of dump processed, some formatting of the hard-copy dump is possible. See the PRTDUMP command for further information.

Obtaining a Hard-Copy APAR form for Submission

Use the APAR EXEC to create a hard-copy APAR form on the line printer, and place all problem-related data either on the line printer or on a tape.

Using IPCS Commands

This section contains reference information for the IPCS commands. For proper usage, see "Notational Conventions" on page 221. Each command description includes format, keywords, operands, options, and responses, if any. Where applicable, usage notes are provided.

For information on error messages and return codes, see:

- *VM/SP System Messages and Codes* or *VM/SP HPO System Messages and Codes*
- *VM/SP System Messages Cross-Reference* or *VM/SP HPO System Messages Cross-Reference.*

You enter IPCS commands from a terminal on a CMS virtual machine. There are eight IPCS commands:

- **APAR** (a CMS EXEC) - invokes the functions of the PRB and PROB commands and produces a hard-copy APAR form for submission to IBM.
- **DUMPSCAN** - lets you interactively examine a dump and a symptom record existing as a CMS file created by IPCSDUMP. For more information on using DUMPSCAN see Appendix A, "Using DUMPSCAN Subcommands" on page 259.
- **IPCSDUMP** - reads the dump from the virtual reader, creates a CMS file containing the symptom record and dump, and creates a problem report by extracting pertinent data from the dump.
- **MAP** - processes nucleus load maps, and other secondary maps such as segment maps, to be appended to the dump by the IPCSDUMP command or the IPCSMAP subcommand of DUMPSCAN.
- **PRB** (a CMS EXEC) - updates the status of problems in the symptom summary file
- **PROB** - creates problem reports and adds information to existing problem reports.
- **PRTDUMP** - formats and/or prints, under CMS, the symptom record on the first page with a disk dump file previously created by IPCSDUMP.
- **STAT** - lists the current status of either a specific problem or a group of problems.

Note: Read/write access to the user's A-disk is required for proper execution of all IPCS commands.

Notational Conventions

The notation used to define the command syntax in this book is:

- Commands and subcommands are shown in uppercase and lowercase; the uppercase letters represent the minimum truncation of the command or keyword operand that the system accepts.
- An all-lowercase operand indicates a variable value supplied by you – for example, raddr (real address) or fn (filename).
- Where operands are shown between braces { }, one **must** be selected.
- Where operands are shown between brackets [ ], you can choose any one or none.
- An underscored operand is the system-selected default value used when no operand within that set of brackets is specified.

For additional information on syntactical representation of commands, see *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference.*

IPCS Command Formats and Usage

The IPCS commands are described as follows:

- APAR (see page 222)
- DUMPSCAN (see page 230)
- IPCSDUMP (see page 233)
- MAP (see page 240)
- PRB (see page 244)
- PROB (see page 247)
- PRTDUMP (see page 251)
- STAT (see page 254).

APAR Command

An APAR EXEC is included in IPCS to let you automatically generate a hard-copy APAR for submission to IBM. The EXEC prompts you for the problem number, the IBM assigned APAR number, and the ID of the component to be APARed. It lets you place the APAR documentation (e.g., the dump, load map, trace output, etc.) on tape or hard-copy.

The APAR EXEC executes:

- The PRB command to update the SYMPTOM SUMMARY and problem report with the APAR number
- The PROB command to add the component ID APARed and submitter's name and title to the problem report
- The PRTDUMP command to format and print the dump, and the CMS PRINT command for all other files, if the output is directed to the printer.

A CUSTOMER PROFILE is provided so that the EXEC can produce a hard-copy APAR that contains all the required information. You are expected to complete the CUSTOMER PROFILE, using the CMS System Product Editor, invoked by the XEDIT command, and keep it available for IPCS. The format of this file is shown in Figure 20.

```
CUSTOMER NAME = AMERICAN CUSTOMER COMPANY     CUST. NO. = 12345-67
CUST. ADDRESS = 123 ANYPLACE AVENUE
                ANYWHERE, NY 12345
CUST. CONTACT = JOHN Q. SYSPROGRAMMER
CUST PHONE    = 123-456-7890
--------------------------------------------------------------------
I.B.M. PS REP = PSR name                      EMPLOYEE NO. = 000000
I.B.M. ADDRESS= IBM CORPORATION
                street address
                city, state    zip
--------------------------------------------------------------------
REGION = 01  B/O = 123  WTC COUNTRY NO. =         NAME =
ITPS CODE = BROA  B/O TEL. = 123-456-7890     TIE LINE = 8-123-7890
--------------------------------------------------------------------
```

Figure 20. Sample Customer Profile

The format of the APAR command is:

APAR	

Usage Notes:

1. Because the SYMPTOM SUMMARY and problem report files are updated by the EXEC, they must reside on the A-disk and this disk must be accessed in read/write mode.

2. To ensure complete messages in the case of error, the EMSG setting is set to ON. If this conflicts with your normal setting, you should reset EMSG when the EXEC completes running.

3. If you select tape output, it is your responsibility to have a tape mounted on a tape drive attached at virtual address 181 before execution of the EXEC.

4. The APAR EXEC uses VMFPLC2 for all tape functions. (See the VM/SP Installation Guide for details.)

5. The APAR EXEC prompts you to find out whether you want the problem documentation printed or dumped to tape. It prompts you for the:

   - Problem number
   - APAR number
   - Component id
   - Any maps produced by the system generation process
   - Any additional files that do not meet the PRBnnnnn naming convention.

   All input accepted from you, with the exception of your name and title, is validated. Invalid input results in repetition of the prompt message.

6. You may terminate execution of the EXEC at any time by entering QUIT in response to any prompt message. Terminating the EXEC thus results in a termination message displayed at the terminal and a return code of zero. When the output is to tape, the tape is rewound and unloaded.

7. The hard-copy APAR produced by the APAR EXEC is a combination of the CUSTOMER PROFILE and the problem report. If the CUSTOMER PROFILE is not available, only the problem report is used.

Responses:

1. Common responses:

```
- - Executing VM/IPCS automatic APAR generation
              *         *         *         *
    A copy of the data required for the APAR form will be
    printed.
    Do you wish APAR documentation to be PRINTed or dumped
    to TAPE?
    Reply:  PRint | TApe
- - Enter (up to five) digits of problem number to be APARed:
- - The problem number to be APARed is 'PRBnnnnn'
- - Is this correct? Enter: YES | NO
- - Enter 7-digit APAR number assigned by IBM:
    (E.g.:  VM01234 or PP09876)
- - Enter component ID for this APAR:
    (E.g.: 5749 DMK00)
- - Enter your NAME:
- - Enter your TITLE:
    (E.g.: SYSTEM PRGMR)
- - Compid 'compid compgm' added to Problem Report
- - Enter name (fn ft fn) of load map (or 'NONE'):
    (Nucleus map produced by System Generation process,
    not IPCS MAP)
- - Enter name (fn ft fm) of any other files to be added that
- - do NOT have a filename of 'PRBnnnnn'.
- - If there are none, enter 'NONE'.
- - NOTE - -  EMSG has been set ON. Reset to normal
              status.
```

2. Tape responses:

```
- - This procedure will dump all problem-related data to TAPE
    using VMFPLC2. It will update your SYMPTOM SUMMARY to show
    that this problem has been APARed.
    To EXIT at any time, enter: QUIT
    *         *         *         *         *         *         *
    A tape drive at virtual address 181 is required.
    *         *         *         *         *         *         *
- - Proceeding to dump all problem-related data to TAPE .....
- - Checking contents of TAPE:
- - Is this correct?  Reply: YES | NO
- - Tape has been unloaded
- - Mark tape CLEARLY with: 'aparno' AND 'PRBnnnnn'
- - OPERATOR has been notified of tape identity
- - TAPE REWOUND - Correct invalid files and re-run
- - APAR exec
- - EXEC TERMINATED BY USER - TAPE HAS BEEN REWOUND
- - AND UNLOADED
```

3. Print responses:

```
- - This procedure will PRINT all problem-related data and
    update your SYMPTOM SUMMARY to show that this problem
    has been APARed. To EXIT at any time, enter: QUIT
    *        *        *        *        *        *        *
- - APAR 'PRBnnnnn aparno' has been printed
- - Proceeding to PRINT all problem related data .....
    Enter dump formatting parameters (if required),
    or press ENTER:
- - All data related to problem 'PRBnnnnn' has been
- - printed.
- - EXEC TERMINATED BY USER - PRINTER OUTPUT HAS BEEN PURGED
```

Error Messages:

```
| DMKCQG040E Device devtype does not exist
  DMMSUM100S ERROR 'nnn' READING FILE 'fn ft fm'
  DMMSUM502S PROBLEM 'PRBnnnnn' NOT FOUND IN SYMPTOM SUMMARY
  DMMSUM200S ERROR 'nnn' WRITING FILE 'fn ft fm'
| DMSTPJ043E Tapn(vdev) is file protected  [RC=36]

| DMSTPJ058E End-of-file or end-of-tape   [RC=40]
| DMSTPJ110S Error reading tapn(vdev)   [RC=100]
| DMSTPJ111S Error writing tapn(vdev)   [RC=100]
| DMSSTT002E File fn ft fm not found   [RC=28]

  - - There is NO tape drive attached at virtual address 181.
      Attach drive and mount tape (with ring), then restart APAR
      exec.
  - - PROBLEM NUMBER ENTERED IS NOT VALID
  - - APAR NUMBER ENTERED IS NOT VALID
  - - FORMAT OF COMPONENT ID ENTERED IS NOT VALID
  - - Unable to locate ' CUSTOMER PROFILE '
  - - Printing 'PRBnnnnn' REPORT as APAR ' PRBnnnnn aparno '
  - - TAPE DUMP FAILED ON 'PRBnnnnn' DUMP  ( RC = 'rc' )
  - - TAPE DUMP FAILED ON 'PRBnnnnn' RELATED FILES
      ( RC = 'rc' )
```

Sample Execution of the APAR EXEC: A sample session using the APAR EXEC follows. ("====>" indicates data entered by you; each word of your response is truncated to eight characters by the CMS EXEC processor):

```
    ====> APAR

- - Executing VM/IPCS automatic APAR generation

              *         *         *         *

    A copy of the data required for the APAR form will be
    printed.
    Do you wish APAR documentation to be PRINTED or dumped
    to TAPE?
    Reply:  PRint | TApe

    ====> TAPE
```

```
- - This procedure will dump all problem-related data to
    TAPE using VMFPLC2.  It will update your SYMPTOM SUMMARY
    to show that this problem has been APARed.
    To EXIT at any time, enter: QUIT

    *         *         *         *         *         *

    A tape drive at virtual address 181 is required.

    *         *         *         *         *         *

    Tape 181 on tape 582

- - Enter (up to five) digits of problem number to be APARed:

   ====> 4001


- - The problem number to be APARed is PRB04001 .
- - Is this correct? Enter: YES | NO

   ====> YES


- - Enter 7-digit APAR number assigned by IBM:
    (E.g.:  VM01234 or PP09876)

   ====> VM01234


*** APAR NUMBER POSTED ***
- - Enter component ID for this APAR: (E.g.: 5749 dmk00)

   ====> 5749 DMK00


- - Enter your NAME:

   ====> JOHN A SMITH


- - Enter your TITLE: ( E.g.: SYSTEM PRGMR)

   ====> SYSTEM PROGRMER
```

```
- - COMPID 5749 DMK00 ADDED TO PROBLEM REPORT.
PRT FILE 1342  FOR  USERID  COPY 01 NOHOLD
- - APAR ' PRB04001 VM01234 ' has been printed.
- - PROCEEDING TO DUMP ALL PROBLEM RELATED DATA TO TAPE .....
 Dumping.....
 PRB04001 DUMP     R1
VMFPLC2 DUMP PRB04001 VM01234 A1
 Dumping.....
 PRB04001 VM01234  A1
VMFPLC2 DUMP PRB04001 ANYFILE A1
 Dumping.....
 PRB04001 ANYFILE  A1
VMFPLC2 DUMP PRB04001 CONSFILE A1
 Dumping.....
 PRB04001 CONSFILE A1
VMFPLC2 DUMP PRB04001 REPORT R1
 Dumping.....
 PRB04001 REPORT   R1
- - Enter name (fn ft fm) of load map (or 'NONE'):
    (tNucleus map produced by System Generation process,
    not ipcs map.)

   ====> NUC04001 MAP R

 Dumping.....
 NUC04001 MAP      R1
- - Enter name (fn ft fm) of load map (or 'NONE'):
    (tNucleus map produced by System Generation process,
    not ipcs map.)

   ====> NONE

- - Enter name (fn ft fm) of any other files to be added that
- - do NOT have a filename of 'PRBnnnnn'.
- - If there are none, enter 'NONE'.

   ====> DATA FILE A

 Dumping.....
 DATA     FILE     A1
- - Enter name (fn ft fm) of any other files to be added that
- - do NOT have a filename of 'PRBnnnnn'.
- - If there are none, enter 'NONE'.

   ====> DATA2 FILE A

 Dumping.....
 DATA2    FILE     A1
- - Enter name (fn ft fm) of any other files to be added that
- - do NOT have a filename of 'PRBnnnnn'.
- - If there are none, enter 'NONE'.

   ====> NONE
```

```
- - Please verify contents of tape:
 Scanning....
 PRB04001 DUMP     R1
 PRB04001 VM01234  A1
 PRB04001 ANYFILE  A1
 PRB04001 CONSFILE A1
 PRB04001 REPORT   R1
 NUC04001 MAP      R1
 DATA     FILE     A1
 DATA2    FILE     A1
 END-OF-FILE OR END-OF-TAPE
- - Is this correct? Enter: YES | NO.

   ====> YES

- - Tape has been unloaded
- - Please mark tape clearly with: ' VM01234 ' and ' PRB04001 '
- - Operator has been notified of tape identity
```

See Figure 21 on page 229 for a sample APAR as created by the preceding example.

```
CUSTOMER NAME = AMERICAN CUSTOMER COMPANY     CUST. NO. = 12345-67
CUST. ADDRESS = 123 ANYPLACE AVENUE
                ANYWHERE, NY 12345
CUST. CONTACT = JOHN Q. SYSPROGRAMMER
CUST PHONE    = 123-456-7890
------------------------------------------------------------------------
I.B.M. PS REP = PSR name                      EMPLOYEE NO. = 000000
I.B.M. ADDRESS= IBM CORPORATION
                street address
                city, state    zip
------------------------------------------------------------------------
REGION = 01  B/O = 123  WTC COUNTRY NO. =          NAME =
ITPS CODE = BROA  B/O TEL. = 123-456-7890     TIE LINE = 8-123-7890
------------------------------------------------------------------------
PROBLEM: 00003   OCCURRED: 03/28/84   15:29:33     SEVERITY: 3
                  CREATED: 03/29/84   09:02
------------------ SYSTEM DESCRIPTION ----------------------------------
CPU TYPE: 0168        CPU SERIAL: 060707
------------------ PROBLEM DETERMINATION -------------------------------
SYMPTOMS:
ENV=CP              COMPID=5749DMK00     SCPLVL=600          MAINTLVL=007
FAILURE=ABENDPRG001 MODULE=DMKSCH        DISP=022E           INSTR=B060
ENTRY=DMKSCHDL
PROBLEM RELATED DATA:
RUNUSER ADDRESS AND NAME: 003E6930 MVSTEST
PROGRAM CHECK AT: 0002385E
GENERAL REGISTERS 0-15            00000000  BF58B000  001A6050  BF58B000
                                  00000000  FD865EFF  A00063EE  00002732
                                  A002234E  B00222CC  003F2FF0  003E6930
                                  5000D410  00023148  A000DFDA  00005C14
PRECEDING CODE:     B059  9140  B060  4770  C26E  9640       INSTR=B060B205
TESTING LOCAL CHANGES TO MVS
------------------ SUPPLEMENTARY DATA ----------------------------------
PRB00003 DUMP A1
PRB00003 TRACE A1
PRB00003 CONSLOG A1
------------------ STATUS AND TRACKING INFORMATION ---------------------
*** STATUS UPDATED       09:02:06 03/29 REPORT OPNIBM
*** STATUS UPDATED       10:24:18 04/01 APARED OPNIBM    APAR VM01234
*** ADDED ***            10:24:18 04/01
COMPID APARED IS 5749  DMK00
APAR ENTERED BY : JOHN A SMITH
TITLE : SYSTEM PROGRMER
```

Figure 21. Sample APAR Form

DUMPSCAN Command

DUMPSCAN creates an environment that lets you interactively inspect dumps, formatted as CMS files by IPCSDUMP. DUMPSCAN then prompts you for the dump filename and filemode. Once the dump is located, subcommands can be entered. CP, CMS, GCS, and TSAF provide subcommands within DUMPSCAN to let you interactively inspect dumps. For more information on the DUMPSCAN subcommands see Appendix A, "Using DUMPSCAN Subcommands" on page 259.

Note: RSCS and PVM provide subcommands for the DUMPSCAN environment for their respective dumps. See the following RSCS and PVM manuals for details on these subcommands:

- *VM/SP RSCS Networking Diagnosis Reference*
- *VM/SP Pass-Through Facility Logic.*

DUMPSCAN is most effective on a display terminal, where the unit of display is approximately one screen or X'130' bytes of data from the dump. If the terminal is a typewriter-like terminal, the unit of display is one line.

Use DUMPSCAN to look at the dump processed by the IPCSDUMP command:

- You can display:
  - Any chosen area specified directly (or indirectly) by its address
  - Registers, PSWs, timers, and clocks
  - The addresses in a chain of homogeneous control blocks
  - Any module or entry point by entry name
  - The symptom record.
- You can locate:
  - A string of hexadecimal or EBCDIC data between two addresses
  - The module containing a given address.
- You can print:
  - The displayed data resulting from the subcommands.

In addition, dump dependent subcommands may be available to allow other functions. For example:

- In a CP dump, you can display:
  - The trace table entries, by number of entries, and starting address
  - Real and virtual device control blocks by device address
  - A list of all logged-on users with their VMBLOK addresses and status
  - Formatted information from a selected user's VMBLOK

– The formatted contents of the CORTABLE entry for any real address.

The format of the DUMPSCAN command is:

DUMPSCAN	[**HELP** *nnnnn* [*fm* A]]

where:

nnnnn
: are the significant digits of the PRBnnnnn DUMP name; leading zeros may be omitted.

fm
: is the filemode of the disk containing the dump. The filemode defaults to A.

HELP
: invokes the CMS HELP screen of the DUMPSCAN command.

Usage Note:

If you invoke DUMPSCAN with no parameters, you are prompted for the dump file identification (dump number and filemode). When the dump is found, you can use one or more of the subcommands described in "DUMPSCAN Subcommand Formats and Usage." For more information on DUMPSCAN Scroll Interface see Appendix B, "DUMPSCAN Scroll Interface" on page 343.

Responses:

```
DMMDSC701R A VALID DUMP HAS NOT BEEN SPECIFIED
           PLEASE ENTER ONE OF THE FOLLOWING OPERANDS:
             NNNNN <MODE>     - TO VIEW PRBNNNNN FROM CMS DISK 'MODE'
             HELP             - TO REQUEST HELP INFORMATION
             END, QUIT, HX    - TO EXIT
```

is issued when DUMPSCAN is invoked with no parameters.

```
PROCESSING 'PRBnnnnn DUMP an' CREATED mm/dd/yy AT hh:mm:ss

*** READY ***       DUMP TYPE = 'type'
                              (for display terminals)
```

Note: 'type' = system type (e.g., CP or CMS).

-- or --

```
R*                            (for typewriter terminals)
```

Error Messages:

```
DMMCOM109S VIRTUAL STORAGE CAPACITY EXCEEDED
DMMCOM864E STORAGE INITIALIZATION INCOMPLETE
DMMDSC100S ERROR 'nnn' READING FILE 'fn ft fm'
DMMDSC705W 'type' DUMPSCAN FUNCTIONS NOT AVAILABLE
DMMDSC719I ERROR 'nnn' IN FSSTATE FILE 'fn ft fm'
DMMDSC856E UNABLE TO LOCATE 'type' 'routine' ROUTINE 'name'
```

IPCSDUMP Command

The IPCSDUMP command processes the following type dumps:

- CP ABEND dumps
- Dumps from the CP VMDUMP command
- Stand-Alone dumps
- GCS GDUMP
- TSAF dumps
- RSCS dumps
- PVM dumps.

The VMDUMP command may be invoked either as a user command, or within a subsystem via the CP DIAGNOSE X'08' or CP DIAGNOSE X'94' interface. For more information about the command format, see the *VM/SP CP Command Reference* or *VM/SP HPO CP Command Reference.* The IPCSDUMP command generates a problem report for all dumps except those identified as partial dumps. Maps are not appended to partial dumps (See "Partial Dump Usage" on page 235.).

The IPCSDUMP command also generates a symptom record, which is based on problem report information. The symptom record is created and written as the first physical record of the dump, except for the following areas:

- When a partial dump is requested in the IPCSDUMP command
- When problem report information generation is fully manual.

The generation of the problem report has three variations:

- Fully automatic
  - The data extraction routine supplies all of the necessary keyword data. You are prompted only for the severity, supplementary data, and text (e.g., as for CP ABEND dumps).
- Partially manual
  - You are prompted for the same information as if the process were fully automatic. However, the data extraction routine is unable to obtain some keyword data; therefore, you are prompted for this additional information (e.g., maintenance level for the CMS subsystem).
- Fully manual
  - The format of the dump is unrecognizable or there is no data extraction routine available for this type of dump. Therefore, you are prompted for *all* of the information contained in the problem report.

The IPCSDUMP command provides IPCS functions that:

- Identify the processor

- Read the dump
- Assign a unique number to the dump
- Append a condensed load map
- Extract keyword symptom information
- Collect additional information
- Create a problem report
- Create a symptom record and include it in the dump file
- Initiate a problem tracking entry
- Perform a duplicate search.

The format of the IPCSDUMP command is:

IPCSDUMP	

Usage Notes:

General:

1. The IPCSDUMP command processes the dump and performs automatic data extraction if it recognizes the dump format and if proper data extraction routines exist. For more information on IPCSDUMP extraction routines see Appendix C, "IPCS SVC 199 Services" on page 345.

2. A disk-resident IPCS load map, corresponding to the dump, is *required* for most types of dumps (not required for RSCS V2) for correct execution of the IPCSDUMP extraction function. Also, an appended IPCS load map is required for the MAPA and MAPN functions of DUMPSCAN. The IPCS map is created by the MAP command.

3. The filenames of dumps created by IPCSDUMP are of the form PRBnnnnn DUMP.

4. After IPCSDUMP generates the problem report (see Figure 18 on page 205), a search is made of the symptom summary file for a possible duplicate problem. One to ten possible duplicate problems may be displayed with their associated status. Whether or not the problem is a duplicate, an entry is made in the symptom summary file for this problem.

5. The keywords shown on the problem report provide the information necessary to perform a comprehensive data-base search for the problem. Figure 22 on page 237 lists those keywords that are valid for CP and CMS dumps.

6. IPCSDUMP first attempts to process any virtual machine dump. If it cannot find one, it then tries for a CP system dump. Only authorized users (those authorized to use CP DIAGNOSE X'34') can process CP system dumps. For others, an error message will be generated.

7. IPCS may require additional processing routines depending on the type of dump. For example, for DUMPSCAN, the following routines are needed for CP dumps:

   - DMKTED
   - DMKTEE
   - DMKTEF
   - DMKTEM
   - DMKTES.

   These text decks must be on an accessed disk when DUMPSCAN is invoked for a CP dump.

   For more information, see Appendix G, "IPCS Interface Files" on page 363.

Partial Dump Usage:

1. You may dump one or more portions of a running guest virtual machine for dump viewing purposes only. No problem number is assigned nor is a problem report generated. $$ must be specified as the first two characters in the VMTYPE field of the FORMAT operand.

2. If a partial dump is encountered, IPCSDUMP creates a CMS dump file named "PRB00000 DUMP." There is no prompting.

3. Only one PRB00000 DUMP file can exist at a time. If you wish to save the partial dump you must rename it to PRBnnnnn DUMP. Exercise caution in selecting nnnnn, not to conflict with the normal IPCSDUMP PRB-number assignment.

   ```
   Example:  RENAME PRB00000 DUMP A1   PRB90000 DUMP A1
   ```

4. The dump file does not have an appended map.

5. No problem report, symptom record, or SYMPTOM SUMMARY entry is created for a partial dump.

6. You have the use of **only** the common DUMPSCAN subcommands excluding the following:

   - MAPA and MAPN which use an appended map
   - IPCSMAP and USERMAP subcommands
   - SYMP subcommand.

CP Dump Usage:

In AP or MP mode, the data is extracted from the failing processor.

Responses:

```
DMMVAL806R FOR type, ENTER 'FN FT FM' OF THE maptype MAP,
           OR ENTER A NULL LINE, CMS, NONE, OR QUIT
DMMVAL824I 'type' IPCS MAP 'fn ft fm' APPENDED TO 'PRBnnnnn'
DMMVMF830R ENTER KEYWORD 'keyword='
```

Error Messages:

```
DMMCOM100S ERROR 'nnn' READING FILE 'fn ft fm'
DMMCOM109S STORAGE CAPACITY EXCEEDED
DMMCOM200S ERROR 'nnn' WRITING FILE 'fn ft fm'
DMMCOM864E INITIALIZATION NOT COMPLETE
DMMCPA805I xxxxxx ABEND CODE NOT RECOGNIZED

DMMINI100S ERROR 'nnn' READING FILE 'fn ft fm'
DMMINI400S ERROR 'nnn' CLOSING FILE 'fn ft fm'
DMMPRM200S ERROR 'nnn' WRITING FILE 'fn ft fm'
DMMPRM804E ERROR IN DATA EXTRACTION

DMMSUM501S INVALID PARM 'badparm' PASSED TO SUMMARY
           UPDATE PROGRAM
DMMVAL100S ERROR 'nnn' READING FILE 'fn ft fm'
DMMVAL200S ERROR 'nnn' WRITING FILE 'fn ft fm'
DMMVAL807I UNABLE TO LOCATE maptype MAP 'fn ft fm'
DMMVAL814E 'type' IPCS MAP NOT APPENDED

DMMVAL820I INSUFFICIENT MAP PROCESSING DISK SPACE
           FOR 'fn ft fm'
DMMVAL821I 'type' IPCS MAP 'fn ft fm' NOT VALID FOR
           DUMP 'PRBnnnnn'
DMMVAL822E IPCSMAP FUNCTION NOT SUPPORTED FOR 'type'
DMMVMF100S ERROR 'nnn' READING FILE 'fn ft fm'
DMMVMF200S ERROR 'nnn' WRITING FILE 'fn ft fm'

DMMVMF560E FILE 'PRBXXXXX DUMP A1' ALREADY EXISTS - RENAME
           OR ERASE
DMMVMF600S ERROR 'nnn' RENAMING FILE 'fn ft fm'
DMMVMF853S NO DUMP FILES EXIST
DMMVMF855I 'type' EXTRACTION ROUTINE NOT FOUND

DMMVMF856E UNABLE TO LOCATE 'type' ROUTINE 'name'
DMMVMF860E FATAL I/O ERROR READING DUMP
DMMVMF871E USER NOT AUTHORIZED TO PROCESS CP SYSTEM DUMPS,
           NO OTHER DUMPS EXIST
```

Keyword	Explanation	Example	**
CALLER	Name of module that called the module in error.	CALLER = DMKVSP	P
CMND	Command that caused the failure.	CMND = COPYFILE	P
COMPID	Component ID of failing component.	COMPID = 5749DMK00	AP
DATA	Type of incorrect output.	DATA = MISSING	P
DEGRADE	Type of performance degradation.	DEGRADE = VIRTMACH	P
DEVTYPE	Device type that produced the incorrect output or on which incorrect output was detected.	DEVTYPE = 3211	P
DISP	Displacement into the failing module at which the error occurred or was detected.	DISP = 003C	A
DOC	Type of documentation error discovered.	DOC = PUB	P
DOCNO	Document (manual) number in which error exists.	DOCNO = GC201801	P
ENTRY	Entry point of failing or calling module.	ENTRY = DMKVSPWA	A
ENVIR	Environment in which problem was detected (e.g., CMS, CP, GCS, TSAF, etc.).	ENVIR = CMS	AP

FAILURE	General type of failure reported:
	`Code        Meaning`
	`MESSAGE     AN error message was issued`
	`ABENDaaaaaa Where aaaaaa is the abend type`
	`INCORROUT   Incorrect output and/or results`
	`WAIT        Wait state`
	`LOOP        System in loop`
	`DOC         Documentation`
	`PERFORM     Performance`
	`INF         An information report`
		FAILURE = ABENDFRE001	AP

INSTR	First half word of failing instruction.	INSTR = 9018	A
MAINTLVL	Service level of the VM component which caused the dump.	MAINTLVL = 007	AP
MODULE	Module where the error appears to be located.	MODULE = DMSCPV	AP
MSGID	Exact format of the failing message identifier.	MSGID = DMKDMS123I	P
PAGE	Page number where error exists.	PAGE = 37	P
PREVMSG	Pertinent message preceding error message.	PREVMSG = DMSCPY300S	P
RCODE	Return code.	RCODE = 4	P
SCPLVL	Release level of the VM component which caused the dump.	SCPLVL = 006	AP
STATE	Type of loop state detected.	STATE = ENA	P
WAIT	Wait state code from PSW.	WAIT = 0E00	AP

** A=Automatically supplied by IPCSDUMP
P=Generated by prompting

Figure 22. Keywords Used in a PRBnnnnn REPORT to Identify Problems

Sample Initiation of an IPCSDUMP Operation:

After logon, you can take the following steps to initiate an IPCSDUMP operation ("====>" indicates data entered by you):

= = = = > **QUERY RDR ALL ***

```
ORIGINID FILE CLASS RECDS  CPY HOLD DATE  TIME     NAME   TYPE DIST
SYSTEM   0126 D DMP 000065 00  NONE MM/DD HH:MM:SS CPDUMP FILE
USER1    0088 V DMP 000061 00  NONE MM/DD HH:MM:SS VMDUMP CMS  BO12
```

Entering QUERY READER ALL * discloses that one of these files is a system generated Class D file; the other is a subsystem generated Class V file. Class D files are CP initiated dumps; Class V files are subsystem initiated dumps. These dumps, although not usable in their present state, are translated by subsequent IPCSDUMP processing.

CMS is first loaded into your virtual machine, and IPCSDUMP is invoked. The processing messages related to creating the file follow. If the file is processed and saved using the IPCSDUMP program, the CP spool file is erased unless your reader is spooled hold, and the newly created disk dump file becomes PRBnnnnn DUMP, where nnnnn is a number from 00001 through 99999.

The following example illustrates the sequence of events:

= = = = > **IPL CMS**
= = = = > **IPCSDUMP**

```
VM/IPCS --- BEGINNING DUMP PROCESSING ---
ASSIGNED PROBLEM NUMBER PRB00014
DMMVAL806R  FOR type, ENTER 'FN FT FM' OF THE CMS IPCS MAP,
            OR ENTER A NULL LINE, CMS, NONE, OR QUIT.
```

= = = = > **(NULL)**

```
DMMVAL824I CMS IPCS MAP 'CMSIPCS MAP A1' APPENDED TO
'PRB00014'.  ENTER SEVERITY.  1-4(N)
```

Note: Prompts occur only on exception conditions, such as no map present. If you said QUIT, indicating no IPCS map, you will be prompted for more information concerning the dump because the map was not available. It is always best to have the IPCS processed map which is associated with the dump to be processed by the IPCSDUMP command.

= = = = > **2**

```
ENTER FN FT FM OF SUPPORTING DATA AND DESCRIPTION
E.G. PRBNNNNN TRACE A1-CONS. OUTPUT OF TRACE
ENTER NAME OF FILE OR NULL
```

```
    ====> PRB00014 TRACE A1

ENTER NAME OF FILE OR NULL

    ====> (NULL)

ENTER TEXT DESCRIPTION OF PROBLEM OR NULL LINE
ENTER TEXT (MAX 80 CHAR/LINE)

    ====> USER TEXT DESCRIPTION

ENTER TEXT (MAX 80 CHAR/LINE)

    ====> (NULL)

THE FOLLOWING PROBLEM(S) ARE POSSIBLE DUPLICATES
00013  OPNUSR  01/26/85
VM/IPCS --- DUMP PROCESSING COMPLETED ---
SYMPTOM RECORD IS CREATED
Ready; T=0.40/1.65 14:01:53
```

MAP Command

The MAP command is used to convert various types of load maps into the proper format for use by IPCS.

The format of the MAP command is:

MAP	*type* [*Prompt*]

where:

type
: must be specified as CP, CMS, GCS, TSAF, PVM, RSCSV2, or RSCSNET. These are the only types currently supported by the map function. The map produced is disk resident and usable by IPCSDUMP.

Prompt
: specifies that the user does not want to use the assigned default names for input or output. A prompt message is issued for each filename required. The acceptable reply to this message is:

FN [FT [FM]]
: Filetype and filemode default to 'MAP' 'A1' respectively, if not specified.

or one of the following:

CMS
: Enter CMS subset.

A null line
: Use the assigned default name for this file.

QUIT
: Terminate the MAP command.

NONE
: Do not process this input map. This option is provided for secondary input maps only and is ignored for other files.

Usage Notes:

General:

1. The MAP command creates a disk resident IPCS format load map for appending to a dump during IPCSDUMP processing. For more information on MAP processing routines see Appendix C, "IPCS SVC 199 Services" on page 345.

2. The primary input to the MAP function is the system dependent nucleus load map. Once the IPCS load map has been created from the nucleus load map, other system dependent secondary maps, such as the CMS segment maps, may be processed and included in the IPCS map. The format of the CP and CMS nucleus maps must be CP loader output and the CMS segment maps must be CMS loader output.

3. In addition to ensuring that the input load maps are in the proper format, the following validity checking is done. For nucleus load maps, the first module or csect must begin at hexadecimal location zero and must be the proper name for the subsystem or SCP being processed (e.g., DMKPSA for CP). When adding secondary maps, the entire address range of the map must be completely outside the address range of all other maps previously included in the IPCS map.

4. Before appending the IPCS load map to the dump, the map is checked for validity by comparing the location of a predefined module or csect (usually the last module in the resident nucleus) in the map with its location in the dump.

5. The output is a compressed version of the input, containing only module, csect, and entry point names and their beginning addresses, with the entry point addresses sorted into ascending sequence.

6. IPCS may require additional processing routines depending on the type of dump. For example, for DUMPSCAN, the following routines are needed for CP dumps:

   - DMKTED
   - DMKTEE
   - DMKTEF
   - DMKTEM
   - DMKTES.

   These text decks must be on an accessed disk when DUMPSCAN is invoked for a CP dump.

   For more information, see Appendix G, "IPCS Interface Files" on page 363.

CP Dump:

1. The default name of the CP nucleus load map is CPNUC MAP A1.
2. The default name of the IPCS load map is CPIPCS MAP A1.

CMS Dump:

1. The default name of the CMS nucleus load map is CMSNUC MAP A1.

2. The default name of the IPCS load map is CMSIPCS MAP A1.

3. When 'CMS' is entered as the type, the CMSDOS segment map is included with the nucleus map in the IPCS load map. If this map is not applicable to your CMS system, you may bypass processing it by replying "none" to the DMMMAP806R prompt message.

4. The USERMAP subcommand of DUMPSCAN may be used to add a CMS user load map to the dump at some later time.

GCS Dump:

Refer to Chapter 5, "Debugging GCS" on page 153 for details regarding GCS dumps and map support.

TSAF Dump:

1. The default name of the TSAF load map is TSAF MAP A1.
2. The default name of the CMS nucleus load map is CMSNUC MAP A1.
3. The default name of the IPCS load map is TSAFIPCS MAP A1.

Other Dumps:

Refer to the following for details regarding their dumps and map support:

1. *VM/SP RSCS Networking Diagnosis Reference*
2. *VM/SP Pass-Through Facility Logic.*

The following chart lists each component with its default source map name, followed by the map name that IPCS creates:

Component	Default Source Map Name	Created IPCS Map Name
CP	CPNUC MAP	CPIPCS MAP
CMS	CMSNUC MAP	CMSIPCS MAP
GCS	GCSNUC MAP	GCSIPCS MAP

Component	Default Source Map Name	Created IPCS Map Name
TSAF	CMSNUC MAP and TSAF MAP	TSAFIPCS MAP
PVM	CMSNUC MAP and PVM MAP	PVMIPCS MAP
RSCSNET (RSCS Vers.1, Rel.3)	RSCSNET MAP	RSCSIPCS MAP
RSCSV2 (RSCS Vers.2)	GCSNUC MAP	GCSIPCS MAP

Responses:

```
DMMMAP802I FOR maptype MAP 'fn ft fm' WAS ADDED TO THE IPCS MAP
DMMMAP806R FOR type, ENTER 'FN FT FM' OF THE maptype MAP,
           OR ENTER A NULL LINE, CMS, NONE, OR QUIT
DMMMAP808I 'type' IPCS MAP 'fn ft fm' CREATED FROM MAP 'fn ft
           fm'
DMMMAP819R ENTER 'FN' OF THE 'type' IPCS MAP TO BE
           CREATED, OR ENTER A NULL LINE
```

Error Messages:

```
DMMMAP100S ERROR 'nnn' READING FILE 'fn ft fm'
DMMMAP200S ERROR 'nnn' WRITING FILE 'fn ft fm'
DMMMAP801I maptype MAP 'fn ft fm' IS NOT VALID
DMMMAP807I UNABLE TO LOCATE maptype MAP 'fn ft fm'
DMMMAP810I FORMAT OF maptype MAP 'fn ft fm' IS INVALID

DMMMAP813E MAP FUNCTION NOT SUPPORTED FOR 'type'
DMMMAP814E 'type' IPCS MAP NOT CREATED
DMMMAP815E PROCESSING ERROR IN 'type' MAP ROUTINE 'name'
DMMMAP816E 'type' IPCS MAP 'fn ft fm' ALREADY EXISTS

DMMMAP817E maptype MAP 'fn ft fm' OVERLAPS A PREVIOUS MAP
DMMMAP818E INPUT MAP LIMIT REACHED FOR 'type' IPCS
           MAP 'fn ft fm'
DMMMAP856E UNABLE TO LOCATE 'type' 'routine' ROUTINE 'name'
DMMMAP862E NO PARAMETERS ENTERED
DMMMAP863E INVALID OPERAND - operand
```

PRB Command

The PRB command updates the STATUS, FUNCTN, SEV, or DUP/APAR/PTF fields in a symptom summary record (the record associated with a given problem number). This command is an EXEC procedure that invokes the VM/SP IPCS program SUMMARY after the input data is checked for validity. Changes to the symptom summary record are also reflected in the problem report, and notes the change in the problem report for the record.

Use the PRB command to update the STATUS, FUNCTN, SEV, or DUP/APAR/PTF information for a specific problem number or to display a specific problem report.

The format of the PRB command is:

```
PRB   { nnnnn  { APAR       aparnumber              } }
      {        { CLOSE                              } }
      {        { DSPLY                              } }
      {        { DUPOF      { nnnnn      }          } }
      {        {            { aparnumber }          } }
      {        { IBM                                } }
      {        { NEEDINFO                           } }
      {        { PTFIS      [filename] filetype     } }
      {        { PTFON                              } }
      {        { SEV        [1 2 3 4]               } }
      {        { USER                               } }
      { HELP                                          }
```

where:

nnnnn
: identifies which problem report and symptom summary record to change; leading zeros may be omitted.

APAR *aparnumber*
: is the seven digits assigned to the APAR (Authorized Program Analysis Report). Along with the problem number, it identifies the problem to be resolved.

CLOSE
: indicates that the problem has been resolved.

DSPLY
displays the requested PRBnnnnn REPORT on the user's terminal.

DUPOF *aparnumber*
indicates that problem nnnnn is a duplicate of problem aparnumber.

HELP
produces information that tells you how to use the PRB command.

IBM
indicates that problem nnnnn has been reported to IBM.

NEEDINFO
indicates that more information is needed about problem nnnnn before it can be resolved.

PTFIS [*filename*] *filetype*
indicates that there is a PTF file named 'filename filetype' available for use in resolving problem nnnnn.

PTFON
indicates that the PTF has been applied to the system to resolve problem nnnnn.

SEV [*1 2 3 4*]
indicates that a severity number has been assigned to problem nnnnn, from 1 through 4; the lower the number, the greater the severity.

USER
indicates that problem nnnnn is a user problem.

Usage Note:

All status update transactions entered with the PRB command are added to the PRBnnnnn REPORT data file for the given problem (see Figure 18 on page 205).

The SEVERITY (or other information) listed on the first line of the PRBnnnnn REPORT is not changed, instead the status and tracking information section of the file is updated indicating the new severity (or other information).

The information in the SYMPTOM SUMMARY file for the updated problem is updated too.

Responses:

The following responses are displayed after you post the status of a problem and thus update it to a new state:

```
*** APAR NUMBER POSTED ***

*** PROBLEM CLOSED ***

*** PROBLEM POSTED AS DUPLICATE ***

*** PROBLEM POSTED AS REPORTED TO IBM ***

*** PROBLEM POSTED AS WAITING FOR INFO ***

*** PTF AVAILABLE POSTED ***

*** PROBLEM POSTED AS PTF APPLIED ***

*** SEVERITY UPDATED ***

*** PROBLEM POSTED AS AN OPEN USER PROBLEM ***
```

Error Messages:

```
NO PROBLEM NUMBER SUPPLIED
OPERAND NOT SPECIFIED
OPERAND nnnnn NOT VALID
ENTER PRB HELP TO OBTAIN COMMAND FORMAT
PTF FILE NOT GIVEN

DUP PROBLEM NUMBER NOT GIVEN
PROBLEM NUMBER MUST BE NUMERIC, nnnnn IS INVALID
PROBLEM NUMBER nnnnn IS TOO LARGE
APAR NUMBER NOT GIVEN
NEW SEVERITY MUST BE NUMERIC (1-4)

NEW SEVERITY MUST BE BETWEEN 1 AND 4
NEW SEVERITY NOT GIVEN
UPDATE NOT EXECUTED
PROBLEM NUMBER MAY BE 1 TO 5 DIGITS -- REENTER --
```

PROB Command

The PROB command lets you either enter, into the IPCS system, a problem which could not be handled by IPCSDUMP or add information to an existing problem report. Through a prompting technique, PROB systematically collects information from you about the problem.

If this is a new problem, PROB produces a problem report and adds the problem to the symptom summary file (see "IPCS Files" on page 203). It then searches the symptom summary file for previously entered problems of the same description, and informs you of any matches.

If you are adding information to an existing problem report, PROB prompts you for the additional information and adds it to the existing problem report. A sample session using the PROB command follows the description of the command.

The format of the PROB command is:

PROB	

Usage Notes:

1. This command creates a problem report by prompting you for information concerning a new problem that has resulted in a dump which could not be processed by IPCSDUMP. Use of the PROB command can also result in prompting for additional information to be added to an already existing problem report file.

2. When specific lengths or kinds of data are expected, IPCS checks the data and, if discrepancies are met, it prompts you again for the desired information.

3. When information may be updated by line number, prompts are prefixed by the line number (:Lnnnn). After a line number is displayed, if you wish to change the information you entered in response to a **previous** prompt, simply enter the line number exactly as it appeared at your terminal and you will be reprompted for the information. If the reprompt requested is a "branch" prompt (when subsequent prompting depends on the information entered), the prompting continues from that point in the prompting logic and all information entered subsequent to the original prompt is lost. If the reprompt request is not a "branch" prompt, you are returned to the point where you left off.

4. Information entered by you is used to search for already reported problems with the same symptoms. Therefore, **accuracy** and **consistency** in entering information is important.

5. Entering HX or :HX at any time during prompting causes immediate termination of the prompting program. No data is saved.

Responses:

New problem: A prompting sequence must be followed to gather problem descriptions and problem-related data.

Existing problem: You must supply the additional text description or the name given to the supplementary data file at this time. You can examine this file during problem analysis by using the CMS TYPE or XEDIT commands.

Error Messages:

None

Sample Session Using the PROB Command:

The following example shows a typical prompting sequence encountered when using the PROB command. An example of a reprompt request is included. In the example, "====>" indicates data that you enter.

```
====> PROB

  ****** DOES THIS PERTAIN TO AN EXISTING PROBLEM REPORT?
  (Y OR N)

====> N

  :L0022 DOES PROBLEM PERTAIN TO THIS CPU? (Y OR N)

====> Y

  :L0025 ENTER SEVERITY. 1-4 (N)

====> 2

  :L0026 IS BYPASS FOR PROBLEM REQUESTED? (Y OR N)

====> N

  :L0027 ENTER OPERATING ENVIRONMENT.
         CP,CMS,RSCS,VS1,VS2,DOS,ETC. (20 CHAR MAX)

====> CMS

  :L0028 ENTER COMPONENT ID IF KNOWN, EG   5749DMK00
  (MAX 10 CHAR)

====> 5749DMS00

  :L0029 ENTER SCP LEVEL. (1-3 CHARS-OMIT LEADING ZEROS)

====> 5

  :L0030 ENTER MAINT. LEVEL. (1-3 CHARS-OMIT LEADING ZEROS)
```

```
 ====> 8

   :L0031 ENTER DATE OF FAILURE. (MM/DD/YY)

 ====> 11/08/78

   :L0032 SELECT ONE OF THE FOLLOWING KEYWORDS
   ****** MSG   ABEND
   ****** DOC   PERFORMANCE (PER)
   ****** LOOP  INCORROUT (INC)
   ****** WAIT  INFORMATION (INF)
   ****** PTFERROR (PTF)

 ====> ABEND

   :L0037 ENTER ABEND CODE (6 CHAR MAX)

 ====> 0C8

   :L0038 ENTER FAILING MODULE IF KNOWN. EG DMKPAG (8 CHAR
   MAX)

 ====> DMSCPY

   :L0039 ENTER DISPLACEMENT WITHIN FAILING MODULE. (4 CHAR
   EXACTLY)

 ====> 004C

   :L0040 ENTER CALLING MODULE IF KNOWN (8 CHAR MAX)
```

At this point you decide to make a change to previously entered data on line 39. Entering ":l0039" causes IPCS to issue a reprompt for the information required for that response, and then continue with the original prompting sequence.

```
 ====> :L0039

   :L0039 ENTER DISPLACEMENT WITHIN FAILING MODULE. (4 CHAR
   EXACTLY)

 ====> 0348

   :L0040 ENTER CALLING MODULE IF KNOWN (8 CHAR MAX)

 ====> DMSSVC

   :L0041 ENTER COMMAND WHICH CAUSED FAILURE IF APPLICABLE

 ====> COPYFILE

   :L0059 ENTER LOCATION OF SUPPORTING DATA.
   :L0060 ENTER FN FT FM PLUS DESCRIPTION OR NULL WHEN DONE
```

= = = = > PRB00002 CONSLOG CONSOLE ACTIVITY AT TIME OF FAILURE

```
:L0060 ENTER FN FT FM PLUS DESCRIPTION OR NULL WHEN DONE
```

Because you have no additional files of supporting data to associate with the problem, a null line is entered to terminate this prompting sequence and continue with the next data request.

= = = = >   (NULL LINE)

```
:L0061 ENTER TEXT DESCRIPTION OF PROBLEM OR NULL LINE
:L0062 ENTER TEXT (MAX 80 CHAR/LINE)
```

= = = = > INFORMATION CONCERNING THE PROBLEM WILL BE SUPPLIED BY THE

```
:L0062 ENTER TEXT (MAX 80 CHAR/LINE)
```

= = = = > USER IN THIS AREA.

```
:L0062 ENTER TEXT (MAX 80 CHAR/LINE)
```

At this point you have no more text to enter. Entry of a null line terminates this particular prompt, which in this case is the last prompt in the entire sequence.

= = = = >   (NULL LINE)

```
THIS PROBLEM HAS BEEN ASSIGNED NUMBER 00002
Ready; T=0.12/0.71 12:59:30
```

PRTDUMP Command

The PRTDUMP command formats and/or prints the symptom record on the first page along with a disk dump file previously processed by IPCSDUMP.

The format of the PRTDUMP command is:

PRTDUMP	*PRBnnnnn* [*option...*]

where:

PRBnnnnn
the file name of a disk dump that has been previously created by IPCSDUMP. This parameter is required.

options:

CP dump

NOFORM
specifies that formatting of all control blocks be omitted.

NOREAL
specifies that formatting of real machine control blocks be omitted. If NOFORM is specified, this option is automatically set on.

NOVIRT
specifies that formatting of virtual machine control blocks be omitted. If NOFORM is specified, this option is automatically set on.

NOHEX
specifies that the hexadecimal printing of the dump be omitted.

NOMAP
specifies that the printing of the IPCS load map appended to the dump be omitted. This option is ignored unless NOHEX is also specified.

CMS dump
There are no available CMS options. The standard print routine, which prints the registers, the PSW, all the pages in the dump, and the IPCS map if appended, is used.

GCS dump
There are no available GCS options. For a GCS dump you will be prompted for the format, VSCS or VTAM. For an unformatted dump answer NO to both prompts, the resulting dump will be unformatted.

TSAF dump
There are no available TSAF options. The standard print routine, which prints registers, the PSW, all the pages in the dump, and the IPCS map if appended, is used.

Other dumps
Most other dumps (RSCS or PVM) do not provide any option for this command. Refer to their respective books for their options:

- *VM/SP RSCS Networking Diagnosis Reference*
- *VM/SP Pass-Through Facility Logic.*

Partial dump
There are no valid options. The standard print routine, which prints the registers, the PSW, and all the pages in the dump, is used.

Usage Note:

The first parameter entered is the name of an existing dump file. The name is of the form PRBnnnnn, where 'nnnnn' is the unique number assigned by IPCSDUMP. If no name is present, an error message is produced. All options, entered after the dump name, are passed to a formatting routine. They are not checked for validity. If there is no formatting routine, they are ignored. If there are no options or there is no formatting routine, PRTDUMP prints the symptom record on the first page, the PSW and register set obtained from the DMPINREC (dump information record) and all pages in the dump, followed by the IPCS map, if it was appended to the dump.

For more information on PRTDUMP formatting routines see Appendix C, "IPCS SVC 199 Services" on page 345.

Note: The option list is scanned into 8-character tokens.

Responses:

1. The following message is displayed upon entering PRTDUMP:

   ```
   PROCESSING FILE 'PRBnnnnn DUMP'
   ```

2. When the specified file has been formatted and printed, the following message is displayed:

   ```
   DUMP FILE 'PRBnnnnn DUMP' PROCESSED
   ```

3. IPCS may require additional processing routines depending on the type of dump. For example, for DUMPSCAN, the following routines are needed for CP dumps:

   - DMKTED
   - DMKTEE
   - DMKTEF
   - DMKTEM

- DMKTES.

These text decks must be on an accessed disk when DUMPSCAN is invoked for a CP dump.

For more information, see Appendix G, "IPCS Interface Files" on page 363.

Error Messages:

```
DMMPRT100S ERROR 'nnn' READING FILE 'fn ft fm'.
DMMPRT855I 'type' FORMAT ROUTINE NOT FOUND
DMMPRT856I UNABLE TO LOCATE 'type' FORMATTING ROUTINE 'name'
DMMPRT861E FILE 'fn' NOT FOUND.
DMMPRT862E NO PARAMETERS ENTERED.

DMMPRT863E INVALID OPERAND - operand
DMMTIU868I REQUIRED RESOURCES ARE NOT AVAILABLE, SYMPTOM RECORD
           FOR FILE 'PRBnnnnn' CANNOT BE DISPLAYED.
DMMTIU869I ERROR 'nnn' OCCURRED WHILE READING SYMPTOM RECORD
           FOR FILE 'PRBnnnnnn'.
DMMTIU870I SYMPTOM RECORD FOR FILE 'PRBnnnnn' CANNOT BE FOUND.
```

STAT Command

The STAT command lists the current status, as found in the symptom summary file, for a given problem, a subset of problems, or all problems. Requests for the status of all problems produces a file named STATALL LOCAL that you can print. The results of all other requests are displayed on the terminal for immediate viewing.

When making requests for a subset of problems, you can specify a valid status or failure type. The system then searches the symptom summary file for that subset.

The format of the STAT command is:

```
STAT    { nnnnn                                                }
        { ALL  [ {OPENUSER} ]1 [ ABend       ]2                }
        {      [ {OPNUSR  } ]  [             ]                 }
        {      [            ]  [ {DOC}       ]                 }
        {      [ {OPENIBM } ]  [ {DD }       ]                 }
        {      [ {OPNIBM  } ]  [             ]                 }
        {      [            ]  [ INcorr      ]                 }
        {      [ {OPEN}     ]  [             ]                 }
        {      [ {OPN }     ]  [ INF         ]                 }
        {      [            ]  [             ]                 }
        {      [ APARED     ]  [ {LOOP}      ]                 }
        {      [            ]  [ {LP  }      ]                 }
        {      [ {NEEDINFO} ]  [             ]                 }
        {      [ {NDINFO  } ]  [ MSg         ]                 }
        {      [            ]  [             ]                 }
        {      [ {PTFRCVD}  ]  [ {PERFORM}   ]                 }
        {      [ {PTFRCV }  ]  [ {PR     }   ]                 }
        {      [            ]  [             ]                 }
        {      [ PTFON      ]  [ {WAIT}      ]                 }
        {      [            ]  [ {WS  }      ]                 }
        {      [            ]  [             ]                 }
        {      [            ]  [ {PTFERROR}  ]                 }
        {      [ CLOSED     ]  [ {PE      }  ]                 }
        {                                                      }
        { HELP                                                 }
```

Note:
[1] One of these status keywords may be specified with the ALL operand.
[2] One of these failure keywords may be specified with the ALL operand.

where:

nnnnn
is the problem number of a problem whose status is to be displayed

HELP
produces information that tells you how to use the STAT command.

ALL
used alone creates a file containing all known problems. When used with keywords, displays on the terminal the following subsets:

Status Keywords:

APARED
indicates all problems for which an APAR has been submitted.

CLOSED
indicates all problems that have been resolved and are marked as completed.

NEEDINFO (NDINFO)
indicates all problems for which additional information has been requested.

OPEN (OPN)
indicates all problems that are either unresolved or not marked closed.

OPENIBM (OPNIBM)
indicates all unresolved problems that have been reported to IBM.

OPENUSER (OPNUSR)
indicates all unresolved problems that have not yet been reported to IBM.

PTFON
indicates all problems for which a PTF has been applied.

PTFRCVD (PTFRCV)
indicates all problems for which a PTF has been received.

Failure Keywords:

ABEND (AB)
indicates all problems where an abnormal termination has been detected.

DOC (DD)
indicates problems where a document is wrong or unclear.

INCORR (IN)
indicates all problems giving incorrect output or invalid results.

INF
indicates all problems that contain a request for information.

LOOP (LP)
indicates all problems that go into a loop either in CP or in a virtual machine.

MSG (MS)
indicates all problems in which an error message was the primary error indication.

PERFORM (PR)
indicates all problems causing poor performance.

PTFERROR (PE)
indicates all problems caused by applying PTFs with errors in them.

WAIT (WS)
indicates all problems causing a CP or virtual machine wait.

Usage Notes:

1. If the operand ALL is used alone, a CMS file containing the status of all problems known to the system is created.

   This file has a filename filetype of STATALL LOCAL. This file can be displayed using the CMS TYPE command or printed using the CMS PRINT command. Note that if nnnnn is 00000, STAT ALL is assumed, status keywords are ignored, and the STATALL LOCAL file is created.

2. Both a status keyword and a failure keyword may be used with the ALL operand in the same command. Either a status or a failure keyword may be entered first.

3. The status and/or failure keywords are used with the ALL operand to request a list of a category of problems. For example:

   STAT ALL OPEN ABEND retrieves all open problems with a failure type of ABEND.

   STAT ALL OPNIBM AB retrieves all open problems with a failure type of ABEND that have been reported to IBM.

4. When both status and failure keywords are specified, no notification is given of an invalid search argument unless both arguments are incorrect.

   a. If one of the keywords is invalid, a search is made with the valid one.

   b. If both of the keywords are valid, a search is made on the intersection of the two of them. A list is provided for all entries that match both arguments.

5. If both search arguments are selected from the same list (both from status or both from failure), the first argument is ignored and the second one is used.

6. If status and/or failure keywords are specified with a problem number, they are ignored.

Responses:

The status of the requested problems, preceded by a header line, is shown in Figure 23.

```
ENTER PROBLEM NUMBER. (OR 00000 FOR ALL)
STATALL LOCAL A1  FILE CREATED
NO MATCHES FOUND
PROBLEM NOT FOUND IN SYMPTOM SUMMARY FILE
```

Error Messages:

```
DMMSTA100S ERROR 'nnn' READING FILE 'SYMPTOM SUMMARY *'
DMMSTA200S ERROR 'nnn' WRITING FILE 'fn ft fm'
DMMSTA601S OPERAND 'xxxxxxxx' NOT RECOGNIZED
```

```
 (1)      (2)      (3)     (4)     (5)       (6)         (7)    (8)    (9)        (10)

 PROB    CREATED   LAST   FUNCTN STATUS DUP/APAR/PTF    RLSE   SEV FAILURE      ENVIR
 00009   12/13/84  12/14 REPORT OPNIBM                  600007 2   ABENDPTR001 CP
 00010   12/13/84  12/14 APARED OPNIBM APAR VM07891     600007 2   ABENDPRG001 CP
 00011   12/13/84  12/14 CLOSE  CLOSED USER             600007 3   INFORMATION CP
 00012   12/13/84  12/13 CREATE OPNUSR DUP OF 00011     600007 3   INFORMATION CP
 00013   12/14/84  12/14 REPORT OPNIBM                  600007 3   ABENDFRE001 CP
 00014   12/14/84  12/14 CREATE OPNUSR                  600007 4   INCORROUT   CMS
```

Figure 23. STATALL LOCAL File Generated by the STAT ALL Command

Output Fields of the STAT Command:

(1) PROB
The system assigned problem number of this problem (nnnnn).

(2) CREATED
The date this problem was entered into the IPCS system using PROB or IPCSDUMP.

(3) LAST
Date of last control record status change of this problem.

(4) FUNCTN
The last significant update function performed relating to this problem. Possibilities are: CREATE, REPORT, APARED, NDINFO, PTFRCV, PTFON, USER, and CLOSE.

(5) STATUS
The current status of this problem. Possibilities are: OPNUSR, OPNIBM, and CLOSED.

(6) DUP/APAR/PTF
The filename and filetype of a PTF that is intended to fix this problem. This field may alternatively indicate another problem of which this is a duplicate (DUP or APAR). If this is the case, the APAR number assigned by IBM or any PTF activity is reflected in the first recorded problem.

(7) RLSE
Release and maintenance level of the system at time problem was detected.

(8) SEV
Severity assigned to this problem.

(9) FAILURE
One-word description of the general type of failure.

(10) ENVIR
Environment—the general area in which the problem was detected.

Appendix A. Using DUMPSCAN Subcommands

The DUMPSCAN subcommands are functionally grouped into two categories:

1. Common subcommands:

   - REUSE (See page 263.)
   - + and - (See page 265.)
   - &NAME (See page 266.)
   - ? (See page 264.)
   - CHAIN (See page 272.)
   - CMS (See page 275.)
   - DISPLAY (See page 279.)
   - DUMPID (See page 283.)
   - END (See page 284.)
   - G (See page 289.)
   - HELP (See page 290.)
   - HX (See page 293.)
   - IPCSMAP (See page 294.)
   - LOCATE (UP) (See page 297.)
   - MAPA (See page 300.)
   - MAPN (See page 301.)
   - PRINT (See page 307.)
   - QUIT (See page 309.)
   - REGS (See page 310.)
   - SCROLL (See page 314.)
   - SYMP (See page 317.).

2. Subsystem dependent subcommands:

   a. CP dump subcommands:

      - AREGS (See page 268.)
      - ARIOBLOK (See page 269.)
      - C (See page 271.)
      - CORTABLE (See page 278.)
      - MREGS (See page 303.)
      - MRIOBLOK (See page 304.)
      - RIOBLOK (See page 312.)
      - VIOBLOK (See page 336.)
      - VMBLOK (See page 338.).

   b. CMS dump subcommands:

      - CMSPOINT (See page 276.)
      - DOSPOINT (See page 282.)
      - OSPOINT (See page 306.)
      - USERMAP (See page 334.).

   c. GCS dump subcommands:

      - IUCV (See page 296.)
      - TACTIVE (See page 320.)
      - TLOADL (See page 322.)
      - TSAB (See page 332.)

- VMLOADL (See page 341.).

d. TSAF dump subcommands:

- FDISPLAY (See page 285.).

e. CP and TSAF dump subcommands:

- TRACE (See page 324.).

f. Other subcommands:

Refer to the following RSCS and PVM manuals, respectively, for additional details regarding their subcommands:

- *VM/SP RSCS Networking Diagnosis Reference*
- *VM/SP Pass-Through Facility Logic.*

Note: The DUMPSCAN subcommands are presented in this appendix in alphabetical order.

Refer to the *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* or *VM/SP HPO Data Areas and Control Block Logic - CP* and *VM/SP Data Areas and Control Block Logic Volume 2 (CMS)* for control block definitions of the data displayed by the DUMPSCAN subcommands.

General Usage Notes:

1. If any subcommand is not recognized you are informed by messages:
   - DMMDSC732E
   - DMMDSC733E
   - DMMxxx863E.
2. If a requested address happens to be in a page that was not dumped, you are informed by message 708I.
3. The cuu-type address needs no leading zeros and may be up to three significant digits in length.
4. The output of the subcommands is usually described in terms of a screen display. Unless stated otherwise, the output on a typewriter terminal is the same without any specified screen clear operation.
5. IPCS may require additional processing routines depending on the type of dump. For example, for DUMPSCAN, the following routines are needed for CP dumps:
   - DMKTED
   - DMKTEE
   - DMKTEF
   - DMKTEM
   - DMKTES.

These text decks must be on an accessed disk when DUMPSCAN is invoked for a CP dump.

For more information, see Appendix G, “IPCS Interface Files” on page 363.

Reuse Subcommand

Functional Category: Common

Use the Reuse subcommand to reissue the previous CHAIN, LOCATE, or SCROLL subcommand.

The Reuse subcommand will reissue the previous TRACE subcommand for CP or TSAF, using the scroll direction that you set with the TRACE. If you did not specify the scroll operand with the TRACE subcommand, Reuse will default to scroll up.

(null line)	

Usage Notes:

1. Pressing the ENTER key with no data entered repeats the previous CHAIN, LOCATE, or SCROLL subcommand with an updated address.

2. On a typewriter terminal only the new current line is displayed.

3. The reuse function is valid for CHAIN only if error message DMMCHN610I has been displayed; otherwise, DMMDCS733E is displayed.

4. The running total of all members found will be displayed in the output of the reissued CHAIN subcommand.

Responses:

Full screen display from LOCATE or SCROLL subcommands.

Error Messages:

See LOCATE (page 297), SCROLL (page 314), and TRACE (page 324) subcommands.

Sample Output:

See LOCATE (page 297), SCROLL (page 314), and TRACE (page 324) subcommands.

? Subcommand

Functional Category: Common

Use the ? subcommand to display the last subcommand entered.

?	

Usage Notes:

(none)

Responses:

Display of last command line entered from the terminal.

Error Messages:

```
DMMDSC863E INVALID OPERAND - operand
```

Sample Output:

Display of last command line entered from the terminal.

+ and - Subcommands

Functional Category: Common

Use the '+' or '-' subcommand to adjust the address pointer and reissue the DISPLAY subcommand.

+ -	*number*

where:

number
is the hexadecimal number to be added or subtracted.

Usage Notes:

1. If you issue the '+' or '-' subcommand following the TRACE subcommand, the address of the last entry displayed by the TRACE subcommand is used as the value of the address pointer. It is this address pointer that is modified by the '+' or '-' subcommands.

2. Although there is no upper limit to the increment value, the resulting address must be within the range of the dump or an error message is displayed.

Responses:

A full screen display of the dump data with the current line (line 10) positioned at the calculated address.

Error Messages:

```
DMMDSC863E INVALID OPERAND - operand
DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
DMMSCR708I PAGE 'page' NOT FOUND IN DUMP
```

Sample Output:

The '+' and '-' subcommands have the same output as the DISPLAY subcommand. See the DISPLAY subcommand on page 279.

&name Subcommand

Functional Category: Common

Use the &NAME subcommand to create a table of frequently used subcommands which may be invoked by another name, or to invoke a subcommand by its other name.

&name **&**	[*subcommand*]

where:

- The name portion of this subcommand may be one-to-seven characters in length and must be preceded by the ampersand.
- &NAME alone invokes the subcommand.
- & lists the table entries.
- &NAME plus the subcommand establishes another name for the subcommand.

Usage Notes:

1. When entering data into the &NAME table, you may not enter another &NAME subcommand. For example:

   ```
   &name1 &name2
   ```

   is not allowed.

2. If you try to invoke an &name that is not in the table, the response is:

```
DMMDSC733E ERROR DETECTED WHILE PROCESSING THE SUBCOMMAND -
           subcommand
```

3. The subcommand in the table is not checked for validity until it is invoked by entering &name. Only then are any errors in it detected.

4. The PRINT subcommand is not allowed in the &NAME table.

5. All entries into the &name table are limited to eight characters for each operand. Up to seven operands may be entered for each &NAME.

6. Up to 64 operands may be contained in the &NAME table at any one time.

Responses:

If &NAME is entered, the response is from the subcommand executed.

If & is entered, a list of the current entries in the &NAME table is displayed.

If &NAME subcommand is entered, the ready response indicates the subcommand has been added to the &NAME table.

Error Messages:

```
DMMAMP721I THE &NAME TABLE IS FULL
DMMAMP722I INVALID ENTRY INTO &NAME TABLE
DMMDSC733E ERROR DETECTED WHILE PROCESSING THE SUBCOMMAND - subcommand
```

Sample Output:

Figure 24 gives an example of several commands entered using the & subcommand, and the display of those same commands by entering the & subcommand with no operands.

```
&chain1 chain 35250 8 35250
*** READY ***    DUMP TYPE = CP
&chain2 chain 18908 c 0
*** READY ***    DUMP TYPE = CP
&chain3 chain 3b6670 0 0
*** READY ***    DUMP TYPE = CP
&lkoper vmblok operator
*** READY ***    DUMP TYPE = CP
&ereal rioblok 00e
*** READY ***    DUMP TYPE = CP
&evirt vioblok 00e
*** READY ***    DUMP TYPE = CP
&
&CHAIN1 CHAIN 35250 8 35250
&CHAIN2 CHAIN 18908 C 0
&CHAIN3 CHAIN 3B6670 0 0
&LKOPER VMBLOK OPERATOR
&EREAL RIOBLOK 00E
&EVIRT VIOBLOK 00E
*** READY ***    DUMP TYPE = CP
```

Figure 24. Sample Output of the & Subcommand

AREGS Subcommand

Functional Category: CP dump only

Use the AREGS subcommand to display:

- Registers
- Clocks
- PSWs
- CAW

for the AP (non-IPL).

Aregs	

Usage Notes: (none)

Responses:

A formatted display of the contents of all the registers, clocks, PSWs, CSW, CAW, and timers in the attached or the non-IPL processor.

Error Messages:

```
DMMDCP863E INVALID OPERAND - operand
DMMREG708I PAGE 'page' NOT FOUND IN DUMP
DMMREG724I NON-IPL REGISTERS REQUESTED IN UNIPROCESSOR DUMP
```

Sample Output:

The output of the AREGS subcommand is the same as the REGS subcommand (see page 310), with the addition of a heading line indicating that the display is for the non-IPL processor.

ARIOBLOK Subcommand

Functional Category: CP dump only

Use the ARIOBLOK subcommand to display:

- RCHBLOK
- RCUBLOK
- RDEVBLOK

for the specified device attached to the non-IPL processor in an MP configuration.

ARIoblok	*raddr*

where:

raddr
is the hexadecimal device address.

Usage Notes:

The raddr-type address needs no leading zeros and may be up to four significant digits in length.

Responses:

The contents of the RCHBLOK, RCUBLOK, and RDEVBLOK for device 'raddr' attached to the non-IPL processor are displayed.

Error Messages:

```
| DMMDCP731E OPERAND MISSING OR INVALID
| DMMDCP863E INVALID OPERAND - operand
  DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
  DMMIOB712I DEVICE 'raddr' ADDRESS NOT FOUND
  DMMIOB726I NON-IPL RIOBLOKS REQUESTED IN NON-MULTIPROCESSOR DUMP
```

Sample Output:

Figure 25 gives an example of the output of the ARIOBLOK subcommand.

```
ARIO   E                  ON NON-IPL PROCESSOR
RCHBLOK CHAN 00XX          ADDRESS 04D7B0
000 00000040 00210000 0004D7B0 0004D7B0 00000000 10000000 0004D7B0 00000000
020 00000050 FFFF00A0 00F0FFFF 0140FFFF 0190FFFF 01E0FFFF 0230FFFF 0280FFFF
040 02D0FFFF 0320FFFF 0370FFFF 03C0FFFF 0410FFFF 0460FFFF FFFFFFFF FFFFFFFF

RCUBLOK UNIT 000X          ADDRESS 04CC20
000 00080001 20000000 0004CC20 0004CC20 0004D7B0 00000000 00000000 00000000
020 0004CC20 00000000 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 0040FFFF
040 0050FFFF 0060FFFF 00000000 00000000

RDEVBLOK DEV 000E          ADDRESS 03EFD0
000 000E0000 20821043 0003EFD0 0003EFD0 0004CC20 00000000 00000000 C1000000
020 00000000 00000000 0004DB30 00000000 40404040 00CD9410 00000000 00000000
040 00000000 00000500 00000000 00000000 32030000 00000000 00000000 00000000
060 00FF0000 00000000 00000000 00000000 00000000 00000000 00000082 00000000
080 000D0000 00440240 0003F050 0003F050
*** READY ***    DUMP TYPE = CP
```

Figure 25. Sample Output of ARIOBLOK Subcommand

C Subcommand

Functional Category: CP dump only

Use the C subcommand to display the control registers for the failing processor.

C	

Usage Notes:

On a display terminal, the screen is cleared and the control registers from the failing processor (AP mode) are displayed at the top of the screen.

For AP/MP dumps, the indicator that shows if the registers are associated with an IPL or non-IPL processor, is displayed. However, if absolute page zero (the prefix storage area) is not in the dump the indicator is not displayed. Absolute page zero is required to determine whether the failing processor is the IPL or the non-IPL processor.

Responses:

The formatted contents of the control registers from the failing processor are displayed.

Error Messages:

```
DMMDCP863E INVALID OPERAND - operand
```

Sample Output:

Figure 26 gives an example of the output of the C subcommand.

```
CTL REGS 0 - 15
80800CC0 02088B00 FFFFFFFF 00000000 00000000 00000000 00000000 00000000
00008A00 00000000 00000000 00000000 00000000 00000000 EFC00000 003FE4D0
*** READY ***    DUMP TYPE = CP
```

Figure 26. Sample Output of C Subcommand

CHAIN Subcommand

Functional Category: Common

Use the CHAIN subcommand to verify the chain of homogeneous control blocks that start at the specified location.

CHain	*fromhexloc increment endval*

where:

fromhexloc
is the address of the first control block to be checked.

increment
is the hexadecimal value to be added to the control block address to find the pointer to the next entry in the chain.

endval
is the hexadecimal value of the pointer in the last block of the chain.

Usage Notes:

1. If the number of control blocks found exceeds 144, error message DMMCHN610I is displayed. Entering a null line (Reuse function) will continue chain verification starting with the last address displayed.

2. If the address of the next control block in the chain has already been found in the current group of up to 144 members, error message DMMCHN611I is displayed.

3. CP chains whose hexadecimal addresses are greater than 16Mb cannot be displayed.

Responses:

The hexadecimal address of each member found in the chain (up to 144 members) and a decimal count of the number of members found, is displayed.

Error Messages:

```
DMMCHN610I POSSIBLE INVALID CHAIN - COMMAND TERMINATED
DMMCHN611I LOOP DETECTED IN CHAIN - COMMAND TERMINATED
DMMCHN708I PAGE 'page' NOT FOUND IN DUMP
DMMDSC863E INVALID OPERAND - operand
DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
```

Sample Output:

Figure 27 on page 273 gives an example of the output of the CHAIN subcommand for a valid chain. Figure 28 on page 273 gives an example of the output of the CHAIN subcommand for a chain with a loop. Figure 29 on page 274 gives an example of the output of the CHAIN subcommand for a chain that has more than 144 entries.

```
CHAIN 35250 8 35250
00035250  003F4538  003F4BC8  003F5258  003F58E8  003F5FD8  003F6728  003F6CC0
003F7970  003F6A28  003F8788  003EBD00  0037BC70  00381408  003EC840  003EB408
003EA620  003F7020  003F72B8  003F8B78  0035A0B8  0035A588  0037C978  0037CC10
00348D68  0038C0F0  003F91D0  003F98F8  003FBB38  003FC9D8  003FD2B0  003F5558
003E8340  003DA660  003E9530  002F1970  00359D68  002F4D68  003E02E0  003DD990
00373888  00358198  00358D68  002F1140  003DBD10  00352068  00325368  00348988
0037B9D8  002D9D68  002D9248  00326D68  003CE168  00388248  002C4458  002E63B0
0038C830  003E2100  003DE008  001CCD68  001C6570  0024ED68  003DC6A0  003730F8
00328570  00328D68  000E3A08  000E33F0  002F4570  002FE1A0  001ABD68  003701A0
00370578  000F2D68  000B64C8  000B6D68  000B6A60  000B7D68  001557C8  00228718
003E39E0  003EC128  003E5250  003D4578  003D3B98  003825A8  003E6D60  003E9C60
003F3930  003E5BD0  003E6FF8  003695A8  003E7718  003E9890  003E26E8  002E6648
00347908  0034F6D8  001B79A8  00274C30  002D9AD0  003FDBC0  003FEB78
   103 ENTRIES FOUND IN CHAIN
*** READY ***    DUMP TYPE = CP
```

Figure 27. Sample Output of CHAIN Subcommand (Valid Chain)

```
CHAIN 18908 C 0
00018908  003CDB48  003AA6B8  003B6808  003CA428  003E82F8  003D91D0  003F7790
003DE930  003A8FA8  00024A00  003B3FB8  003A19B8  003F7EC8  0004FFD0  003B6520
003B9A78  003B54A0  003AAD60  003AA688  003A94A0  003ACB90  003A4940  003A9EF0
003FF900  003F8400  003B0748  00370E70  003B3868  003AF6F8  003E4818  003B10C0
003D87F0  003B03A0  003AD490  003B1540  003AA6B8
   036 ENTRIES FOUND IN CHAIN
DMMCHN611I LOOP DETECTED IN CHAIN - COMMAND TERMINATED
*** READY ***    DUMP TYPE = CP
```

Figure 28. Sample Output of CHAIN Subcommand (Loop in Chain)

```
CHAIN 3B6670 0 0
003B6670  003A2228  003A2288  003A22E8  003A2348  003A24C8  003A2528  003A2588
003A25E8  003A2648  003A26A8  003A2708  003A2768  003A27C8  003A2828  003A2888
003A28E8  003A2948  003A29A8  003A2468  003A2A08  003A2AC8  003A2B28  003A2B88
003A2BE8  003A2C48  003A2CA8  003A2D08  003A2D68  003A2DC8  003A2E28  003A2E88
003A2EE8  003A2F48  003A2FA8  003A3368  003A33C8  003A3428  003A3488  003A3548
003A35A8  003A3608  003A3668  003A36C8  003A3728  003A3788  003A2A68  003A3848
003A38A8  003A3908  003A3968  003A37E8  003A3A28  003A3AE8  003A3C68  003A3EA8
003A3F08  003A3F68  003A3FC8  003A4028  003A4088  003A40E8  003A4148  003A4328
003A4388  003A43E8  003A4448  003A44A8  003A4508  003A4568  003A5228  003A52E8
003A5348  003A53A8  003A5408  003A54C8  003A5A08  003A5AC8  003A5B28  003A5B88
003A5BE8  003A5C48  003A5CA8  003A5D08  003A5528  003A5588  003A55E8  003A5648
003A56A8  003A5708  003A5768  003A57C8  003A5828  003A5888  003A58E8  003A5948
003A59A8  003A5A68  003AAA18  003AA5C8  003A39C8  003B05E0  003D3800  003A3D28
003A1748  003A0E48  003A4688  003C4290  003A1D48  003A4268  003C5220  003EC070
003A46E8  003A4208  003A4628  003A3CC8  003B4888  003A4D48  003A4AA8  003A49E8
003A4E68  003E1C18  003C5010  003A3A88  003BEC50  003ABAB0  003AE708  003D6B18
003A94D0  003ADF28  003B3E98  003B53F8  003D79A0  003A23A8  003C00C0  003C8A30
003AE8A0  003C6BE8  003A4F28  003A3308  003E3EB8  003C6FF0  003C5130  003D7E50
   144 ENTRIES FOUND IN CHAIN
DMMCHN610I POSSIBLE INVALID CHAIN - COMMAND TERMINATED
*** READY ***    DUMP TYPE = CP
```

Figure 29. Sample Output of CHAIN Subcommand (Long Chain)

CMS Subcommand

Functional Category: Common

Use the CMS subcommand to enter the CMS subset environment.

CMS	

Usage Notes:

The subset command RETURN restores the DUMPSCAN environment.

Responses:

On entry: CMS SUBSET

On return: *** READY *** DUMP TYPE = 'type'
(for display terminals)

-- or --

R* (for typewriter terminals)

Error Messages:

```
DMMDSC863E INVALID OPERAND - operand
```

Sample Output:

Figure 30 gives an example of the output of the CMS subcommand.

```
On entry:     CMS SUBSET

On return:    ***  READY  ***   DUMP TYPE = 'type'
                                      (for display terminals)

-- or --

R*                                 (for typewriter terminals)
```

Figure 30. Sample Output of the CMS Subcommand

CMSPOINT Subcommand

Functional Category: CMS dump only

Use the CMSPOINT subcommand to display the formatted contents of 17 pointers from CMS NUCON.

CMSPoint	

Usage Notes:

(none)

Responses:

```
LASTCMND= (last command executed)
PREVCMND= (previous command executed)
LASTEXEC= (last EXEC executed)
PREVEXEC= (previous EXEC executed)
CURRSAVE= (address of current system svc save area)

FREELOWE= (address of low extent of system storage)
MAINHIGH= (address of high extent of user storage)
FRDSECT = (address of free storage chain used by DMSFRE)
PGMSECT = (address of program interrupt routine)
IOSECT  = (address of I/O interrupt routine)

EXTSECT = (address of external interrupt routine)
ADTSECT = (address of active disk table)
DEVTAB  = (address of CMS device table)
DIOSECT = (address of disk I/O control)

SVCSECT = (address of svc handler control block used by DMSITS)
TAXEADDR= (address of terminal attention interrupt exit)
ALDRTBLS= (address of loader tables)
```

Error Messages:

```
DMMDCC708I PAGE 'page' NOT FOUND IN DUMP
DMMDCM863E INVALID OPERAND - operand
```

Sample Output:

Figure 31 on page 277 gives an example of the output of the CMSPOINT subcommand.

```
LASTCMND= CP
PREVCMND= LIST
LASTEXEC= PF
PREVEXEC= PROFILE
CURRSAVE= 0000C748
FREELOWE= 000E8000
MAINHIGH= 0002B500
FRDSECT = 00002F60
PGMSECT = 00002600
IOSECT  = 00002570
EXTSECT = 000024A0
ADTSECT = 000015F0
DEVTAB  = 00001390
DIOSECT = 00002940
SVCSECT = 000026A0
TAXEADDR= 00000000
ALDRTBLS= 00100000
*** READY ***    DUMP TYPE = CMS
```

Figure 31. Sample Output of CMSPOINT Subcommand

CORTABLE Subcommand

Functional Category: CP dump only

Use the CORTABLE subcommand to display page status and the formatted contents of the CORTABLE entry for the hexadecimal location specified.

CORtable	*hexloc*

where:

hexloc
is the hexadecimal address desired.

Usage Notes:

(none)

Responses:

The formatted contents of the CORTABLE entry for the address requested are displayed.

Error Messages:

```
DMMCOR609I REQUESTED ADDRESS NOT IN REAL STORAGE
DMMCOR708I PAGE 'page' NOT FOUND IN DUMP
DMMDCP863E INVALID OPERAND - operand
DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
```

Sample Output:

Figure 32 gives an example of the output of the CORTABLE subcommand.

```
COR 4400
CORFPNT = 00030FB0 CORBPNT  = 00000000 SWPTABLE = 0039C148 PAGTABLE = 0039C108
CORFLAG = 02      USE = OWNED BY CP
*** READY ***     DUMP TYPE = CP
```

Figure 32. Sample Output of CORTABLE Subcommand

DISPLAY Subcommand

Functional Category: Common

Use the DISPLAY subcommand to display areas in the dump. The actual address or an indirect address may be specified.

Display	{ *hexloc* *hexloc%* *Thexloc* *Thexloc%* }	[*nnnn*]

where:

hexloc
: is the starting address to be displayed. (% gives indirect addressing.)

Thexloc
: provides compatibility with the CP DISPLAY command. No other leading letter is acceptable. (% gives indirect addressing.)

nnnn
: is the length in bytes to be displayed.

Usage Notes:

1. Use of the DISPLAY subcommand (*without* the nnnn operand) causes a full-screen display on a display terminal. If a line-mode terminal is in use, output is limited to one line. In all cases, however, the minimum DISPLAY subcommand output is one 16-byte line with translation.

2. If a % is put on the end of hexloc, that address is rounded down to a word boundary. The full word at that address is obtained, and its low-order three bytes are then used as the display address.

3. If DISPLAY hexloc% is entered, the redisplay of the command shows the indirect request as entered and the actual address displayed.

4. Hexloc followed by a period (.) rather than a space before the nnnn operand is also acceptable.

5. DISPLAY hexloc with no length also displays the storage key for the 2K half-page containing the specified hexloc. The key appears between the hexadecimal data and its EBCDIC translation on the "current" line marked with an arrow.

Responses:

For DISPLAY hexloc, there is a full-screen display of hexadecimal and translated dump data with the current line (line 10) positioned at hexloc.

For DISPLAY hexloc nnn, there is a display of 'nnn' (rounded up to a multiple of 16 decimal) bytes of hexadecimal and translated data from the dump.

Error Messages:

```
DMMDIN702I NON-HEX CHARACTER IN COUNT - RETRY
DMMDIN703I NON-HEX CHARACTER IN ADDRESS - RETRY
DMMDIN708I PAGE 'page' NOT FOUND IN DUMP
DMMDSC863E INVALID OPERAND - operand
DMMFEX703I NON-HEX CHARACTER IN ADDRESS - RETRY
DMMFEX708I PAGE 'page' NOT FOUND IN DUMP
```

Sample Output:

Figure 33 gives an example of the output of the DISPLAY subcommand. Figure 34 on page 281 gives an example of the output of the DISPLAY command with indirect addressing.

```
D 000550B0
    00055040  14F115EF  4780C04C  1FE1186E  B711B010     .1.....<...>....
    00055050  B1201000  4780C062  412000C0  58F004F8     ..............0.8
    00055060  0A084770  CA5C1806  06604460  C0CA1E90     .....*...-.-....
    00055070  1E101B30  18634720  C04C5890  D00C4810     .........<......
    00055080  90025510  CB1447B0  C9F48B10  00024780     ........I4......
    00055090  C0D09180  B1994780  CA0447F1  C09A47F0     ..j..r.....1...0
    000550A0  C17447F0  C20E47F0  C23647F0  C26647F0    A..0B..0B..0B..0
==> 000550B0  C29A47F0  C31247F0  C38247F0  C3F647F0  04 B..0C..0Cb.0C6.0
    000550C0  C40247F0  C40E47F0  C422D200  90002000    D..0D..0D.K.....
    000550D0  98129010  18315430  04984770  C9EC5520    q.........q..I...
    000550E0  CB184740  C9EC45A0  C6FCB711  B010B120     ... I...F.......
    000550F0  10004780  C1004120  00C058F0  04F80A08     ....A......0.8..
    00055100  4770CA5C  58F0CB1C  05EF4770  CA145880     ...*.0..........
    00055110  B1A01288  4770C128  D703D00C  D00C1889     ...h..A.P......i
    00055120  5080B1A0  B2058030  98019010  90018010    &.......q.......
    00055130  4300B0A9  4200802C  9420802E  91809000     ...z....m...j...
    00055140  4780C14E  D2078008  90089680  802E9140     ..A+K.....o...j
    00055150  90004780  C15A9640  802E9200  80009680     ....A!o ..k...o.
    00055160  B19947F0  CAEC900B  D0105FC0  C9BAD703     .r.0........¬.I.P.
*** READY ***     DUMP TYPE = CP
```

Figure 33. Sample Output of DISPLAY Subcommand

```
DISPLAY 2C% 40   (00043B2A)
    00043B20  4550C9BA  D5074008  D0404780  CB3E5830    .&I.N. .. ......
    00043B30  40001A43  12334780  CBDC47F0  CB249104     ..........0..j.
    00043B40  D00E4710  CB744820  402C8920  000C5850    ........ .i....&
    00043B50  D0245020  50004830  40280630  89300002    ..&.&... ...i...
    00043B60  4823402E  8920000C  5E2004C8  5850D028    .. .i...;..H.&..
*** READY ***    DUMP TYPE = CP
```

Figure 34. Sample Output of DISPLAY Subcommand (Indirect Addressing)

DOSPOINT Subcommand

Functional Category: CMS dump only

Use the DOSPOINT subcommand to display the formatted contents of five pointers used by DOS simulation.

DOSPoint	

Usage Notes:

If the DOSPOINT subcommand is invoked and DOS simulation is not in effect, an error message is displayed in addition to the formatted display.

Responses:

```
BGCOM  = (address of background communications area)
SYSCOM = (address of systems communication area)
LTASAVE= (address of logical transient save area)
ACBLIST= (address of acb list built by open/close)
DOSSECT= (address of first DOSCB control block)
```

Error Messages:

```
DMMDCD728I DOS SIMULATION NOT IN EFFECT
DMMDCD708I PAGE 'page' NOT FOUND IN DUMP
DMMDCM863E INVALID OPERAND - operand
```

Sample Output:

Figure 35 gives an example of the output of the DOSPOINT subcommand.

```
DMMDCD728I DOS SIMULATION NOT IN EFFECT
BGCOM  = 00000DB8
SYSCOM = 00000CA0
LTASAVE= 00001180
ACBLIST= 00000000
DOSSECT= 00000000
*** READY ***    DUMP TYPE = CMS
```

Figure 35. Sample Output of DOSPOINT Subcommand

DUMPID Subcommand

Functional Category: Common

Use the DUMPID command to display a dump identification message and the dumpid information contained within the dump.

DUMPID	

Usage Notes:

1. If the dumpid field contains zeros, an error message is displayed.

2. CP dumps do not contain dumpid information, so message DMMDID729I always appears for CP dumps. The "PROCESSING 'PRBnnnnn DUMP nn' CREATED mm/dd/yy AT hh:mm:ss" response appears before the message is displayed.

Responses:

```
PROCESSING 'PRBnnnnn DUMP nn' CREATED mm/dd/yy AT hh:mm:ss
```

Up to 100 bytes of dumpid information are displayed on the terminal.

Error Messages:

```
DMMDID729I NO DUMPID INFORMATION FOUND
DMMDSC863E INVALID OPERAND - operand
```

Sample Output:

Figure 36 gives an example of the output of the DUMPID subcommand.

```
PROCESSING 'PRB00014 DUMP A1' CREATED 04/14/84 at 13:42:10
FAILURE WHILE RUNNING USER PROGRAM 'ABC'
*** READY ***    DUMP TYPE = CMS
```

Figure 36. Sample Output of DUMPID Subcommand

END Subcommand

Functional Category: Common

Use the END subcommand to end the session and return to CMS.

END	

Usage Notes:

END, HX, and QUIT are functionally equivalent. All are implemented to provide compatibility with accepted CMS usage.

Responses:

```
Ready; T=n.nn/n.nn hh:mm.
```

Error Messages:

(none)

Sample Output:

Figure 37 gives an example of the output of the END subcommand.

```
Ready; T=n.nn/n.nn hh:mm.
```

Figure 37. Sample Output of the END Subcommand

FDISPLAY Subcommand

Functional Category: TSAF dumps only

Use the FDISPLAY (Formatted Display) subcommand to display data control blocks, tables and arrays important to the TSAF virtual machine.

```
FDISPlay   { ALL                     }
           { COLLect                 }
           { LINKCtl     {BSC }      }
           {             {CTCa}      }
           { LINKDef                 }
           { NEIGhbor                }
           { PATH                    }
           { RESOurce                }
           { ROUTing                 }
           { SERVice                 }
```

where:

ALL
displays all of the information produced by the operands of this command. This information is useful when you want a hard copy of the information.

COLLect
displays the collection control block.

LINKCtl
displays the link control blocks for the:

- Channel-to-Channel adapter drivers (CTCA), (this includes channel-to-channel adapters and 3088 drivers), and
- Bisynchronous Drivers (BSC).

LINKDef
displays the link definition array.

NEIGhbor
displays the neighbor table.

PATH
displays the path array.

RESOurce
displays the resource table.

ROUTing
displays the routing array.

SERVice
displays the service table.

Usage Notes:

To produce a SPOOL file with the data, issue:

PRINT FDISPLAY ALL

PRINT TRACE FORMAT FOR *nnn*
(Where *nnn* is the number of trace entries. The maximum this number can be is 999.)

PRINT CLOSE

Responses:

(none)

Error Messages:

```
ATSZTD078E  OPERAND MISSING OR INVALID
ATSZTD084I  PAGE 'page' NOT FOUND IN DUMP
ATSZTD089E  UNABLE TO LOCATE GLOBAL CONTROL BLOCK (ATSCGM)
```

Sample Output:

Figure 38, Figure 39, Figure 40, and Figure 41 on page 288 give examples of the output of the FDISPLAY subcommand.

```
 ***  TSAF Link-Definition Table ***

Read/Write units: 03E0/03E0        ATSLINKS FILE record number: 4
Number of bytes read: 16560        Number of bytes sent: 18040
Time link came up: 0               Link delay: 154
Driver index: 1                    Link control block address: 003F9D40
Link state: 1                      Link flags: 'A00000'X

Read/Write units: 0342/0342        ATSLINKS FILE record number: 20
Number of bytes read: 544988       Number of bytes sent: 9149564
Time link came up: 0               Link delay: 16
Driver index: 1                    Link control block address: 003F9CF8
Link state: 1                      Link flags: 'C00000'X
*** READY ***    DUMP TYPE = TSAF
```

Figure 38. Sample Output of FDISPLAY Subcommand (LINKDef operand)

```
 ***  BiSync Link Control Blocks  ***

 ATSZBD096I THE DATA STRUCTURE IS EMPTY
*** READY ***    DUMP TYPE = TSAF
```

Figure 39. Sample Output of FDISPLAY Subcommand (LINKCtl BSC operand)

```
 ***  CTC/3088 Link Control Blocks  ***

Write CCW:  01032F90 20000004         Read CCW:  022C0408 200001F8
Sense CCW:  043F9D65 20000001         Link number:  1
Send queue front pointer:  00000000   Send queue rear pointer: 00000000
Link state: 0                         Sense byte: '40'X
*** READY ***    DUMP TYPE = TSAF
```

Figure 40. Sample Output of FDISPLAY Subcommand (LINKCtl CTCa operand)

```
  ***  TSAF Routing Table  ***

 Destination processor:  GDLVMBUB       Via link number:   2
 Entry flags:  '80000000'X              Path version number:  9
 Link weight:  50                       Hop count:  0

 Destination processor:  ZELDA          Via link number:   1
 Entry flags:  '80000000'X              Path version number:  9
 Link weight:  50                       Hop count:  0
 *** READY ***    DUMP TYPE = TSAF
```

Figure 41. Sample Output of FDISPLAY Subcommand (ROUTing operand)

G Subcommand

Functional Category: Common

Use the G subcommand to display the set of GPRs in the failing processor or virtual machine.

G	

Usage Notes:

The GPRs are displayed. On a display terminal, the screen is cleared and the formatted contents of the registers are displayed at the top of the screen.

For AP/MP dumps, an indicator that shows if the registers are associated with an IPL or non-IPL processor, is displayed. However, if absolute page zero (the prefix storage area) is not in the dump the indicator is not displayed. Absolute page zero is required to determine whether the failing processor is the IPL or the non-IPL processor.

Responses:

The formatted contents of the GPRs.

Error Messages:

```
DMMDSC863E INVALID OPERAND - operand
DMMREG708I PAGE 'page' NOT FOUND IN DUMP
```

Sample Output:

Figure 42 gives an example of the output of the G subcommand.

```
GEN REGS 0 - 15
0038E770 0000000C 0038E780 0027CD08 00000000 003F9EB0 0001E6C0 0001E240
00019288 00018908 003B1108 003825A8 00008878 00009878 50008886 00000000
*** READY ***    DUMP TYPE = CP
```

Figure 42. Sample Output of G Subcommand

HELP Subcommand

Functional Category: Common

DUMPSCAN passes the parameters specified on the HELP invocation to the CMS HELP facility. If no parameters are specified on the subcommand line then the HELP DUMPSCAN MENU is passed to the CMS HELP facility.

A simple subset of the operands most useful in the DUMPSCAN session is discussed in this section. For further information on the CMS HELP command see the *VM/SP CMS Command Reference.*

The format of the HELP subcommand subset is:

HELP	[**IPCS**	*subcommand*
	IPCS	**MENU**
	DUMPSCAN	**MENU**
	MESSAGE	*message-id*
	MSG	*message-id*]

where:

IPCS *subcommand*
: displays HELP information for the specified IPCS command or subcommand.

IPCS MENU
: displays the main IPCS HELP menu.

DUMPSCAN MENU
: displays the main DUMPSCAN HELP menu.

MESSAGE *message-id*
MSG *message-id*
: displays HELP information concerning a specified message. *message-id* is the 7, 8, 10 or 11 character message-id you specify to display the HELP file for a message. Specify the message-id in one of four forms:

```
xxxnnnt
xxxnnnnt
xxxmmmnnnt
xxxmmmnnnnt
```

where:

xxx
: indicates the component (for example, DMM for IPCS messages).

mmm
is the module identifier.

nnn or nnnn
is the message number.

t
is the message type.

Examples: Following are some examples of HELP requests.

- To request a HELP file for message DMMTRC731E, issue either:

  HELP MESSAGE DMM731E

  or

  HELP MESSAGE DMMTRC731E

- To request a menu of DUMPSCAN subcommands and help information, issue either:

  HELP

  or

  HELP DUMPSCAN MENU

- To request information about the LOCATE subcommand, issue:

  HELP IPCS LOCATE

Usage Notes:

1. You must enter the *entire* HELP command format to get the correct HELP information on a DUMPSCAN subcommand. For example, enter:

   HELP IPCS TRACE

   to see the DUMPSCAN TRACE subcommand. If you had entered "HELP TRACE," you would see the CP TRACE command information. Note that only eight parameters, including a parenthesis, can be issued on the command line.

2. Enter HELP without parameters to get the DUMPSCAN Menu screen.

Responses:

Displays the DUMPSCAN help menu if just "HELP" is entered. If the subcommand "HELP IPCS subcommand" is entered, then the HELP information for the subcommand is displayed.

Error Messages:

Since the subcommand is passed to the CMS HELP facility, the error messages which may be generated are CMS error messages. See the CMS HELP command description in the *VM/SP CMS Command Reference*.

Sample Output:

See the CMS HELP command description in the *VM/SP CMS Command Reference* for examples of output of the HELP command.

HX Subcommand

Functional Category: Common

Use the HX subcommand to end the session and return to CMS.

HX	

Usage Notes:

END, HX, and QUIT are functionally equivalent. All are implemented to provide compatibility with accepted CMS usage.

Responses:

```
Ready; T=n.nn/n.nn hh:mm.
```

Error Messages:

(none)

Sample Output:

Figure 43 gives an example of the output of the HX subcommand.

```
Ready; T=n.nn/n.nn hh:mm.
```

Figure 43. Sample Output of the HX Subcommand

IPCSMAP Subcommand

Functional Category: Common

Use the IPCSMAP subcommand to add an IPCS map to the dump being viewed.

IPCSMAP	

Usage Notes:

1. The dump must be on your A-disk, and the A-disk must be accessed in read/write mode.

2. You may enter CMS Subset by entering 'CMS' in response to the prompt for the fileid of the IPCS map. If you enter 'NONE', processing continues with no map appended.

   CMS
      Enter CMS subset.

   A null line
      Use the assigned default name for this file.

   QUIT
      Terminate the IPCSMAP command.

   NONE
      Do not process this input map. This option is provided for secondary input maps only and is ignored for other files.

3. Message DMMDSC725R always follows DMMDSC720I. Any response other than 'YES' will terminate the subcommand.

Responses:

```
DMMDSC720I LOAD MAP ALREADY PRESENT
DMMDSC725R DO YOU WISH TO REPLACE IT? ENTER: YES|NO
DMMVAL806R FOR type, ENTER 'FN FT FM' OF THE maptype MAP,
           OR ENTER A NULL LINE, CMS, NONE, OR QUIT.
DMMVAL824I 'type' IPCS MAP 'fn ft fm' APPENDED TO 'PRBnnnnn'.
```

Error Messages:

```
DMMDSC723I THE DUMP IS NOT ON THE A-DISK
DMMDSC863E INVALID OPERAND - operand
DMMVAL100S ERROR 'nnn' READING FILE 'fn ft fm'
DMMVAL200S ERROR 'nnn' WRITING FILE 'fn ft fm'
DMMVAL807I UNABLE TO LOCATE maptype MAP 'fn ft fm'

DMMVAL814E 'type' IPCS MAP NOT APPENDED
DMMVAL820I INSUFFICIENT MAP PROCESSING DISK SPACE
           FOR 'fn ft fm'
DMMVAL821I 'type' IPCS MAP 'fn ft fm' NOT VALID FOR
           DUMP 'PRBnnnnn'
DMMVAL822E IPCSMAP FUNCTION NOT SUPPORTED FOR 'type'
```

Sample Output:

Figure 44 gives an example of the output of the IPCSMAP subcommand.

```
DUMPSCAN 00001
IPCSMAP
DMMVAL806R FOR type, ENTER 'FN FT FM' OF THE IPCS MAP,
           OR ENTER A NULL LINE, CMS, NONE, OR QUIT
<null line>

DMMVAL824I CP IPCS MAP CPIPCS MAP A APPENDED TO PRB00001
*** READY ***    DUMP TYPE = CP
```

Figure 44. Sample Output of the IPCSMAP Subcommand

IUCV Subcommand

Functional Category: GCS dump only

Use the IUCV subcommand to display all entries in the IUCV path table. The IUCV path table contains information about all the IUCV paths in this virtual machine.

IUcv	

Usage Notes:

(none)

Responses:

Displays for each path:

- Owner's id block address
- Exit address
- User word
- Task control block address
- Path status.

Error Messages:

```
CSIIIU31S  Insufficient free storage is available
CSIIIU503I No IUCV PATH table
CSIIIU504I Page 'nnnnnnnn' not found in dump
CSIIIU542I NUCON extension PTR is zero. Can't find IUCV PATH table
CSIIIU543I IUCV anchor block ptr is zero.  Can't find IUCV PATH
           table
CSIIIU544I IUCV PATH table ptr is zero
```

Sample Output:

Figure 45 gives an example of the output of the IUVC subcommand.

```
USER-ID-BLOCK-ADDR  EXIT-ADDR  USER-WORD  TASK-BLOCK-ADDR  PATH-STATUS

HHHHHH              HHHHHH     HHHHHHHH   HHHHHH           HHHH
*** READY ***    DUMP TYPE = GCS
```

Figure 45. Sample Output of the IUCV Subcommand

LOCATE (UP) Subcommand

Functional Category: Common

Use the LOCATE subcommand to search the dump for a particular string of data.

Locate **Locate Up**	$\left\{\begin{array}{l}\textit{string}\\ \textit{X'string}\end{array}\right\}$ *fromhexloc* *tohexloc* $\left[\begin{array}{l}\textit{increment}\\ \underline{\textit{1}}\end{array}\right]$

where:

string
: is up to eight EBCDIC characters without imbedded blanks. If you want imbedded blanks, use x'string.

X'*string*
: is up to 16 hexadecimal digits.

fromhexloc
: is the starting location for the search.

tohexloc
: is the ending location for the search.

increment
: is an optional hexadecimal number to be added to fromhexloc after each match attempt. The permissible range for increment is 1 to 1000 (decimal 4096). If you enter a number greater than 1000, it will be truncated to 1000.

Usage Notes:

1. All input, except the hexadecimal string in the LOCATE subcommand, is truncated to eight bytes. Hexloc-type addresses do not need leading zeros and may have up to six significant digits plus two leading zeros.

2. The LOCATE subcommand may be reissued by pressing the ENTER key (or its equivalent). The current address is placed in "fromhexloc" and the subcommand is then reissued. Use this when you find one occurrence of a string and wish to continue searching for the next occurrence.

   Note: The tohexloc location has not been updated; only the fromhexloc location is updated.

3. Use of the increment operand in the LOCATE subcommand can reduce search time by eliminating unwanted matches. For example, to search a CP trace table for activity on device 0191, starting at location 70000, enter:

   LOCATE X′0191 70002 80000 10

   This checks only the four half-bytes at 70002, 70012, etc., until the upper limit is reached.

4. If the LOCATE command is placed in the &name table, the maximum string of eight characters includes the hexadecimal identifier X'.

5. If LOCATE UP is specified, fromhexloc must be greater than tohexloc. If it is not, error message DMMLOC717I is issued.

6. When a relocate is done, the new "from" address is rounded to a four-word boundary. If the user has specified something other than this for a particular search, it will be lost on the second and subsequent "reuse" operations.

Responses:

A full screen display of the dump data (hexadecimal and translated) is displayed. The current line pointer (line 10) is positioned at the hexloc where 'string' begins.

Error Messages:

```
DMMDSC717E INVALID FORM OF LOCATE COMMAND
DMMDSC863E INVALID OPERAND - operand
DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
DMMLOC708I PAGE 'page' NOT FOUND IN DUMP
DMMLOC715I NON-HEX CHARACTER IN STRING

DMMLOC716I STRING 'string' NOT FOUND
DMMLOC717E INVALID FORM OF LOCATE COMMAND
```

Sample Output:

Figure 46 on page 299 gives an example of the output of the LOCATE (UP) subcommand.

```
LOCATE UP X'0331 77F82 77000 10
    00077EC0  0B000331  000793D8  00079418  00000000     ......1Q..m.....
    00077ED0  10025D58  BE000000  00079B70  0002318C     ..).............
    00077EE0  07025D58  0000000A  00079D98  5001A1EA     ..).........q&...
    00077EF0  07026750  0000000A  00079CD8  5001A1EA     ...&........Q&...
    00077F00  0200287E  0002000C  000C0000  00021D86     ...=...........f
    00077F10  08026750  00048000  80800000  0001A0D6     ...&...........O
    00077F20  0A000000  00026750  070D1000  000200CE     .......&........
==> 00077F30  05000331  0000F9A8  00079448  0C000000 06  ......9y..m.....
    00077F40  10025D58  BE000000  000793D8  0002318C     ..)........1Q....
    00077F50  07025D58  0000000A  00079DF8  5001A1EA     ..).........8&...
    00077F60  03000401  00040002  070D1000  000201A2     ...............s
    00077F70  02074AD6  00020008  000C1000  00003CF2     ..¢O...........2
    00077F80  03000000  00060005  000C2000  00043B2A     ................
    00077F90  07025D58  0000000A  00079DF8  5001A1EA     ..).........8&...
    00077FA0  05000190  00000000  0007A168  0C000000     ................
    00077FB0  10026750  28048000  00079390  00049126     ...&......l...j.
    00077FC0  0201E810  00020008  000C0000  000491E8     ..Y...........jY
    00077FD0  07026750  0000001E  0007A0C8  4001E938     ...&.......H..Z.
    00077FE0  020491E8  0002000C  000C0000  0001E9E2     ..jY..........ZS
*** READY ***     DUMP TYPE = CP
```

Figure 46. Sample Output of LOCATE (UP) Subcommand

MAPA Subcommand

Functional Category: Common

Use the MAPA subcommand to locate the module that contains the address specified.

MAPA	**hexloc**

where:

hexloc
is the hexadecimal address for which the module information is desired.

Usage Notes:

The output displays the name and hexadecimal location of the next lowest entry point and the displacement of hexloc from that entry point address. Also displayed are the name of the module containing that entry point and the displacement.

Hexloc addresses higher than 16 megabytes are invalid.

Responses:

```
'hexloc' is: 'disp' INTO ENTRY  'name' AT 'address'
             'disp' INTO MODULE 'name' AT 'address'
```

Error Messages:

```
DMMDSC863E INVALID OPERAND - operand
DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
DMMMOD706I 'entry' NOT FOUND IN LOAD MAP
DMMMOD708I PAGE 'page' NOT FOUND IN DUMP
DMMMOD718I THIS DUMP HAS NO LOAD MAP
```

Sample Output:

Figure 47 gives an example of the output of the MAPA subcommand.

```
002000 IS: 000005FC INTO ENTRY  DMKMCHSE AT 00001A04
           00000910 INTO MODULE DMKMCH   AT 000016F0
*** READY ***    DUMP TYPE = CP
```

Figure 47. Sample Output of MAPA Subcommand

MAPN Subcommand

Functional Category: Common

Use the MAPN subcommand to search the load map for an entry point.

MAPN	*entrypointname*

where:

entrypointname
is an 8-character entry point.

Usage Notes:

1. If a match is not found on the entire string as entered, the string is truncated by 1 and the search is repeated until a match is found or the string is exhausted.
2. The address in the load map is converted to the real address at the time of the dump.
3. Use of the MAPN subcommand causes a full-screen display on a display terminal. If a 2741-type terminal is in use, the output is limited to the line indicating the address found.

Responses:

```
'entrypointname' MAP - 'address', REAL - 'address'
```

For display terminals there is a full screen display of a page of dump data (hexadecimal and translated). The current line pointer (line 10) is positioned at the entry point address.

Error Messages:

```
DMMDSC863E INVALID OPERAND - operand
DMMMOD706I 'entry' NOT FOUND IN LOAD MAP
DMMMOD707I 'module' PAGE NOT VALID
DMMMOD708I PAGE 'page' NOT FOUND IN DUMP
DMMMOD718I THIS DUMP HAS NO LOAD MAP
```

Sample Output:

Figure 48 on page 302 gives an example of the output of the MAPN subcommand on a display terminal.

```
DMKVDD   MAP - 07C000, REAL - 06A000





==> 0006A000  C4D4D2E5  C4C44040  900BD010  5FC0CA78 04 DMKVDD  ....¬...
    0006A010  D703D00C  D00C9608  D00D1F22  5020D018    P.....o.....&...
    0006A020  4100001C  58F004E8  05EF1841  92404000    .....0.Y....k  .
    0006A030  D2DE4001  400058F0  CA840A08  12224772    K. . ..0.d......
    0006A040  C7849102  D00C4780  C07218AB  9180D00C    Gdj.........j...
    0006A050  4710C6FA  4150C064  9120D00C  4780C106    ..F..&..j.....A.
    0006A060  4550C114  58F0CA88  05EF4770  C88A47F0    .&A..0.h....H..0
    0006A070  C1209180  D00C4710  C6FA9180  B05A4710    A.j.....F.j..!..
    0006A080  C0869640  D00C4150  C0969120  D00C4780    .fo ...&.oj.....
    0006A090  C1064550  C1144550  C7629120  80044710    A..&A..&G.j.....
    0006A0A0  C8A2910C  80064780  C0BA9110  80054710    Hsj.......j.....
    0006A0B0  C8D69120  80054710  C7CC9110  D00C4710    HOj.....G.j.....
*** READY ***    DUMP TYPE = CP
```

Figure 48. Sample Output of MAPN Subcommand

MREGS Subcommand

Functional Category: CP dump only

Use the MREGS subcommand to display:

- Registers
- Clocks
- PSWs
- CSW
- CAW
- Timers

for the main (IPL) processor.

Mregs	

Usage Notes:

(none)

Responses:

The formatted contents of all registers, clocks, PSWs, CSW, CAW and timers in the main or the IPL processor are displayed.

Error Messages:

```
DMMDCP863E INVALID OPERAND - operand
DMMREG724I IPL REGISTERS REQUESTED IN UNIPROCESSOR DUMP
DMMREG708I PAGE 'page' NOT FOUND IN DUMP
```

Sample Output:

The output of the MREGS subcommand is the same as the output of the REGS subcommand with the addition of a heading line indicating that the display is for the IPL processor. See the "REGS Subcommand" on page 310 for an example of the output.

MRIOBLOK Subcommand

Functional Category: CP dump only

Use the MRIOBLOK subcommand to display:

- RCHBLOK
- RCUBLOK
- RDEVBLOK

for the specified device attached to the IPL processor in an MP configuration.

MRIoblok	*raddr*

where:

raddr
: is the hexadecimal device address.

Usage Notes:

The 'raddr' address needs no leading zeros and may be up to four significant digits in length.

Responses:

The contents of the RCHBLOK, RCUBLOK, and RDEVBLOK for device 'raddr' attached to the IPL processor are displayed.

Error Messages:

```
DMMDCP731E OPERAND MISSING OR INVALID
DMMDCP863E INVALID OPERAND - operand
DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
DMMIOB712I DEVICE 'raddr' ADDRESS NOT FOUND
DMMIOB726I IPL RIOBLOKS REQUESTED IN NON-MULTIPROCESSOR DUMP
```

Sample Output:

Figure 49 on page 305 gives an example of the output of the MRIOBLOK subcommand.

```
MRIO   E                ON IPL PROCESSOR
 RCHBLOK CHAN 00XX        ADDRESS 04D7B0
 000 00000040 00210000 0004D7B0 0004D7B0 00000000 10000000 0004D7B0 00000000
 020 00000050 FFFF00A0 00F0FFFF 0140FFFF 0190FFFF 01E0FFFF 0230FFFF 0280FFFF
 040 02D0FFFF 0320FFFF 0370FFFF 03C0FFFF 0410FFFF 0460FFFF FFFFFFFF FFFFFFFF

 RCUBLOK UNIT 000X        ADDRESS 04CC20
 000 00080001 20000000 0004CC20 0004CC20 0004D7B0 00000000 00000000 00000000
 020 0004CC20 00000000 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 0040FFFF
 040 0050FFFF 0060FFFF 00000000 00000000

 RDEVBLOK DEV 000E        ADDRESS 03EFD0
 000 000E0000 20821043 0003EFD0 0003EFD0 0004CC20 00000000 00000000 C1000000
 020 00000000 00000000 0004DB30 00000000 40404040 00CD9410 00000000 00000000
 040 00000000 00000500 00000000 00000000 32030000 00000000 00000000 00000000
 060 00FF0000 00000000 00000000 00000000 00000000 00000000 00000082 00000000
 080 000D0000 00440240 0003F050 0003F050
*** READY ***    DUMP TYPE = CP
```

Figure 49. Sample Output of MRIOBLOK Subcommand

OSPOINT Subcommand

Functional Category: CMS dump only

Use the OSPOINT subcommand to display the formatted contents of three pointers used in OS simulation.

OSPoint	

Usage Notes:

(none)

Responses:

```
CVTSECT= (address of simulated communications vector table)
FCBSECT= (address of first file control block)
OPSECT = (address of reading and writing parameter list)
```

Error Messages:

```
DMMDCM863E INVALID OPERAND - operand
DMMDCO708I PAGE 'page' NOT FOUND IN DUMP
```

Sample Output:

Figure 50 gives an example of the output of the OSPOINT subcommand.

```
CVTSECT= 00002C58
FCBSECT= 00000000
OPSECT = 00003BA0
*** READY ***   DUMP TYPE = CMS
```

Figure 50. Sample Output of OSPOINT Subcommand

PRINT Subcommand

Functional Category: Common

Use the PRINT subcommand to print displayed data.

Print **PRT**	[*subcommand* **ON** **OFF** **CLOSE** **?**]

where:

subcommand
: is the DUMPSCAN subcommand to be issued and its results printed.

ON
: turns PRINT on, to collect data for printing.

OFF
: turns PRINT off, but does not close the virtual printer.

CLOSE
: closes the spool file but does not turn PRINT off.

?
: displays PRINT status (ON or OFF).

Usage Notes:

1. When PRINT is ON, all data displayed on the terminal is also written to the virtual printer.
2. PRINT with no operands reissues the previous subcommand and prints the data.
3. The PRINT subcommand is not allowed in the &NAME table.
4. CLOSE is issued for you at the end of the session.
5. If PRINT is OFF and the 'subcommand' is issued, with or without an operand, PRINT is turned on for that operation, then turned off. If PRINT was on, it is left on. The CLOSE subcommand is not issued.
6. PRINT cannot be an operand of PRINT. You cannot specify:
   - PRINT P
   - PRINT PRT
   - PRINT PRINT
   - PRINT and any other abbreviation of Print.

Responses:

```
*** READY 'type' DUMP ***
PRINT IS ON
PRINT IS OFF
```

Error Messages:

Issued by the subcommand. See the subcommand entered for the exact messages. DUMPSCAN responds DMMDSC733E if the subcommand is not recognized.

Some messages are:

```
DMMDSC733E ERROR DETECTED WHILE PROCESSING THE SUBCOMMAND - subcommand
DMMDSC863E INVALID OPERAND - operand
```

Sample Output:

Figure 51 gives an example of the output of the PRINT subcommand.

```
PRINT ?
PRINT IS OFF
*** READY ***    DUMP TYPE = CP
```

Figure 51. Sample Output of the PRINT Subcommand

QUIT Subcommand

Functional Category: Common

Use the QUIT subcommand to end the session and return to CMS.

QUIT	

Usage Notes:

END, HX, and QUIT are functionally equivalent. All are implemented to provide compatibility with accepted CMS usage.

Responses:

```
Ready; T=n.nn/n.nn hh:mm.
```

Error Messages:

(none)

Sample Output:

Figure 52 gives an example of the output of the QUIT subcommand.

```
Ready; T=n.nn/n.nn hh:mm.
```

Figure 52. Sample Output of the QUIT Subcommand

REGS Subcommand

Functional Category: Common

Use the REGS subcommand to display:

- Registers
- Clocks
- PSWs
- Timers
- CSW
- CAW.

Regs	

Usage Notes:

1. When in AP or MP mode, the data for a CP dump is from the failing processor.

2. For any dump format other than CP, only the general registers are displayed.

3. For AP/MP dumps, an indicator that shows if the registers are associated with an IPL or non-IPL processor, is displayed. However, if absolute page zero (the prefix storage area) is not in the dump the indicator is not displayed. Absolute page zero is required to determine whether the failing processor is the IPL or the non-IPL processor.

Responses:

The registers, clocks, PSWs, CSW, CAW, and timers are formatted and displayed.

Error Messages:

```
DMMDSC863E INVALID OPERAND - operand
DMMREG708I PAGE 'page' NOT FOUND IN DUMP
```

Sample Output:

Figure 53 on page 311 gives an example of the output of the REGS subcommand.

```
GEN REGS 0 - 15
00001DD0 00000000 00000000 000C0000 000C0000 80043B24 00026810 00002522
FFFFFFFF 00004A08 00000000 00026750 00043000 0007F940 400439DC 00000000
CTL REGS 0 - 15
80800CC0 00029600 FFFFFFFF 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 EFC00000 00000000
F/P REGS 0 - 6
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

TOD CLOCK                    CPU TIMER                   CLOCK COMPARATOR
8DB95C10 8CE39000            7FFFFFFF CFF22000           8DB95C10 A97AA000

CSW                          CAW                INT TIMER        PREFIX REG
00079448 0C000000            00079418           00000300           00000000

EXT OLD      0080 030C0000 0001A582             EXT NEW  000C0000 00000950
SVC OLD      0008 000C1000 00003CF2             SVC NEW  000C0000 00000C90
PGM OLD  00060005 000C2000 00043B2A             PGM NEW  000C0000 00002408
MCH OLD           00000000 00000000             MCK NEW  00080000 00001578
I/O OLD      0331 000D0000 0000F9A8             I/O NEW  000C0000 000054C8
*** READY ***    DUMP TYPE = CP
```

Figure 53. Sample Output of REGS Subcommand

RIOBLOK Subcommand

Functional Category: CP dump only

Use the RIOBLOK subcommand to display:

- RCHBLOK
- RCUBLOK
- RDEVBLOK

for the specified 'raddr'.

RIOblok	*raddr*

where:

raddr
is the hexadecimal device address.

Usage Notes:

1. The 'raddr' address needs no leading zeros and may be up to four significant digits in length.

2. In an MP configuration, the display is for device 'raddr' on the failing processor.

Responses:

The contents of the RCHBLOK, RCUBLOK, and RDEVBLOK for device 'raddr' are displayed.

Error Messages:

```
DMMDCP731E OPERAND MISSING OR INVALID
DMMDCP863E INVALID OPERAND - operand
DMMDSC863E INVALID OPERAND - operand
DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
DMMIOB712I DEVICE 'raddr' ADDRESS NOT FOUND
```

Sample Output:

Figure 54 on page 313 gives an example of the output of the RIOBLOK subcommand.

```
RIO 00E
 RCHBLOK CHAN 00XX        ADDRESS 04D7B0
 000 00000040 00210000 0004D7B0 0004D7B0 00000000 10000000 0004D7B0 00000000
 020 00000050 FFFF00A0 00F0FFFF 0140FFFF 0190FFFF 01E0FFFF 0230FFFF 0280FFFF
 040 02D0FFFF 0320FFFF 0370FFFF 03C0FFFF 0410FFFF 0460FFFF FFFFFFFF FFFFFFFF

 RCUBLOK UNIT 000X        ADDRESS 04CC20
 000 00080001 20000000 0004CC20 0004CC20 0004D7B0 00000000 00000000 00000000
 020 0004CC20 00000000 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 0040FFFF
 040 0050FFFF 0060FFFF 00000000 00000000

 RDEVBLOK DEV 000E        ADDRESS 03EFD0
 000 000E0000 20821043 0003EFD0 0003EFD0 0004CC20 00000000 00000000 C1000000
 020 00000000 00000000 0004DB30 00000000 40404040 00CD9410 00000000 00000000
 040 00000000 00000500 00000000 00000000 32030000 00000000 00000000 00000000
 060 00FF0000 00000000 00000000 00000000 00000000 00000000 00000082 00000000
 080 000D0000 00440240 0003F050 0003F050
*** READY ***    DUMP TYPE = CP
```

Figure 54. Sample Output of the RIOBLOK Subcommand

SCROLL Subcommand

Functional Category: Common

The SCROLL subcommand repeats the most recent DISPLAY or TRACE subcommand with an adjusted address. SCROLLU decreases the target address to display the preceding screen of data, while SCROLL increases the target address to display the next screen of data.

Scroll U **ScrollU**	[**HEX**] [**FORMat**]

where:

HEX
displays the trace entries in the short (HEX) version instead of the format used with the previous formatted displays of the trace entries. This operand is valid only when you issue the Scroll subcommand after a formatted display of the trace entries.

FORMat
displays the trace entries in the long (FORMAT) version instead of the format used with the previous formatted displays of the trace entries. This operand is valid only when you issue the Scroll subcommand after a formatted display of the trace entries.

Usage Notes:

1. The SCROLL subcommand is intended for use on a display terminal.

2. The SCROLL subcommand scrolls from the last entry displayed; it displays the following addresses.

3. The SCROLLU subcommand scrolls from the first entry displayed; it displays the preceding addresses.

4. The SCROLL and SCROLLU subcommands may be reissued by pressing the ENTER key (or its equivalent).

Responses:

There is a full-screen display of the dump data (hexadecimal and translated or formatted if following TRACE). The current line pointer (line 10) is positioned X'130' bytes from its previous location.

Error Messages:

```
ATSZTO076E FORMATTED DATA ENTRY EXCEEDS MAXIMUM SIZE
ATSZTS087E ATTEMPT TO GO BEYOND STORAGE BOUNDARY
DMMDSC732E UNRECOGNIZED SUBCOMMAND - subcommand
DMMDSC733E ERROR DETECTED WHILE PROCESSING THE SUBCOMMAND -
           subcommand
DMMSCR708I PAGE 'page' NOT FOUND IN DUMP

DMMSCR709I NO VALID SCROLL ADDRESS
DMMSCR863E INVALID OPERAND - operand
DMMTRD743I TRACE ENTRY IS TOO LARGE TO BE COMPLETELY DISPLAYED
DMMTRD744E COMMAND TERMINATED, CAN NOT PROCESS BEYOND STORAGE
           BOUNDARY
```

Sample Outputs:

The hex and format control options let you change the current TRACE format control. For example, the following is a valid sequence of TRACE subcommands:

TRACE FORMAT
. (display a screen of trace entries, setting the
. FROM address and the FORMAT option)
.
SCROLL
(display the next screen full of formatted (FORMAT)
trace entries)

To change the HEX or FORMAT setting while continuing to scroll, you can issue the SCROLL subcommand and specify HEX or FORMAT. For example:

SCROLL U FORMAT
or
TRACE SCROLL U FORMAT

The SCROLL option of the TRACE subcommand returns control to the trace formatter after using the DISPLAY subcommand to display storage. For example, you might do the following:

- Display trace entries.
- Use the SCROLL subcommand to format and display more trace entries.
- Issue a DISPLAY subcommand to look at storage.
- Issue a SCROLL subcommand.

However, the storage will not be formatted as trace entries even if the displayed area is in the trace table. Since the SCROLL subcommand is not processing trace entries at this time, the FORMAT and HEX operands are not valid operands. The following is an example of an invalid sequence:

```
TRACE FORMAT
 . (displays a screen of trace entries, setting the
 .  FROM address and the FORMAT)
 .
SCROLL U HEX
 . (displays a screen of trace entries with the HEX operand)
 .
DISPLAY 240000
 . (looks at this area of storage in display format)
 .
SCROLL
 . (looks at the next screen of displayed storage)
 .
SCROLL FORMAT
 . (Error message DMMSCR863E, stating invalid parameter)
 .
```

The SCROLL option on the TRACE subcommand, explained in "TRACE Subcommand," lets you return to formatting trace entries after a display of some storage area. The following is similar to the previous example but contains a valid sequence of commands:

```
TRACE FORMAT
 . (to display a screen of trace entries, setting the
 .  FROM address and the FORMAT)
 .
SCROLL U HEX
 . (to display a screen of trace entries with the HEX operand)
 .
DISPLAY 240000
 . (to look at an area of storage in display format)
 .
SCROLL
 . (to look at next screen of displayed storage)
 .
TRACE SCROLL U FORMAT
 . (to look beyond last displayed trace entries and use the FORMAT option)
 .
```

SYMP Subcommand

Functional Category: Common

Use the SYMP subcommand to format and display the summary of the symptom record.

SYMP	

Usage Notes for the REG/PSW Symptom:

The PSW address is compared with the contents of the registers at the time of the failure. The registers are checked in descending order from 15 to 0. The first two registers whose contents are less than, and within 4K of the instruction address in the PSW, are used to develop the symptoms. Each of the two register values is subtracted from the instruction address in the PSW and this 'difference' or 'offset' is the symptom value. Each 'difference' value is saved in the form of REG/rrddd, where:

```
'REG'   is the Structured Data Base (SDB) prefix
'rr'    is the hex register number
'ddd'   is the difference.
```

For example:

When one or two registers meet the criteria, one or two symptoms are created.

```
REGS/0C14E REGS/0A050...  two qualified registers
```

If no registers meet the described criteria, then one REG/PSW 'difference' symptom is generated, where 'rrddd' contains 'FFFFF'.

```
REGS/FFFFF..............  no qualified registers
```

If the address in the failing PSW is less than X'200', there will be only one symptom where X'rr' contains a hex X'FE' and X'ddd' contains the hex address in the PSW.

```
REGS/FE006..............  PSW less than X'200'
```

Responses:

```
----- SYMPTOM RECORD FOR PRBnnnnn -----
DATE OF DUMP: yy/mm/dd
TIME OF DUMP: hh:mm:ss:th
GMT:          hh:mm:ss:th
SYSTEM INFORMATION:
              MODEL    =
              SERIAL   =
              RELEASE  =
              FEATURE  =
              COMPID   =
DUMP TYPE: CPDUMP [or VMDUMP]
COMPONENT INFORMATION:
               RELEASE = n
               FEATURE = nn

-------- SYMPTOM STRING ------------------------------
[ERROR           AB/Sxxxxxx     ]
[            (or AB/Uxxxxxx)    ]
[            (or WS/Dxxxx  )    ]
[            (or WS/Exxxx  )    ]
[FAILING COMPID  PIDS/nnnnxxx00 ]
[FAILING MODULE  RIDS/xxxxxx    ]
 REG/PSW         REGS/rrddd [REGS/rrddd]
-------------------------------------------------------
```

where:

Date of Dump
= The date the dump was processed by the IPCSDUMP command.

Time of Dump
= The time the dump was processed by the IPCSDUMP command.

GMT
= Greenwich Mean Time (time when the dump was created)

MODEL
= CPU Model Number

SERIAL
= CPU Serial Number

RELEASE
= CP Release

FEATURE
= CP Service Level

COMPID
= CP Component ID

The symptom string is shown in Structured Data Base (SDB) format. The following is a list of SDB keywords and their meanings:

	Keyword	
ERROR	AB/Sxxxxxx	abend codes associated with a system abend.
	AB/Uxxxxxx	abend codes associated with a user abend.
	WS/Dxxxx	wait codes associated with a disabled wait.
	WS/Exxxx	wait codes associated with an enabled wait.
COMPID	PIDS/nnnnxxx00	failing component ID (e.g. 5749DMK00 for CP)
MODULE	RIDS/xxxxxx	CSECT name of a failing module.
REG/PSW	REGS/rrddd	The difference between the PSW and the registers.

Error Messages:

```
DMMDSC863E INVALID OPERAND - operand
DMMTIU868I REQUIRED RESOURCES ARE NOT AVAILABLE, SYMPTOM RECORD
           FOR FILE 'PRBnnnnn' CANNOT BE DISPLAYED
DMMTIU869I ERROR 'nnn' OCCURRED WHILE READING SYMPTOM RECORD
           FOR FILE 'PRBnnnnn'
DMMTIU870I SYMPTOM RECORD FOR FILE 'PRBnnnnn' CANNOT BE FOUND
```

Sample Output:

Figure 55 gives an example of the output of the SYMP subcommand.

```
--------------------------- SYMPTOM RECORD FOR PRB00136 ---------------------
DATE OF DUMP:  86/02/22
TIME OF DUMP:  15:26:14:00
GMT:           19:26:14:00
SYSTEM INFORMATION:
              MODEL    =  4381
              SERIAL   =  010511
              RELEASE  =  5
              FEATURE  =  06
              COMPID   =  574900000

DUMP TYPE:                CPDUMP
COMPONENT INFORMATION:
              RELEASE  =  5
              FEATURE  =  06

------------------------------------ SYMPTOM STRING --------------------------
ERROR               AB/SEXT004
FAILING COMPID      PIDS/5749DMK00
REG/PSW             REGS/0C32A REGS/0AE9A
------------------------------------------------------------------------------
*** READY ***     DUMP TYPE = CP
```

Figure 55. Sample Output of the SYMP Subcommand

TACTIVE Subcommand

Functional Category: GCS dump only

Use the TACTIVE subcommand to display the task's active program list.

TACtive	[*taskid* / **ALL**]

where:

taskid
: Identifies the task you want information about. Format is *nnnn*.

ALL
: Requests information for all tasks. ALL is the default.

Usage Notes:

(none)

Responses:

Displays a chart containing the task ID, the address of a task control block, and the task completion code. A state block is a control block that contains information about an active program. There are three types of state blocks:

- Link Blocks represent programs that have been invoked via the LINK, SYNCH, XCTL, or ATTACH Macros, or OSRUN command.
- SVC Blocks represent calls to the SVC interrupt handler.
- Asynchronous Exit Blocks exist for asynchronous exits scheduled on this task.

For every state block on the task's active program list, this subcommand also displays:

- Address of the state block
- Type of state block (Link Block, SVC Block, or Asynchronous Exit Block)
- Name and entry-point address of the program that the block represents
- Register contents associated with the state block.

Error Messages:

```
CSIIAL031S Insufficient free storage is available
CSIIAL504I Page 'nnnnnnnn' not found in dump
CSIIAL505I TASKID 'xxxxxxxx' invalid
CSIIAL545I NUCON extension pointer is zero.  Can't find state block
CSIIAL546I Task block PTR is zero.  Can't find state block

CSIIAL547I State block PTR is zero
CSIIAL548I Taskid table PTR is zero.  Can't find state block
```

Sample Output:

Figure 56 gives an example of the output of the TACTIVE subcommand.

```
TASK-ID  TASK-BLOCK   TASK-COMPLETION   STATE-BLOCK  TYPE   PROGRAM   ENTRY
         ADDRESS      CODE              ADDRESS             NAME      ADDRESS

HHHH    HHHHHH     HHHHHH       HHHHHH     HH    EEEEEEEE  HHHHHH
0001    002348     000000       16B600     40    GDUMP     1870E6
                                R0 =001690D4 R1 =001690F8 R2 =00169310
                                R3 =00000000 R4 =00000590 R5 =00000000
                                R6 =00000678 R7 =00000678 R8 =00000006
                                R9 =002FD000 R10=00169048 R11=00009F38
                                R12=50182F1C R13=00009F38 R14=50183280
                                R15=00000001
                                002518     80    CONSOLE   182EA8
                                R0 =00000000 R1 =00000000 R2 =00000000
                                R3 =00000000 R4 =00000000 R5 =00000000
                                R6 =00000000 R7 =00000000 R8 =00000000
                                R9 =00000000 R10=00000000 R11=00000000
                                R12=00000000 R13=00000000 R14=00000000
                                R15=00000000
TASK    BLOCK    COMPLETION     BLOCK            PROGRAM    ENTRY
 ID    ADDRESS      CODE       ADDRESS    TYPE    NAME     ADDRESS
0002    002430     000000       0025B0     80    COMMAND   182A68
                                R0 =00000000 R1 =00000000 R2 =00000000
                                R3 =00000000 R4 =00000000 R5 =00000000
                                R6 =00000000 R7 =00000000 R8 =00000000
                                R9 =00000000 R10=00000000 R11=00000000
                                R12=00000000 R13=00000000 R14=00000000
                                R15=00000000
*** READY ***     DUMP TYPE = GCS
```

Figure 56. Sample Output of the TACTIVE Subcommand

TLOADL Subcommand

Functional Category: GCS dump only

Use the TLOADL subcommand to display the Task Load List.

TLoadl	[*taskid* / <u>ALL</u>]

where:

taskid
Identifies the task you want information about. Format is *nnnn.*

<u>ALL</u>
Requests information for all tasks. ALL is the default.

Usage Notes:

(none)

Responses:

Displays for each program LOADed by this task:

- Address of the control block that contains information as to where the program is loaded
- Associated program name
- Number of times it has been LOADed, but not DELETEd.

Error Messages:

```
CSIITL031S Insufficient free storage is available
CSIITL504I Page 'nnnnnnnn' not found in dump
CSIIAL505I TASKID 'xxxxxxxx' invalid
CSIITL535I NUCON extension PTR is zero.  Can't find task load list
CSIITL536I TASKID table PTR is zero.  Can't find task load list

CSIITL537I Task block PTR is zero.  Can't find task load list
CSIITL538I Task load list PTR is zero
```

Sample Output:

Figure 57 gives an example of the output of the TLOADL subcommand.

```
TASK-ID  TASK-BLOCK-ADDRESS  LOAD-BLOCK-ADDRESS  PROGRAM-NAME  LOAD-COUNT

HHHH     HHHHHH              HHHHHH              EEEEEEEE      HHHH
 .          .                   .                   .             .
 .          .                   .                   .             .
 .          .                   .                   .             .
*** READY ***     DUMP TYPE = GCS
```

Figure 57. Sample Output of the TLOADL Subcommand

TRACE Subcommand

Functional Category: CP and TSAF dumps only

Use the TRACE subcommand to display trace table entries in either a hexadecimal display or in a formatted display.

After the first invocation of the TRACE subcommand, you may specify either SCROLL option to move forward or backward through the trace table, using the formatting options that you established with an earlier TRACE subcommand. You can also issue the REUSE subcommand.

The format of the TRACE subcommand is:

Trace	[[[**FOR**] *count*] [**FROM** *fromloc*] [**Scroll** [**U**] / **ScrollU**]]	[**HEX** / **FORMat**]

where:

FOR
: is an optional keyword operand. If you specify it, the count operand **must** follow immediately. Do not combine this with the SCROLL or SCROLLU operand.

count
: is a decimal number from 1 to 999 that specifies the number of entries that will be displayed in the trace table. If you specify the FORMat keyword operand, the default value will be one screen of data. If you specify the HEX keyword operand (or a default to HEX), the default value will be twenty minus the number of lines used to display any error messages (one screen of data). The default values are set so that the displayed output will be a full screen of data on a standard 24-line display terminal. Do not combine this with the SCROLL or SCROLLU operand.

FROM
: is a keyword operand. If you specify it, the fromloc operand **must** follow immediately. Do not combine this with the SCROLL or SCROLLU operand.

fromloc
: is a hexadecimal address from which the trace entries are displayed. You must specify a location that is on a 16-byte boundary. The default value is the address of the most recent trace table entry.

HEX
: is an optional keyword operand. If you specify it, unformatted trace table entries will be displayed in hexadecimal, along with a brief description of the entry. Note that HEX is the default the first time you enter TRACE in a DUMPSCAN session. Thereafter, it defaults to the previous formatting keyword (HEX or FORMat).

FORMat
: is an optional keyword operand. If you specify it, formatted trace table entries will be displayed, otherwise the default value of HEX is assumed.

Scroll
: is an optional keyword operand. If you specify it, the next screen full of trace entries will be displayed. If you specify SCROLL U, it will have the same affect as specifying SCROLLU.

  This option is only valid if you already issued a successful TRACE subcommand. Do not combine this option with the FROM or FOR options.

ScrollU
: is an optional keyword operand. If you specify it, the preceding screen full of trace entries will be displayed.

  This option is only valid if you already issued a successful TRACE subcommand. Do not combine this option with the FROM or FOR options.

Usage Note:

The formatted output produced by the TRACE subcommand is valid for VM/SP Release 5 dumps only. Unpredictable results can occur in the formatted output if this function is used against a pre-VM/SP or pre-VM/SP HPO Release 5 dump. The invoker should rely only on the hexadecimal TRACE data and not the extra formatted descriptions or the brief descriptions that are given with the hexadecimal data.

You may specify the operands in any order except for the two cases previously explained.

Even if the trace table pointers are invalid, use of the FROM operand and location allows you to view the trace table.

Responses:

See "Sample Output of Unformatted Hexadecimal CP Display" on page 327, "Sample Output of Unformatted Hexadecimal TSAF Display" on page 328, "Sample Output of Formatted CP Display" on page 328, and "Sample Output of Formatted TSAF Display" on page 330.

Error Detection:

The TRACE subcommand detects three types of errors:

1. Invocation errors
2. Errors in the dump
3. Necessary resources that are not available.

Errors in the invocation may consist of:

1. Conflicting operands.

2. Missing operands.

3. Non-hexadecimal characters in a hexadecimal operand.

4. If the "FROM" address is outside of the range of addresses indicated in the valid trace table pointers.

5. For CP, FROM address is not on a 16-byte boundary. CP rounds down the address to point to a 16-byte address. For TSAF, the address must be at the beginning of a valid TSAF trace entry. Otherwise, the Trace subcommand will fail.

6. Invalid count operand (not within the range of 1 to 999).

Errors that are detected in the dump are:

1. Invalid trace table pointers:

   a. Trace table start pointer is zero (CP only).
   b. Trace table start pointer is greater than the trace table stop pointer.
   c. Trace table current pointer points outside of the trace table.
   d. Trace table start or trace table stop pointer is not on a page boundary.
   e. Trace table current pointer is not on a 16-byte boundary. (CP only)
   f. The trace table is less than one page in size. (CP only) For TSAF, the trace table must be full pages.
   g. The map is not appended (TSAF only).
   h. The pointers do not point to valid trace entries (TSAF only).

2. A page that is to be used for the trace function is not in the dump. The page can be either the page containing the pointers, or the page containing the trace entries.

Error Messages:

```
| ATSZTF091E DATA FIELD OVERLAPS TRAILER RECORD
| ATSZTO076E FORMATTED DATA ENTRY EXCEEDS MAXIMUM SIZE
| ATSZTR075E NON-NUMERIC COUNT CHARACTER - RETRY
| ATSZTR077E CONFLICTING OPERAND - operand
| ATSZTR087E ATTEMPT TO GO BEYOND STORAGE BOUNDARY

| ATSZTR078E OPERAND MISSING OR INVALID
| ATSZTS079I TRACE TABLE POINTERS INVALID:
|            START = start      END = end       CURRENT = current
| ATSZTS080I "FROM" LOCATION OUTSIDE OF TRACE TABLE RANGE: fromloc
|            START = start      END = end       CURRENT = current
| ATSZTS081E "FROM" LOCATION NOT A VALID TRACE ENTRY: fromloc
| ATSZTS082E INVALID TRACE ENTRY FOUND AT addr

| ATSZTS083E REQUIRED RESOURCES NOT AVAILABLE
| ATSZTS084I PAGE 'page' NOT FOUND IN DUMP
| ATSZTS086E NO TRACE ENTRIES FOUND - addr
| ATSZTS088E UNABLE TO LOCATE TRACE TABLE POINTERS
| ATSZTS092I TRACE ENTRY SEARCH STOPPED AT addr1
|             TO SEARCH TO LOWER DUMP ADDRESSES, TRY ADDRESS addr2
|             TO SEARCH TO HIGHER DUMP ADDRESSES, TRY (ADDRESS addr3
|             | "SCROLL")

| ATSZTS093I POSSIBLE TRACE ENTRY AT addr
|            USE THE "FROM" OPERAND TO DISPLAY THE ENTRY
  DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
| DMMTRC708I PAGE 'page' NOT FOUND IN DUMP
  DMMTRC730E CONFLICTING OPERAND - operand
  DMMTRC731E OPERAND MISSING OR INVALID

  DMMTRC741I "FROM" LOCATION OUTSIDE OF TRACE TABLE:
             START = start  END = end  CURRENT = current
  DMMTRC742I "FROM" LOCATION NOT ON A 16 BYTE BOUNDARY
  DMMTRC868E REQUIRED RESOURCES NOT AVAILABLE
| DMMTRD708I PAGE 'page' NOT FOUND IN DUMP
| DMMTRD740I TRACE TABLE POINTERS INVALID:
|            START = start  END = end  CURRENT = current

| DMMTRD741E "FROM" LOCATION OUTSIDE OF TRACE TABLE:
|            START = start  END = end  CURRENT = current
| DMMTRD743I TRACE ENTRY IS TOO LARGE TO BE COMPLETELY DISPLAYED
| DMMTRD744E COMMAND TERMINATED, CAN NOT ACCESS BEYOND STORAGE BOUNDARY
```

Sample Output of Unformatted Hexadecimal CP Display:

If the following command is issued:

TRACE 4 HEX

it indicates that the last four entries in the trace table should be displayed. If address X'CCB230' is the most recent trace table entry, the following output will be produced:

```
00CCB200    08FEBCC8 800C4202 03800000 0000F438    ENTER SCHEDULER
00CCB210    09FEBCC8 0000F1AB 084000BC 00F908FC    QUEUE DROP
00CCB220    03000000 00040040 000C2000 0002B8D2    PROGRAM INTERRUPT
00CCB230    02016F04 00020008 000C0000 00018D66    SVC INTERRUPT (CALL)
*** READY ***     DUMP TYPE = CP
```

Figure 58. Sample Output of Unformatted Hexadecimal CP Display

Issuing the following command produces the same output:

TRACE FOR 4 FROM CCB230

The hexadecimal display consists of:

- Trace entry address (first column)
- Hexadecimal display of the trace entry (second through fifth columns)
- Trace entry description (last column).

Sample Output of Unformatted Hexadecimal TSAF Display:

If the following command is issued:

TRACE 8 HEX

it indicates that the last seven entries in the TSAF trace table should be displayed. If address X'D8EF' is the most recent trace table entry, the following output will be produced:

```
0000D82D    9AF3D043 1B168000 D3F1E3 A572 0008     ATSL1T EXIT
0000D846    9AF3D043 1B1D9000 E5E2D2 C00C 0029     DISPATCH A TASK
0000D880    9AF3D043 1B1F2000 D3D4D5 A168 0005     WOKE UP IN STATE
0000D896    9AF3D043 1B21D000 D4E3E8 60E1 0004     MODULE ENTERED AT ATSMTY
0000D8AB    9AF3D043 1B225000 D4E3E8 60E8 0005     TRACE INPUT
0000D8C1    9AF3D043 1B234000 D4E3E8 60E6 0008     ATSMTY EXIT
0000D8DA    9AF3D043 1B298000 D3F1D7 A461 0004     MODULE ENTERED AT ATSL1P
0000D8EF    9AF3D043 1B29D000 D3F1D7 A464 0005     LINK PURGED
*** READY ***    DUMP TYPE = TSAF
```

Figure 59. Sample Output of Unformatted Hexadecimal TSAF Display

Issuing the following command produces the same output:

TRACE FOR 8 FROM D8EF HEX

The hexadecimal display consists of:

- Trace entry address (first column)
- Hexadecimal display of the trace entry trailer record (middle column)
- Trace entry description (last column).

Sample Output of Formatted CP Display:

If the following command is issued:

TRACE 4 FORMAT

it indicates that four entries in the trace table should be displayed beginning with the most recent trace table entry.

If address X'CCB230' is the most recent trace entry, the following output will be displayed:

```
08 ENTER SCHEDULER   **MP** 08FEBCC8 800C4202 03800000 0000F438  ADDR 00CCB200
        VMBLOK = FEBCC8 = (STEVEG  )
        VMRSTAT = 80 = CF WAIT
        VMDSTAT = 0C = RUNNABLE + IN Q
        VMOSTAT = 42 = SHARED SYSTEM + REAL CF
        VMQSTAT = 02 = CF READ
        VMQLEVEL = 03 = Q3 OR DROP Q1 + USING > FAIR SHARE
        VMTLEVEL = 80 = VIRTUAL RUN
        VMIOINT = 0000
        VMPEND = 00
        ENTERED FROM 00F438 DMKDSP+X'0F56'

09 QUEUE DROP        **MP** 09FEBCC8 0000F1AB 084000BC 00F908FC  ADDR 00CCB210
        VMBLOK = FEBCC8 = (SHIRLEY )
        VMEPRIOR = 0000F1AB, VMUPRIOR = 40, VMQPRIOR = 00BC
        VMUHS = 00F908FC
        DMKSCHAL = 08 = RUNNABLE

03 PROGRAM INTERRUPT **MP** 03000000 00040040 000C2000 0002B8D2  ADDR 00CCB220
        ILC = 04, CODE = 0040 = MON CALL
        OLD PSW = 000C2000 0002B8D2, CP LOCATION 02B8D2 DMKSCH+X'109A'

02 SVC INTERRUPT     **MP** 02016F04 00020008 000C0000 00018D66  ADDR 00CCB230
        CALL TO   016F04 DMKIOS+X'04D4' FROM 018D66 DMKIOT+X'0846'
*** READY ***     DUMP TYPE = CP
```

Figure 60. Sample Output of Formatted CP Display

Issuing the following command produces the same output:

TRACE FOR 4 FROM CCB230 FORMAT

Note: This example formats the same input and the same trace table entries as Figure 58 on page 327 (the example of the HEX operand's displayed output).

This illustrates the differences between the two types of output displayed by the two different operands of the TRACE subcommand: HEX and FORMAT.

The **formatted display** consists of the following:

- Main line containing:
  - Trace type number
  - Trace entry description
  - Main processor, Attached processor, or Microcode generated entry indicator
    - **MP** main processor
    - **AP** attached processor (non-IPL) processor
    - MP/ECPS microcode assist on the main processor
    - AP/ECPS microcode assist on the attached processor
  - Hexadecimal display of the trace table entry
  - Trace table address.

- One or more additional lines containing formatted information (with descriptors) from the trace entry.

In Figure 60 on page 329, the main line shows that:

08	is the trace type number
ENTER SCHEDULER	is the trace entry description
MP	is main processor
08FEB...F438	is the HEX display of the trace table entry
ADDR 00CCB200	is the trace table address

Sample Output of Formatted TSAF Display:

If the following command is issued:

TRACE 3 FORMAT

it indicates that two entries in the trace table should be displayed beginning with the most recent trace table entry.

If address X'D8EF' is the most recent trace entry, the following output will be displayed:

```
60E6 ATSMTY EXIT                                          ADDR = 0000D8C1
       CLOCK = 9AF3D043 1B234000             MODULE = ATSMTY
       TIME = 08:43:08.627508 GMT 05/20/1986  LENGTH = 0008
       R14              6002F8E4                          *-.8U*
       R15              0004AB30                          *....*

A461 MODULE ENTERED AT ATSL1P                             ADDR = 0000D8DA
       CLOCK = 9AF3D043 1B298000             MODULE = ATSL1P
       TIME = 08:43:08.627608 GMT 05/20/1986  LENGTH = 0004
       PARM_LIST        000300A4                          *...u*

A464 LINK PURGED                                          ADDR = 0000D8EF
       CLOCK = 9AF3D043 1B29D000             MODULE = ATSL1P
       TIME = 08:43:08.627613 GMT 05/20/1986  LENGTH = 0005
       LINK_NUMBER      00000004                          *....*
*** READY ***    DUMP TYPE = TSAF
```

Figure 61. Sample Output of Formatted TSAF Display

Issuing the following command produces the same output:

TRACE FOR 3 FROM D8EF FORMAT

The **formatted display** consists of the following:

- Main line containing trace:
  - Type number
  - Entry description
  - Table address
- Two lines containing:
  - Time stamp for trace entry
  - Module that produced the trace entry
  - Time stamp in readable format
  - Length of the data files (in hex)
- Zero or more additional lines containing formatted information (with descriptors) from the trace entry.

TSAB Subcommand

Functional Category: GCS dump only

Use the TSAB subcommand to display the subpool map and chain header of a task.

TSab	[*taskid* / <u>**ALL**</u>]

where:

taskid
: Identifies the task you want information about. Format is *nnnn*.

<u>ALL</u>
: Requests information for all tasks. ALL is the default.

Usage Notes:

(none)

Responses:

Displays the:

- Task block address
- Task storage anchor block address
- Chain header of the storage owned by the task
- 256-bit map of the subpools owned by the task.

Error Messages:

```
CSIITA031  Insufficient free storage is available
CSIITA504I Page 'nnnnnnnn' not found in dump
CSIIAL505I TASKID 'xxxxxxxx' invalid
CSIITA539I NUCON extension PTR is zero. Can't find task storage
           anchor block
CSIITA540I TASKID table PTR is zero. Can't find task storage
           anchor block

CSIITA541I Task block PTR is zero. Can't find task storage
           anchor block
CSIITA534I Task storage anchor block PTR is zero
```

Sample Output:

Figure 62 gives an example of the output of the TSAB subcommand.

```
TASK-ID   TASK-BLOCK-ADDRESS   TASK-STORAGE-ANCHOR-BLOCK-ADDRESS   CHAIN-HEADER

HHHH      HHHHHH               HHHHHH                              HHHHHH

 SUBPOOL-MAP: (CONSISTING OF 64 HEX DIGITS)
*** READY ***    DUMP TYPE = GCS
```

Figure 62. Sample Output of the TSAB Subcommand

The first 32 bytes of the TSAB contains the 256 bit map of the subpools owned by the task.

USERMAP Subcommand

Functional Category: CMS dumps only

Use the USERMAP subcommand to add a user load map to the dump being viewed.

USERMAP	

Usage Notes:

1. The dump must be on your A-disk accessible in read/write mode.
2. You are prompted for the correct fileid of your load map.
3. You may enter CMS subset by entering 'CMS' in response to the prompt message.

Responses:

```
DMMMAP802I FOR maptype THE MAP NAMED 'fn ft fm' WAS ADDED
           TO THE IPCS MAP
DMMMAP806R FOR type, ENTER 'FN FT FM' OF THE maptype MAP,
           OR ENTER A NULL LINE, CMS, NONE, OR QUIT
```

Error Messages:

```
DMMDCM718I THIS DUMP HAS NO LOAD MAP
DMMDCM723I THE DUMP IS NOT ON THE A-DISK
DMMDCM856I UNABLE TO LOCATE 'type' 'routine' ROUTINE 'name'
DMMDCM863E INVALID OPERAND - operand
DMMMAP100S ERROR 'nnn' READING FILE 'fn ft fm'

DMMMAP200S ERROR 'nnn' WRITING FILE 'fn ft fm'
DMMMAP801I maptype MAP 'fn ft fm' IS NOT VALID
DMMMAP807I UNABLE TO LOCATE maptype MAP 'fn ft fm'
DMMMAP810I FORMAT OF maptype MAP 'fn ft fm' IS INVALID
DMMMAP815E PROCESSING ERROR IN 'type' MAP ROUTINE 'name'

DMMMAP817E maptype MAP 'fn ft fm' OVERLAPS A PREVIOUS MAP
DMMMAP818E INPUT MAP LIMIT REACHED FOR 'type' IPCS
           MAP 'fn ft fm'
```

Sample Output:

Figure 63 on page 335 gives an example of the output of the USERMAP subcommand.

```
FOR type, ENTER 'fn ft fm' OF THE maptype MAP,
          OR ENTER A NULL LINE, CMS, NONE, OR QUIT

user map1 a

CMS IPCS MAP user map1 a ADDED TO IPCS MAP
*** READY ***    DUMP TYPE = CMS
```

Figure 63. Sample Output of the USERMAP Subcommand

VIOBLOK Subcommand

Functional Category: CP dump only

Use the VIOBLOK subcommand to display the VCHBLOK, VCUBLOK, and VDEVBLOK for the specified cuu and userid.

VIOblok	*cuu*	[*userid* **OPERATOR**]

where:

cuu
: is the hexadecimal device address

userid
: is the logon identification. If the userid is not specified, the default is the system operator userid (i.e. OPERATOR). This is true only the first time you issue the VIOBLOK subcommand in a DUMPSCAN session.

Usage Notes:

1. Once userid has been entered, it becomes the default for this DUMPSCAN session. Unless a new userid is required, it need not be reentered.

2. The cuu address needs no leading zeros and may be up to three significant digits in length.

Responses:

The contents of the VCHBLOK, VCUBLOK, and VDEVBLOK for device 'cuu' for user 'userid' are displayed.

Error Messages:

```
DMMDCP731E OPERAND MISSING OR INVALID
DMMDCP863E INVALID OPERAND - operand
DMMHEX714I NON-HEX CHARACTER IN INPUT - RETRY
DMMIOB712I DEVICE 'cuu' NOT FOUND
DMMIOB713I USER 'USERID' VMBLOK NOT FOUND
```

Sample Output:

Figure 64 gives an example of the output of the VIOBLOK subcommand issued for device "00E."

```
VIO 00E OPERATOR
VCHBLOK CHAN 00XX     ADDRESS 3AE4E0
000 00008000 00000000 0000FFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
020 FFFFFFFF FFFFFFFF 00000FFF 00052788

VCUBLOK UNIT 000X     ADDRESS 3EAC78
000 00000040 00000000 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFF0000 FFFFFFFF
020 006000C0 0120FFFF

VDEVBLOK DEV 000E     ADDRESS 369368
000 000E0000 10410000 00000000 00000000 00000000 00000000 00000000 00000000
020 C1000000 00010000 00000000 00035250 00000000 00000000 00000000 00000000
040 00000000 00000000 00000000 00000000
*** READY ***    DUMP TYPE = CP
```

Figure 64. Sample Output of VIOBLOK Subcommand for Device "00E"

VMBLOK Subcommand

Functional Category: CP dump only

Use the VMBLOK subcommand to display information relating to VMBLOKs.

Vmblok	[*userid* / **SYSSPOOL**]

where:

userid
: is the single userid for which information is desired.

SYSSPOOL
: used to specify VMBLOKs for SYSSPOOL to access virtual SYSSPOOL's storage (this operand applies to **VM/SP HPO only**).

Usage Notes:

1. Use the VMBLOK subcommand with no operand to scan the VMBLOK chain and display userid, VMBLOK address and four status bytes (VMRSTAT, VMDSTAT, VMOSTAT, and VMQSTAT) for all logged on users. Display formatted information from a specified user's VMBLOK by specifying the userid operand.

2. For **VM/SP HPO only**, display formatted information from the SYSSPOOL VMBLOK by specifying the SYSSPOOL operand. For more information about the SYSSPOOL operand, see *VM/SP HPO CP for System Programming.*

Responses:

If userid or SYSSPOOL is entered as a parameter, the output is in the following format:

```
VM USERID = eeeeeeee, VMBLOK = hhhhhhhh
USER'S CONTROL REG 2 = hhhhhhhh     (IF EC MODE PSW SHOWN)
BC MODE PSW = hhhhhhhh hhhhhhhh     (OR: EC MODE PSW = )
VMRSTAT = NO STATUS BITS ON         (OR INTERPRETED BITS)
VMDSTAT = NO STATUS BITS ON                   "
VMOSTAT = NO STATUS BITS ON                   "
VMQSTAT = NO STATUS BITS ON                   "
VMPSTAT = NO STATUS BITS ON                   "
VMESTAT = NO STATUS BITS ON                   "
VMTRCTL = NO STATUS BITS ON                   "
VMQLEVEL = NO STATUS BITS ON                  "
VMTLEVEL = NO STATUS BITS ON                  "
VMSWSTAT = NO STATUS BITS ON                  "
VMSWPFL1 = NO STATUS BITS ON                  "
VMSWPFL2 = NO STATUS BITS ON                  "
VMCLEVEL CLASS(ES) = e/e
VIRTUAL INTERRUPTS PENDING = NONE   (OR INTERPRETED BITS)
VM PRIV OR TRACE INSTRUCTION = hhhhhhhh
LAST SIO VIRTUAL ADDRESS WAS: hhhh  (IF LAST PRIV IS SIO)
GPRS 0-3 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
GPRS 4-7 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
GPRS 8-B hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
GPRS C-F hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
LAST USER ISSUED COMMAND WAS: eeeeeeee
```

where:

e
: represents EBCDIC data.

h
: represents hexadecimal data.

Error Messages:

```
DMMDCP863E INVALID OPERAND - operand
DMMVMB711I LOOP IN VMBLOK CHAIN
DMMVMB713I USER 'userid' VMBLOK NOT FOUND
```

Sample Output:

Figure 65 gives an example of the VMBLOK subcommand output when issued with no operands. Figure 66 on page 340 gives an example of the VMBLOK subcommand output when issued with the OPERATOR operand.

Note: The fields VMSWSTAT, VMSWPFL1, AND VMSWPFL2 are displayed only for dumps from systems with Enhanced Paging Subsystem support of VM/HPO.

```
USERID    ADDRESS     VMRSTAT     USERID    ADDRESS     VMRSTAT
SYSTEM    00030FB0    BE000000    OPERATOR  00035250    E8004280
LOGON037  002C4458    68044000    LOGON044  002E63B0    68044080
PROJECT1  00370578    28005000    RSCS      000F2D68    68004080
DIRECTOR  000B64C8    11004000
*** READY ***      DUMP TYPE = CP
```

Figure 65. Sample Output of VMBLOK Subcommand (No Operand)

```
VM USERID = OPERATOR, VMBLOK = 00035250
BC MODE PSW = 00040000 8000F082

VMRSTAT = CFWAIT/PAGE WAIT/IOBWAIT/INST SIM WAIT/
VMDSTAT = NO STATUS BITS ON
VMOSTAT = SYS OPERATOR/SHARED SEG/REAL CF EXEC/
VMQSTAT = Q1 ELIG/
VMPSTAT = NONSHR SAV SYS/
VMESTAT = WANTS SVC ASSIST/BAD SHAD PG TABLES/
VMTRCTL = NO STATUS BITS ON
VMQLEVEL = NO STATUS BITS ON
VMTLEVEL = VIRT TIMER ON/
VMSWSTAT = NO STATUS BITS ON
VMSWPFL1 = WS PAGES TO Q2/ TRIM PAGES FLUSH LIFO
VMSWPFL2 = NO STATUS BITS ON
VMCMDLEV = ALL CLASS USER
VIRTUAL INTERRUPTS PENDING = I-O/
VM PRIV OR TRACE INSTRUCTION = 83E3010C
GPRS 0-3 00000200 00007000 000013A0 0000C7A8
GPRS 4-7 04100BC2 00012230 00000000 04100002
GPRS 8-B 00000001 00000001 00002940 000015F0
GPRS C-F 5000EFAC 00002A50 00000191 00000000
LAST USER ISSUED COMMAND WAS: IPL
*** READY ***    DUMP TYPE = CP
```

Figure 66. Sample Output of VMBLOK Subcommand (OPERATOR Operand)

VMLOADL Subcommand

Functional Category: GCS dump only

Use the VMLOADL subcommand to display information about all programs currently loaded in this virtual machine.

VMLoadl	

Usage Notes:

(none)

Responses:

Displays for each module loaded in this virtual machine:

- Address of the control block containing related information
- Associated program name
- Program address
- Program size
- Entry point address.

For an ALIAS or IDENTIFYed entry point, this subcommand displays:

- Address of the control block containing related information
- Entry point name
- Entry point address
- Type of control block (alias or identify).

Error Messages:

```
CSIIVL504I Page 'nnnnnnnn' not found in dump
CSIIVL533I Virtual machine load list is empty
```

Sample Output:

Figure 67 gives an example of the output of the VMLOADL subcommand.

```
MAJOR-NUCCBLK    MOD-NAME     MOD-ENTRY-ADDR   MOD-SIZE   MOD-ADDR

      HHHHHH     EEEEEEEE     HHHHHH           HHHH       HHHHHH

MINOR-NUCCBLK    ENTRY-NAME   ENTRY-ADDRESS    TYPE

      HHHHHH     EEEEEEEE     HHHHHH           EEEEEEEE
         .           .           .                 .
         .           .           .                 .
         .           .           .                 .
*** READY ***     DUMP TYPE = GCS
```

Figure 67. Sample Output of the VMLOADL Subcommand

Appendix B. DUMPSCAN Scroll Interface

The DUMPSCAN scroll interface for special scrolling supports invocation of format routines. This interface is intended for use by developers of diagnostic routines supported by IPCS. With this support, you have access to seven IPCS variables. You can set these variables with functions (e.g. TRACE subcommand). When you issue a SCROLL, SCROLLU, or REUSE subcommand following one of these functions, IPCS must perform a special scroll. It does this by invoking the routine indicated in the appropriate variable: REUSEAD, SCROLAD, or SCROLUAD.

The variables are listed in the following table:

Variable	Size	Use
HEXAD	Fullword	Contains the first address of a previous display.
SCROLLEN	Fullword	Contains the last address of a previous display.
FEDFEXSW	One byte	Contains the scroll switch, set to character "S". IPCS will invoke one of the special scroll routines when you issue the SCROLL, SCROLLU, or REUSE subcommands.
REUSEAD	Fullword	Contains the address of the routine for the REUSE subcommand.
SCROLAD	Fullword	Contains the address of the routine for the SCROLL subcommand.
SCROLUAD	Fullword	Contains the address of the routine for the SCROLLU subcommand.
PRINTONE	One byte	When set to X'0F', indicates the output of the previous subcommand should be redisplayed.

The SCROLL function calls a routine to perform special scrolling. IPCS calls this routine the same way that it calls all routines that DMMDSC (the main DUMPSCAN processing module) invokes. When IPCS invokes the format routine, register two will point to the next parameter in the tokenized command parameter list. Register three points to the nontokenized command parameter list.

- If you did not enter any parameters, register two will point at the fence, X'FFFFFFFFFFFFFFFF'. Register three points to the nontokenized command parameter list.

- If IPCS finds a form of the SCROLL subcommand with the "U" parameter, register two will point past the "U" parameter in the parameter list. Register 3 will point past the "U" parameter in the nontokenized parameter list. IPCS will invoke the SCROLLU processing routine since the "U" parameter indicates that SCROLLU should be performed.

Appendix C. IPCS SVC 199 Services

SVC 199 is the Interactive Problem Control System (IPCS) communication facility that provides the interface for subsystem support within IPCS. These SVC services may be used by any code written to change IPCS. SVC 199 services can be used by:

- Extraction routines for IPCSDUMP
- Formatting routines for PRTDUMP
- Component-unique subcommands for DUMPSCAN
- Component-unique processing routines for MAP.

Not all SVC 199 services are available to all IPCS commands. Figure 68 shows the SVC 199 codes available to each command.

Command	Codes												
	10	20	30	31	40	41	50	60	70	71	80	90	91
DUMPSCAN			•	•	•	•	•	•				•	•
IPCSDUMP	•	•	•	•	•	•	•	•	•	•	•	•	•
MAP			•	•	•	•	•	•					
PRTDUMP			•	•	•	•	•	•	•	•		•	•

Figure 68. SVC 199 Codes Available to IPCS Commands

There are thirteen codes associated with SVC 199 under IPCS:

- **Code = 10**: Send the keyword symptom data to IPCSDUMP for inclusion in the problem report. IPCSDUMP prompts the user for all blank data areas for which there is a "keyword=" entry.

```
PLIST   DS    OF
        DC    AL4(*-*)    ADDRESS OF KEYWORD LIST
        DC    H'10'       CODE FIELD
        DC    H'0'        NUMBER OF ENTRIES IN LIST
```

The keyworded list must comply with the following rules:

- All data will be presented in printable format.
- Total number of keywords will not exceed 15.
- Minimum number will be 4 (mandatory).
- Total length of keyword plus data, including a suffix of 2 blanks, will be 20 characters.

```
e.g.  KEYWD1=DATA1234567BB

      12345678901234567890    A total of 20 characters
                              B=blank
```

Return codes are:

```
R15=0   ALL OK
   =4   NUMBER OF ENTRIES INVALID
   =8   CODE INVALID
```

Note: Return code 4 is set if less than 4 or more than 15 keywords are sent.

- **Code = 20**: Send additional data to IPCSDUMP for inclusion in the problem report.

```
PLIST  DS   OF
       DC   AL4(*-*)   ADDRESS OF DATA LIST
       DC   H'20'      CODE FIELD
       DC   H'0'       NUMBER OF 80 BYTE ENTRIES
```

Note: This area allows inclusion of information that can assist in further isolating failures. It can contain pertinent register sets, messages, a control block, etc. This information appears in the text description area of the problem report.

The data list must comply with the following rules:

- All data must be in printable format.
- Entries will be 80-bytes long, including spaces.
- Maximum of 15 entries.

Return codes are:

```
R15=0   ALL OK
   =4   NUMBER OF ENTRIES EXCEED 15
   =8   CODE INVALID
```

- **Code = 30**: Request a work buffer.

```
PLIST  DS   OF
       DC   AL4(*-*)   ADDRESS OF BUFFER RETURNED TO CALLER
       DC   H'30'      CODE FIELD
       DC   H'0'       NUMBER OF BUFFERS REQUESTED
```

Note: The caller needs a work buffer. Up to six 4K buffers may be requested. The request is denied if all space asked for cannot be provided. Buffers are on page boundaries and are contiguous.

Return codes are:

```
R15=0   ALL OK
   =4   REQUEST DENIED
   =8   CODE INVALID
```

- **Code = 31**: Return a work buffer.

```
PLIST  DS   0F
       DC   AL4(*-*)   ADDRESS OF BUFFER(S) TO BE RETURNED
       DC   H'31'      CODE FIELD
       DC   H'0'       NUMBER OF 4K BUFFERS TO BE RETURNED
```

Returns storage previously obtained with SVC subcode 30.

Return codes are:

```
R15=0   ALL OK
   =4   ADDRESS INVALID
   =8   CODE INVALID
```

- **Code = 40**: Request data from an address.

```
PLIST  DS   0F
       DC   AL4(*-*)   ADDRESS OF DATA WANTED
       DC   H'40'      CODE FIELD
       DC   H'0'       NUMBER OF BYTES READ CONTIGUOUS TO
                       THE ADDRESS REQUESTED AND THE END OF
                       OF A 12K BUFFER.
       DC   AL4(*-*)   ADDRESS OF 12K BUFFER RETURNED TO
                       CALLER THAT CONTAINS THE ADDRESS OF
                       THE DESIRED DATA
```

The address of the data requested is the first entry in the buffer returned. The buffer will vary in length if the next page of the dump was not dumped to the page in which address requested was found. The last halfword of the PLIST contains the total number of consecutive bytes. (a maximum of 12K bytes)

Return codes are:

```
R15=0   ALL OK
   =4   PAGE NOT IN DUMP
   =8   CODE INVALID
```

Note: The next subcode call (40 or 41) will overlay the buffer returned by the previous invocation of subcode 40 or 41.

- **Code = 41**: Request data from an address.

```
PLIST  DS   0F
       DC   AL4(*-*)   ADDRESS OF DATA WANTED
       DC   H'41'      CODE FIELD
       DC   H'0'       NUMBER OF USABLE BYTES RETURNED TO
                       USER.  THIS COUNT WILL VARY.
       DC   AL4(*-*)   ADDRESS OF BUFFER RETURNED TO
                       CALLER CONTAINING THE ADDRESS OF
                       THE DESIRED DATA.  THE ADDRESS
                       REQUESTED WILL BE ROUNDED DOWN TO
                       A PAGE BOUNDARY.
```

The address in the buffer will point to the beginning of the page containing the address of the requested data. The preceding page and the following page may also be present. The purpose is to provide the page before and the page after the requested page. The user must index into the page for his address ... or use SVC 199 code 40. The last halfword of PLIST will contain the total number of bytes. (a maximum of 12K bytes)

Return codes are:

```
R15=0   ALL OK
   =1   PRECEDING PAGE NOT PRESENT
   =2   FOLLOWING PAGE NOT PRESENT
   =3   PRECEDING AND FOLLOWING PAGES NOT PRESENT
   =4   PAGE NOT IN DUMP
   =8   CODE INVALID
```

Note: The next subcode call (40 or 41) will overlay the buffer returned by the previous invocation of subcode 40 or 41.

- **Code = 50**: Call for the dump information record. This record contains the registers and the first 256 bytes of low core at the time the dump command was issued.

```
PLIST   DS    OF
        DC    AL4(*-*)    ADDRESS OF 4K BUFFER CONTAINING
                          THE DUMP RECORD
        DC    H'50'       CODE FIELD
```

This record may have no value to a virtual machine being dumped. This evaluation is up to the user. The DSECT (DMPBLOKS) contained in DMKMAC MACLIB should be used to address the particular fields.

Return codes are:

```
R15=0   ALL OK
   =4   CONTROL WORD NOT PRESENT
   =8   CODE INVALID
```

- **Code = 60**: Request PRTDUMP to print a buffer that has been translated.

```
PLIST   DS    OF
        DC    AL4(*-*)    ADDRESS OF BUFFER TO BE PRINTED
        DC    H'60'       CODE FIELD
        DC    H'0'        NUMBER OF CHARACTER LINES
```

The buffer contains translated data with a fixed length of 133 characters, including prefixed print control code.

Return codes are:

```
R15=0   ALL OK
   =4   NUMBER OF LINES = 0
   =8   CODE INVALID
```

- **Code = 70**: Request PRTDUMP to print the registers and PSWs and the dump, from page 0 to the end, and the appended load map.

```
PLIST  DS    0F
       DC    AL4(*-*)   RESERVED
       DC    H'70'      CODE FIELD
```

Return codes are:

```
R15=0   ALL OK
   =8   CODE INVALID
```

- **Code = 71**: Request to format and print the appended load map.

```
PLIST  DS    0F
       DC    AL4(*-*)   RESERVED
       DC    H'71'      CODE FIELD
```

Return codes are:

```
R15=0   ALL OK
   =8   CODE INVALID
```

- **Code = 80**: Change register set and PSW in the dump information record.

```
PLIST  DS    0F
       DC    AL4(*-*)   ADDRESS OF BUFFER CONTAINING
                        REGISTER SET AND PSW (IN THAT ORDER)
       DC    H'80'
```

Note: Used when DMPINREC does not contain valid virtual machine register set and PSW.

Return codes are:

```
R15=0   ALL OK
   =8   CODE INVALID
```

- **Code = 90**: Return to user a module and entry point name, when given an address.

```
PLIST  DS    0F
       DC    AL4(*-*)   ADDRESS OF MODULE OR ENTRY NAME
       DC    H'90'      CODE FIELD
       DC    H          RESERVED
       DC    CL8' '     ENTRY NAME RETURNED TO CALLER
       DC    AL4(*-*)   ENTRY ADDRESS TO CALLER
       DC    CL8' '     MODULE NAME TO CALLER
       DC    AL4(*-*)   MODULE ADDRESS TO CALLER
```

Return codes are:

```
R15=0   ALL OK
   =2   MAP NOT PRESENT
   =4   ADDRESS NOT IN MAP
   =8   CODE INVALID
```

- **Code = 91**: Return to caller an entry or module name address when given a name.

```
PLIST  DS   0F
       DC   AL4(*-*)   ADDRESS RETURNED TO CALLER
       DC   H'91'      CODE FIELD
       DC   CL8'NAME'  NAME FIELD
```

Address is returned to caller.

Return codes are:

```
R15=0    ALL OK  ADDRESS RETURNED
   =2    MAP NOT PRESENT
   =4    NAME NOT FOUND IN MAP
   =8    CODE INVALID
```

For all SVC 199 services, the following codes are returned in R15 on I/O errors:

- Disk Errors

  Reading a file (other than "file does not exist" or "end of file"):

  `R15=100`

  Writing a file:

  `R15=200`

- Printer Errors

  Virtual printer error (other than 'channel 9 or 12 sensed' on virtual type 3211, 3262, 3289... printer):

  `R15=500`

  A message is issued from SVC 199 services before the return is made.

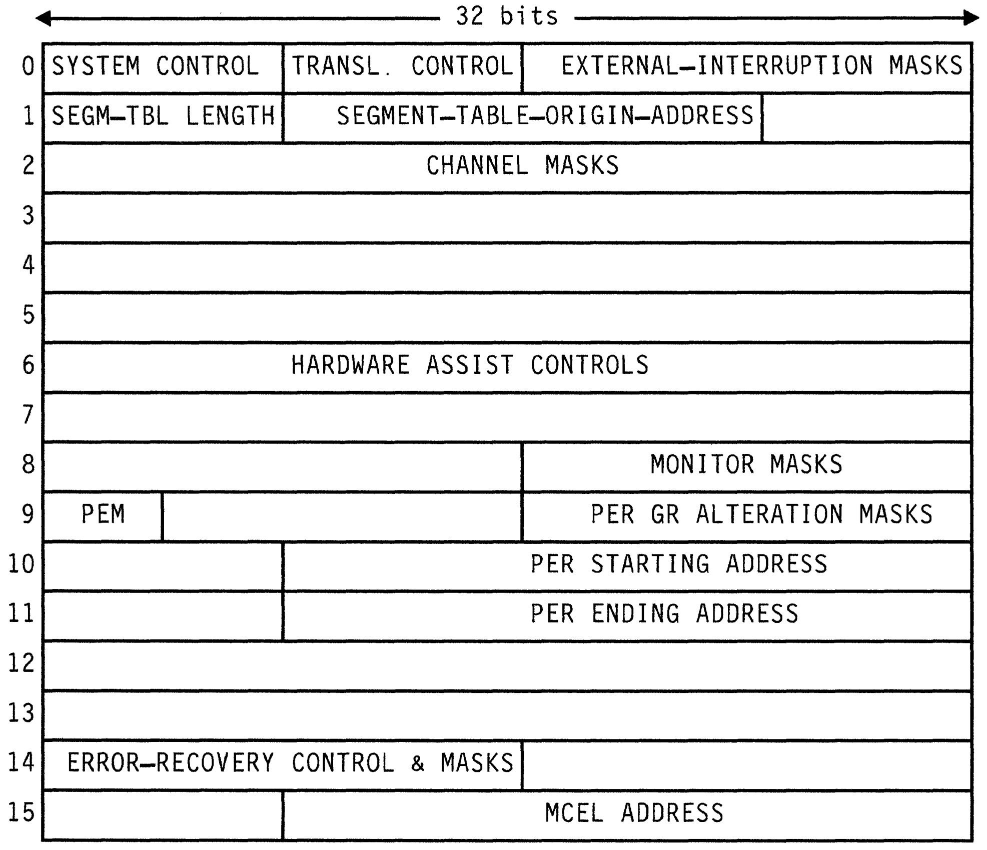
Appendix D. Control Registers

The control registers are used to maintain and manipulate control information that resides outside the Program Status Word (PSW). There are sixteen 32-bit registers for control purposes. The control registers are not part of addressable storage.

At the time the registers are loaded, the information is not checked for exceptions, such as invalid segment-size or page-size code or an address designating an unavailable or a protected location. The validity of the information is checked and the errors, if any, indicated at the time the information is used.

Figure 69 is a summary of the control register allocation.

Figure 70 on page 352 is a description of the Extended Control (EC) PSW.



PEM = PER EVENT MASKS

Figure 69. Control Register Allocation

System Mask	Key	EMWP	0	CC	Program Mask	0
0 7	8 11	12 15	16 17	18 19	20 23	24 31

0	Instruction Address
32 39	40 63

The fields of the PSW are:

Bits	**Contents**
0	Must be zero.
1	PER (Program Event Recording) enabled.
2-4	Must be zero.
5	Address translation.
6	Summary I/O mask.
7	Summary extension.
8-11	The protection key determines if information can be stored or fetched from a particular location.
12	Extended control mode.
13	The machine check flag is set to 1 if machine check interruptions are enabled.
14	The wait state flag is set to 1 when the CPU is in the wait state.
15	The problem state flag is set to 1 when the CPU is operating in the problem rather than the supervisor state.
16-17	Must be zero.
18-19	The condition code reflects the result of a previous arithmetic, logical, or I/O operation.
20-23	The program mask indicates whether or not various program exceptions are allowed to cause program interrupts.
24-39	Must be zero.
40-63	The instruction address gives the location of the next instruction to be executed for program interrupts or of the instruction last executed for external interrupts.

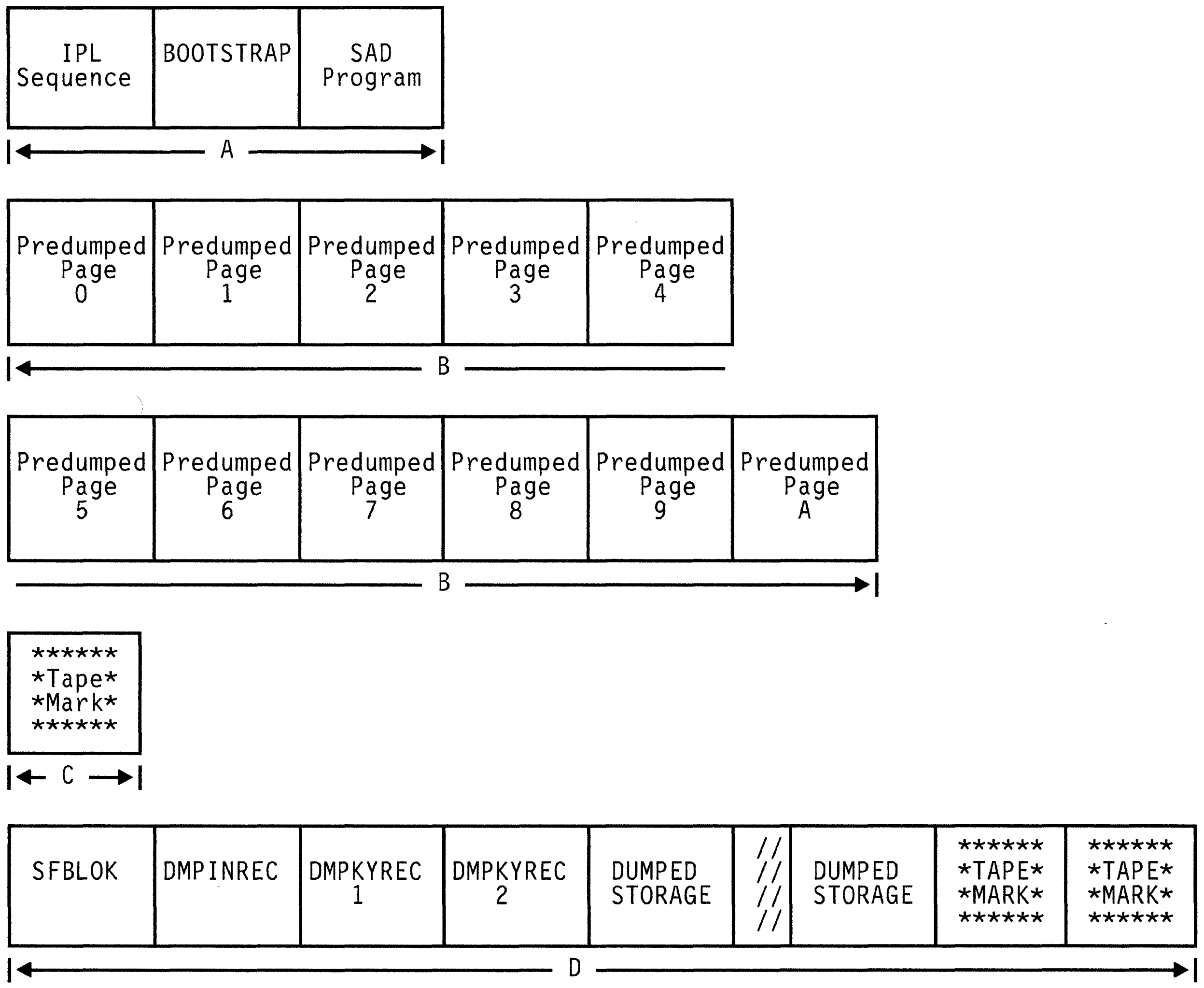
Figure 70. The Extended Control PSW (Program Status Word)

Appendix E. Stand-Alone Dump Formats

Tape Format

A tape used with the stand-alone dump facility has the format shown in Figure 71 on page 354.

- If the IPL tape and the dump device are not the same, the IPL tape includes sections A, B, and C.
- If the dump device is a tape, but not the same tape as the IPL tape, the dump output tape includes sections C and D.
- If the IPL device and the dump device are the same, the tape includes sections A, B, C, and D.



A = = > Written by the Stand-Alone Dump Utility on the IPL tape at generation time.

B = = > Written by BOOTSTRAP on the IPL tape.

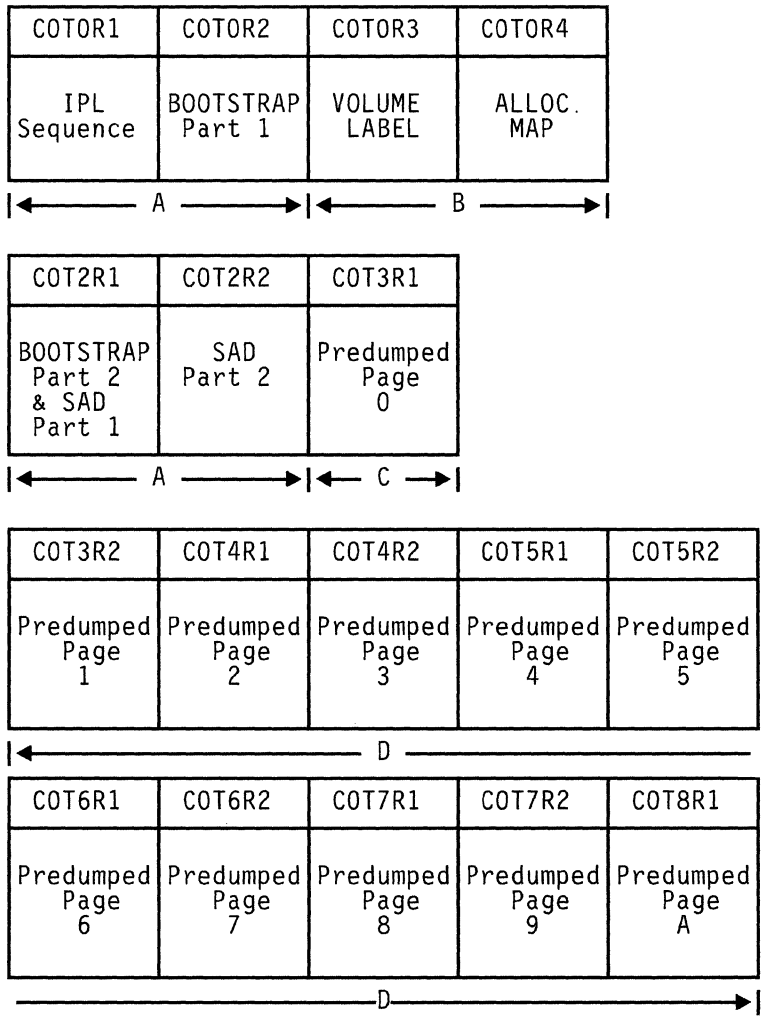
C = = > Written by BOOTSTRAP if the IPL tape is the same as the dump tape. Written by stand-alone dump program if the IPL tape is not the same as the dump tape.

D = = > Written by the stand-alone dump program on the dump tape.

Figure 71. Stand-Alone Dump Facility Tape Format

DASD Format

When you use a DASD device to IPL the stand-alone dump program, the system uses cylinder 0 to hold the program. Cylinder 0 must be CP formatted and allocated as permanent space. The stand-alone dump facility has the format shown in Figure 72.



A = = > Written by the Stand-Alone Dump Utility on the IPL DASD at generation time.

B = = > Written by FORMAT/ALLOCATE program.

C = = > Written by BOOTSTRAP Part 1 on the IPL DASD.

D = = > Written by BOOTSTRAP Part 2 on the IPL DASD.

CnTnRn identifies cylinder, track, and record numbers.

Figure 72. Stand-Alone Dump Facility DASD Format

Printer Format

Dumps to printer devices are printed as follows:

- CP formats the following data fields for each processor, beginning with the processor where the stand-alone dump program was IPLed:
  - CPU address (only if in AP or MP mode)
  - General purpose registers
  - Control registers
  - Floating point registers
  - Clock comparator
  - CPU timer values
  - Stored-status PSW
  - Prefix value (only if in AP or MP mode)
  - External interrupt old/new PSWs
  - SVC old/new PSWs
  - Program check old/new PSWs
  - Machine check old/new PSWs
  - I/O interrupt old/new PSWs.
- The following fields are printed for the processor where the stand-alone dump program was IPLed:
  - TOD clock.
- Lines of duplicate data will have a suppression message after the first line of the data is printed.
- A half page (2048 bytes) of all zeros has one line of zeros printed with the key, followed by a line suppressed message.
- The dump page is interpreted on the right-hand side of the printout.

Error Handling

Basic error recovery is available for DASD, tape, and printer devices used as IPL or output devices. In addition, the following information may be of value when the system detects errors:

- The CSW is at location X'40'.
- The I/O address is at location X'BA'.
- 32 bytes of sense data are at location X'2E0'.
- The starting and ending addresses of the CP Trace Table are stored in the PSA at X'7B0' and X'7B4', respectively, in addition to the low storage locations.

Under certain error conditions, storage areas may be overlaid. This could cause fields in SFBLOK and DMPINREC to be incorrect. (For example, fields containing date and time information.)

Appendix F. Converting Symptom Summary and Dump Files

CONVERT Command

The symptom summary and dump files created by VM/370 Release 6 IPCS cannot be processed by VM/SP IPCS. If your installation is already using VM/370 IPCS and plans to install VM/SP IPCS, a method of converting VM/370 IPCS symptom summary files and PRBnnnnn dumps to the format required by VM/SP IPCS has been provided.

The existing system IPCS problem report files need not be converted for use with the PRB, PROB, or APAR commands of the VM/SP IPCS. The format of the problem report files will vary slightly from the VM/370 IPCS to the VM/SP IPCS, but the data contained within will be accurate.

VM/IPCS Extension files are compatible with the VM/SP IPCS and need not be converted.

The format of the command used to invoke this utility is:

CONVERT	*fn* [*ft* [*fm*]]

where:

fn
: is the filename of the file to be converted.

ft
: is the filetype of the file to be converted.

fm
: is the filemode of the file to be converted.

Usage Notes:

All conversions:

1. A file already converted is **not** converted again.

2. Before using CONVERT, enter:

   SET IMPEX OFF
   so that the CMS CONVERT COMMANDS EXEC will not be invoked. After all conversions are done, enter:

   SET IMPEX ON

Symptom summary conversions:

1. You must supply the filetype and filemode unless they match the defaults of SUMMARY A1.

2. The original symptom summary file is **not** erased.

3. The converted file has a fileid of SYMPTOM1 SUMMARY A1.

Dump conversions:

1. When converting dumps, you supply **only** the filename.

2. The filetype and filemode of a dump to be converted **must** be DUMP A1.

3. The dump file must reside on your A-disk, and that disk must be accessed in write mode.

4. The program checks to make sure there is enough room on the A-disk to complete the conversion. If not, CMS subset is entered to allow you to make room on your A-disk.

Responses:

```
DMMCVD803I 'fn ft fm' CONVERSION COMPLETE
DMMCVS803I 'fn ft fm' CONVERSION COMPLETE
```

Error Messages:

```
DMMCVD100S ERROR 'nnn' READING FILE 'fn ft fm'
DMMCVD200S ERROR 'nnn' WRITING FILE 'fn ft fm'
DMMCVD811I NO IPCS MAP APPENDED TO DUMP
DMMCVD812E 'fn ft fm' ALREADY CONVERTED
DMMCVD820I INSUFFICIENT MAP PROCESSING DISK SPACE
           FOR  'fn ft fm'
DMMCVS100S ERROR 'nnn' READING FILE 'fn ft fm'
DMMCVS200S ERROR 'nnn' WRITING FILE 'fn ft fm'
DMMCVS812E 'fn ft fm' ALREADY CONVERTED
DMMCVS862E NO PARAMETERS ENTERED
DMMCVS866E FILE 'fn ft fm' NOT FOUND
DMMCVS867E 'fn ft fm' NOT A SYMPTOM SUMMARY FILE
```

Conversion of the Symptom Summary File

To convert the symptom summary file, enter the command “CONVERT fn ft fm,” where “fn ft fm” refers to the existing system IPCS symptom summary file.

For example:

CONVERT SYMPTOM SUMMARY D1

The CONVERT command always creates a new file with a fileid of SYMPTOM1 SUMMARY A1. To complete the conversion process, the new file created by CONVERT (i.e., SYMPTOM1 SUMMARY A1), must be renamed SYMPTOM SUMMARY A1 after erasing or renaming the original symptom summary file.

If the original symptom summary file has already been converted, message DMMCVS812E is issued. If the specified file is not a symptom summary file, message DMMCVS867E is issued.

Conversion of Dumps

To convert an existing VM/370 IPCS dump to the format compatible with the VM/SP IPCS, enter “CONVERT fn.”

For example:

CONVERT PRB00001

The CONVERT command converts the VM/370 IPCS dump file to the format usable by the VM/SP IPCS for subsequent dump viewing. If the dump has already been converted, message DMMCVD812E is issued. If the existing dump does not have an IPCS map appended to it, message DMMCVD811I is issued.

Appendix G. IPCS Interface Files

The following table lists the IPCS interface files for CP, GCS, and TSAF. When debugging CP, GCS, and TSAF, these files must be available to IPCS for successful dumps.

Component	Interface File
CP	DMKTED TEXT DMKTEE TEXT DMKTEF TEXT DMKTEM TEXT DMKTES TEXT
GCS	CSIIAL TEXT CSIIDS TEXT CSIIEX TEXT CSIIFL TEXT CSIIFT TEXT CSIIIU TEXT CSIIMP TEXT CSIIPR TEXT CSIITA TEXT CSIITL TEXT CSIIVL TEXT CSIYTD TEXT CSIYTS TEXT
TSAF	ATSZAD TEXT ATSZBD TEXT ATSZCT TEXT ATSZDS TEXT ATSZEO TEXT ATSZEX TEXT ATSZLD TEXT ATSZNT TEXT ATSZPT TEXT ATSZRD TEXT ATSZRT TEXT ATSZST TEXT ATSZTD TEXT ATSZTE TEXT ATSZTF TEXT ATSZTO TEXT ATSZTR TEXT ATSZTS TEXT

Figure 73. IPCS Interface Files

Summary of Changes

This manual contains material formerly found in the *VM/SP System Programmer's Guide* (SC19-6203) or *VM/SP HPO System Programmer's Guide* (SC19-6224), *VM/SP Group Control System Guide* (SC24-5249), and *VM/SP Interactive Problem Control System Guide* (SC24-5260).

To obtain editions of the *VM/SP System Programmer's Guide* you must order using the pseudo-number assigned to the respective edition. For:

Release 4, order ST00-1578
Release 3, order ST00-1352
Release 2, order SQ19-6203
Release 1, order ST19-6203.

To obtain editions of the *VM/SP HPO System Programmer's Guide* you must order using the pseudo-number assigned to the edition. For:

Release 4.2, order ST00-1897

To obtain editions of the *VM/SP Group Control System Guide* you must order using the pseudo-number assigned to the edition. For:

Release 4, order ST00-1842.

To obtain editions of the *VM/SP Interactive Problem Control System Guide* you must order using the pseudo-number assigned to the edition. For:

Release 4, order ST24-5260.

Summary of Changes for VM Diagnosis Guide

Summary of Changes
for LY24-5241-0
for VM Release 5

Transparent Services Access Facility (TSAF)

Is a facility that lets users connect to and communicate with local or remote virtual machines within a group of systems. With TSAF, a user can connect to a program by specifying a name that the program has made known, instead of specifying a userid and nodeid.

High Performance Option (HPO)

This manual was updated so that it applies to both VM/SP and VM/SP HPO.

Manual Organization

Chapters 45, 46, and 47 from the *VM/SP System Programmer's Guide* were moved into Chapters 1, 3, and 4 of this manual.

Information on PER and TRACE from the *VM/SP CMS User's Guide* and *VM/SP CP Command Reference* were moved into Chapter 2 of this manual.

Information on abend dumps from the *VM/SP Operator's Guide* were moved into Chapter 3 of this manual.

Information on network dump and NCPDUMP from the *VM/SP Operator's Guide* was moved into Chapter 3 of this manual.

Information on the Stand-Alone Dump Facility from the *VM/SP Operator's Guide* was moved into Chapter 3 of this manual.

Part of Chapter 2 from the *VM/SP Group Control System Guide* was moved into Chapter 5 of this manual.

Chapters 1, 2, and 3 from the *VM/SP Interactive Problem Control System Guide* was moved into Chapter 7 of this manual.

Chapter 4 from the *VM/SP Interactive Problem Control System Guide* and Appendix B from the *VM/SP Group Control System Guide* were moved into Appendix A of this manual.

Appendix B is new to this manual.

Appendix C from the *VM/SP Interactive Problem Control System Guide* was moved into Appendix C of this manual.

Appendix A from the *VM/SP System Programmer's Guide* was moved into Appendix D of this manual.

Information on Stand-Alone Dump Formats from the *VM/SP Operator's Guide* was moved into Appendix E of this manual.

Information on converting symptom summary and dump files from the *VM/SP Interactive Problem Control System Guide* was moved into Appendix F of this manual.

Appendix G is new to this manual.

Miscellaneous

Minor technical and editorial changes have been made throughout this manual.

Some error messages changed to mixed case.

Glossary of Terms and Abbreviations

This section explains or defines the terms, acronyms, and abbreviations that appear in this manual. For a complete list of terms used in VM/SP refer to the *VM/SP Library Guide, Glossary, and Master Index*, GC19-6207. You may also want to refer to the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699.

A

ACF/SSP. (1) See Advanced Communications Function for the System Support Program. (2) In this publication, ACF/SSP refers to ACF/SSP Version 3 unless otherwise noted.

ADT. Active Disk Table.

Advanced Communications Function for the System Support Program. An IBM program product. ACF/SSP is a collection of utilities and small support programs for NCP, the Network Control Program. You must have ACF/SSP to use NCP. In this manual, ACF/SSP refers to ACF/SSP Version 3.

Advanced Communications Function/Virtual Telecommunications Access Method. A program product that manages the exchange of data in a SNA network controlled by a VM/SP operating system.

Advanced Program-to-Program Communication/VM (APPC/VM). An Application Program Interface (API) for communicating between two virtual machines that is mappable to the SNA LU 6.2 APPC interface and is based on IUCV functions. Along with the TSAF virtual machine, APPC/VM provides this communication within a single system and throughout a collection of systems.

AFT. Active File Table.

AP. see Attached Processor.

APAR. see Authorized Program Analysis Report.

APPC/VM. see Advanced Program-to-Program Communication/VM

AP/MP mode. A mode of VM/SP used when running in an attached processor or multiprocessor system.

ASCII. American National Standard Code for Information Interchange.

Attached Processor (AP). A processor with no I/O capability. An attached processor is always linked to the processor initialized for I/O handling.

Authorized Program Analysis Report (APAR). A report of a problem caused by a suspected defect in a current unaltered release of a program accepted by IBM support for further action.

authorized userid. A GCS userid that you've provided with access to the GCS supervisor, supervisor state, and (in some cases) certain restricted CP commands. You provide this access by including the userid on a list of authorized userids compiled with the GCS GROUP EXEC. The virtual machine associated with an authorized userid is an "authorized" machine, and programs running in that machine are "authorized" applications.

B

Basic Control (BC) mode. A mode in which a virtual machine resumes execution after an I/O interrupt, a page fault, or a DIAGNOSE code X'18'.

Basic Sequential Access Method (BSAM). An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or a direct access device.

BPI. Bytes Per Inch

BSAM. see Basic Sequential Access Method.

C

CAW. see Channel Address Word.

CCS. see Console Communication Service.

CCW. see Channel Command Word.

Channel Address Word (CAW). An area in storage that specifies the location in main storage at which a channel program begins.

Channel Command Word (CCW). A doubleword at the location in main storage specified by the channel address word. One or more CCWs make up the channel program that directs data channel operations.

Channel Status Word (CSW). An area in storage that provides information about the termination of input/output operations.

Channel-to-Channel Adapter. A hardware device that can be used to connect two channels on the same computing system or on different systems.

CKD. Count-Key-Data

CMS. see Conversational Monitor System.

CMS system disk. The virtual disk (S-disk) that contains the CMS nucleus and the disk-resident CMS commands. The CMS system disk can have extensions, usually the Y-disk.

CMS/DOS. refers to the DOS-like simulation environment provided under the CMS component of the VM/SP.

common dump receiver. One userid in a virtual machine group appointed to receive other group members' storage dumps. Unless you specify otherwise, all dumped information automatically goes to this userid (identified with the GCS GROUP EXEC). It should be an authorized userid in order to receive fetch-protected data as well as storage with a key other than 14.

common lock. A doubleword in storage, controlled by the GCS LOCKWD macro. When a program is using common storage, it can turn the common lock "on." Other programs that examine the lock and find it on cannot gain access to common storage.

common storage. A shared segment of reentrant code that contains free storage space, the GCS supervisor, control blocks, and data that all members of a virtual machine group share.

concurrently. Concerning a mode of operation that includes the performance of two or more operations within a given interval of time.

Console Communications Service (CCS). A CP component which helps process information (that's ultimately heading to or from a SNA terminal screen) as it passes between CP and VSCS (VTAM SNA Console Support). This helps a SNA terminal serve as an operator's virtual console.

Control Program (CP). The component of VM/SP that manages the resources of a single computer so that multiple computing systems appear to exist.

Count-Key-Data. Those DASD devices whose architecture defines variable size records consisting of count, key, and data fields.

Conversational Monitor System (CMS). A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities, and operates under the control of CP.

CP. see Control Program.

CPTRAP. This facility is a CP debugging tool. It is used to create a reader spool file of selected trace table entries, CP data, and virtual machine data in the order that they happen. The TRAPRED command can help you access and use this collected data.

CPU. Central Processing Unit.

CSW. see Channel Status Word.

CVT. Communications Vector Table.

D

DASD. see Direct Access Storage Device.

DAT. see Dynamic Address Translation.

DCB. Data Control Block.

DCSS. see Discontiguous Shared Segments.

DDR. DASD Dump/Restore.

Direct Access Storage Device (DASD). A storage device in which the access time is effectively independent of the location of the data.

directory. For VM/SP, a CP disk file that defines each virtual machine's normal configuration: the userid, password, normal and maximum allowable virtual storage, CP command privilege class or classes allowed, dispatching priority, logical editing symbols to be used, account number, and CP options desired.

Discontiguous Shared Segments (DCSS). Synonymous with *discontiguous segment.*

discontiguous segment. A 64K segment of storage that was previously loaded and saved and assigned a unique name. The segment(s) can be shared among virtual machines if the segment(s) contain reentrant code.

DPA. see Dynamic Paging Area.

dyadic. A system having two processors that cannot be configured into two independent uniprocessors that use separate control programs. For example, the 3081 Processor Complex contains two processing units that share central storage.

Dynamic Address Translation (DAT). In System/370 virtual storage systems, the change of a virtual address to a real storage address during execution of an instruction.

Dynamic Paging Area (DPA). An area of real storage that CP uses for virtual machine pages and pageable CP modules.

E

ECB. Event Control Block(s).

ECMODE. Extended Control Mode.

EXEC. refers to EXECs using the System Product Interpreter (REXX), EXEC 2, or CMS EXEC languages.

Extended Control (EC) mode. Extended control mode, a System/370 mode for formatting and use of control and status information. Contrast with "basic control (BC) mode."

F

FBA. see Fixed-Block Architecture.

FCB. (1) Function Control Block. (2) Forms Control Buffer.

File Status Table (FST). A table that describes the attributes of a file on a CMS disk, including filename, filetype, filemode, date last written, and other status information.

Fixed-Block Architecture (FBA). Those DASD devices whose architecture uses fixed blocks or records of 512 bytes.

flush list. A set of pages available to replenish the free list.

free list. A list maintained by CP that points to a set of pages that can be allocated to satisfy both virtual machine and system page requests.

FST. see File Status Table.

G

GCS. see Group Control System

GCS supervisor. Reentrant GCS code that resides in common storage.

General I/O. A set of I/O functions that allow a GCS program to control attached non-DASD devices.

global system lock. A defer lock that provides system integrity for AP and MP support of command processing and code executed via IOBLOK, TRQBLOK, or CPEXBLOK.

GPR. General Purpose Register.

group configuration file. A file that is created by the GROUP EXEC and contains the "blueprint" for building your virtual machine group. The name of this file is *systemname* GROUP, where "systemname" is the name of your GCS saved system.

Group Control System (GCS). A component of VM/SP, consisting of a shared segment that you can IPL and run in a virtual machine. It provides simulated MVS services and unique supervisor services to help support a native SNA network.

GROUP EXEC. A GCS installation tool that prompts you for the specifications needed to build a GCS configuration file.

guest operating system. A second operating system that runs on your primary operating system. An example of a guest operating system is VSE running on VM/SP to support VCNA.

Guest Virtual Machine (GVM). A virtual machine in which an operating system is running.

GVM. see Guest Virtual Machine.

I

interactive. (1) An application in which each user entry calls forth a response from a system or program. (2) The classification given to a virtual machine depending on this virtual machine's processing characteristics. When a virtual machine uses less than its allocated time slice because of terminal I/O, the virtual machine is classified as being interactive. See also non-interactive.

Interactive Problem Control System (IPCS or VM/SP IPCS). A component of VM/SP that permits on-line problem management, interactive problem diagnosis, on-line debugging for disk-related CP or virtual machine abend dumps, problem tracking, and problem reporting.

Inter-User Communication Vehicle (IUCV). A communications facility that allows users to pass information. It enables a program running in a virtual machine to communicate with other virtual machines, with a CP system service, and with itself.

in-queue virtual machines. A virtual machine on the run list waiting to be dispatched.

IPCS. see Interactive Problem Control System.

IPL. Initial Program Load.

IUCV. see Inter-User Communication Vehicle.

I/O. Input/Output.

L

local lock. A doubleword in storage, controlled by the LOCKWD macro. When it's "on," the program that acquired it has exclusive use of the virtual machine.

logon. The procedure by which a user begins a terminal session.

logoff. The procedure by which a user ends a terminal session.

M

Mb. see Megabyte.

Megabyte. 1,048,576 bytes.

minidisk. Synonym for virtual disk.

MP. see Multiprocessor.

Multiprocessor (MP). Two or more interconnected processors that execute programs simultaneously.

multitasking. The act of providing services for many tasks that are active at the same time.

Multiple Virtual Storage (MVS). An alternative name for OS/VS2 release 2.

MVS. see Multiple Virtual Storage.

N

named system. A collection of saved pages a user can IPL or load by name.

native mode. A mode in which an operating system is run stand-alone on the real machine instead of under VM/SP.

native SNA network. A VM/SP network that operates according to the conventions of SNA (Systems Network Architecture) and functions as part of a VM/SP system without help from a guest operating system.

NCP. Network Control Program.

NCPDUMP. Network Control Program DUMP.

noninteractive. The classification given to a virtual machine depending on this virtual machine's processing characteristics. When a virtual machine usually uses all its allocated time slice, it is classified as being noninteractive or compute bound. See also interactive.

non-resident pages. Pages whose contents are on DASD but not in real storage. A page is considered nonresident when an attempt to load its real address returns a nonzero condition code.

O

OS. Operating System.

P

page frame. A block of 4096 bytes of real storage.

page table. A table in CP that indicates whether a page is in real storage and matches virtual addresses with real storage addresses.

PIE. Program Interrupt Element.

preferred paging area. A special area of auxiliary storage where frequently used pages are paged out. It provides high speed paging.

Prefix Storage Area (PSA). A page zero (not always 0 in VM/SP HPO) of real storage that contains machine-used data areas and CP global data.

Primary Option Menu. The title of the first screen you see with the GROUP EXEC. This menu asks you to name your GCS system and then directs you to all the other GROUP EXEC screens.

private storage. A combination of application code and GCS code that is available to only one particular virtual machine. No virtual machine can access or share another's private storage areas.

Program Status Word (PSW). An area in storage used to indicate the order in which instructions are executed, and to hold and indicate the status of the computer system. Synonymous with processor status word.

Program Temporary Fix (PTF). A temporary solution or by-pass of a problem diagnosed by IBM field engineering as the result of a defect in a current unaltered release of the program.

projected working set. The sum of referenced pages and pages stolen from the virtual machine which is used to determine whether the virtual machine can be added to the run list.

PSA. see Prefix Storage Area.

PSR. Program Support Representative.

PSW. see Program Status Word.

PSW key. Bits 8 through 11 in the program status word.

PTF. see Program Temporary Fix.

Q

QSAM. see Queued Sequential Access Method.

Queued Sequential Access Method (QSAM). An extended version of BSAM. When this method is used, a queue is formed of input data blocks that are awaiting processing or output data blocks that have been processed and are awaiting transfer to auxiliary storage or to an output device.

queue-add. The action by the system scheduler, DMKSCH, of placing a runnable virtual machine on the list of virtual machines that can be given control of a processor.

queue-drop. The action by the system scheduler, DMKSCH, of removing a virtual machine from the list of virtual machines that can be given control of a processor.

R

real machine. The actual processor, channels, storage, and I/O devices required for operation of VM/SP.

recovery machine. The first machine to join a virtual machine group. It has responsibility for executing routines that were set with the GCS MACHEXIT macro and cleaning up system resources when machines leave the group.

Restructured Extended Executor (REXX). A command programming language that allows you to combine sequences of commands to create new commands. The System Product Interpreter processes programs written in the REXX language. REXX is very suitable for writing EXECs and editor macros.

REXX. see Restructured Extended Executor

Remote Spooling Communication Subsystem Networking (RSCS). A program product for VM/SP, it is a special-purpose subsystem that sends and receives messages, files, commands, and jobs over a computer network. In this manual, RSCS refers to RSCS Version 2 unless otherwise noted.

RSCS. see Remote Spooling Communications Subsystem Networking.

run list. A queue that contains in-queue virtual machines that are competing for processor resources. These virtual machines are sorted by deadline priority.

S

saved segment. The body of reentrant code that constitutes a saved system, a shared system, a shared segment, or discontiguous shared segment in storage.

SCP. System Control Program.

SDB. Structured Data Base.

second-level storage. The storage that appears to be real to a virtual machine.

segment. A contiguous 64K area of virtual storage (not necessarily contiguous in real storage) that is allocated to virtual machine or CP.

segment table. A table used in dynamic address translation to control user access to virtual storage segments. Each entry indicates the length, location, and availability of a corresponding page table.

shadow page table. A table that maps real storage allocations (first level storage) to a virtual machine's virtual storage (third level storage) for use by the real machine in its paging operations.

shared segment. A named, saved segment that can be shared by all members of a virtual machine group. This segment can be specified in the GROUP EXEC and linked automatically to machines when they IPL GCS. This segment can be loaded within the VMSIZE.

spool, spooled, spooling. Relates to the reading of input data streams and the writing of output data streams on auxiliary storage devices.

SR. Symptom Record.

SSP. System Support Program.

stand-alone dump. A program used to print the contents of storage that runs in a virtual machine not under control of an operating system such as CMS.

supervisor services. Unique services provided by the GCS supervisor.

System/370. applies to the 4300 and 303X series of processors.

S-disk. See CMS system disk.

S-STAT. A block of storage that contains the file status tables (FSTs) associated with the S-disk. The FSTs are sorted so that a binary search can be used to search for files. The S-STAT usually resides in the CMS nucleus so it can be shared. Only files with filemode of 2 will have their associated FSTs in the S-STAT.

T

task-user. A routine, within a GCS task, that can be an IUCV user.

third-level storage. The virtual storage created and controlled by a virtual machine.

time sharing. Sharing of computer time and resources.

| **Transparent Services Access Facility (TSAF).** A facility that lets users connect to and communicate with local or remote virtual machines within a collection of systems. With TSAF, a user can connect to a program by specifying a name that the program has made known, instead of specifying a userid and nodeid.

TRAPRED. This command accesses the CPTRAP reader file and the data collected in the file.

| **TSAF.** see Transparent Services Access Facility.

U

unauthorized userid. A GCS userid that runs in problem state and does not have access to restricted CP commands.

V

virtual address. An address that refers to virtual storage or a virtual I/O device address. It must, therefore, be translated into a real storage or I/O device address when it is used.

virtual disk. A logical subdivision (or all) of a physical disk storage device that has its own address, consecutive storage space for data, and an index or description of the stored data so that the data can be accessed. A virtual disk is also called a minidisk.

virtual machine. A functional simulation of a computer and its associated devices.

Virtual Machine Assist (VMA). A hardware feature available on certain VM/SP-supported processors that causes a significant reduction in the real supervisor state time used to control the operation of virtual machine systems such as VSE, DOS/VS, and OS/VS and to a lesser extent CMS, DOS, and OS when executing under VM/SP.

Virtual Machine Communication Facility (VMCF). A CP function that provides a method of communication and data transfer between virtual machines operating under the same VM/SP systems.

virtual machine group. The concept in the Group Control System of two or more virtual machines associated with each other through the same named system (e.g. IPL GCS1). Virtual machines in a group share common read/write storage and can communicate with one another through facilities provided by the Group Control System.

virtual storage. Storage space that can be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.

Virtual Storage Extended (VSE). Combination of the DOS/VSE system control program and the VSE/Advanced Functions program product. "DOS", in certain cases, is still used as a generic term. For example, disk packs initialized for use with VSE or any predecessor DOS or DOS/VSE system may be referred to as DOS disks.

Virtual Telecommunications Access Method (VTAM). A program product that controls communication and the flow of data in a computer network. It provides single-domain, multiple-domain, and multiple-network capability. VTAM runs under OS/VS1, MVS, VSE, and VM/SP.

VMA. see Virtual Machine Assist.

VMSIZE. The high storage boundary of a virtual machine.

VM/SP. refers to the VM/SP program package when you use it in conjunction with VM/370 Release 6.

VM/SP HPO. refers to the VM/SP program with the VM/SP High Performance Option (HPO) enhancements added to it.

VSCS. see VTAM SNA Console Support.

VSE. see Virtual Storage Extended.

VTAM. see Virtual Telecommunications Access Method.

VTAM SNA Console Support (VSCS). A component of ACF/VTAM that lets SNA terminals function as virtual machine consoles.

W

wrap spool file. A wrap spool file is established when the CPTRAP invoker issues CPTRAP START with the WRAP option. The size of the wrap spool file is determined by the file size information provided with the CPTRAP START WRAP nnnnn command. (nnnnn is the number of 4K blocks of records.) Records will be added to the spool file until the specified SPOOL size limit is reached. Then, newer records will replace older records in the spool file thereby using the same spool area over again.

Y

Y-disk. An extension of the CMS system disk.

Y-STAT. A block of storage that contains the File Status Tables (FSTs) associated with the Y-disk. The FSTs are sorted so that a binary search can be used to search for files. The Y-STAT usually resides in the CMS nucleus so it can be shared. Only files with filemode of 2 will have their associated FSTs in the Y-STAT.

Bibliography

Here is a list of IBM books that can help you use your system. If you don't see the book you want in this list, you might want to check the *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001.

Prerequisite Publications

IBM System/360 Principles of Operation, GA22-6821

IBM System/370 Principles of Operation, GA22-7000

Virtual Machine/System Product (VM/SP):

CMS Command Reference, SC19-6209
CMS Macros and Functions Reference, SC24-5284
CMS User's Guide, SC19-6210
CP Command Reference, SC19-6211
System Product Editor Command and Macro Reference, SC24-5221
System Product Editor User's Guide, SC24-5220
System Product Interpreter Reference, SC24-5239
System Product Interpreter User's Guide, SC24-5238

Virtual Machine/System Product High Performance Option (VM/SP HPO):

CP Command Reference, SC19-6227

Corequisite Publications

Virtual Machine/System Product (VM/SP):

CMS for System Programming, SC24-5286
CP for System Programming, SC24-5285
Data Areas and Control Block Logic Volume 1 (CP), LY24-5220
Data Areas and Control Block Logic Volume 2 (CMS), LY24-5221
Group Control System Command and Macro Reference, SC24-5250
Group Control System Diagnosis Reference, LY24-5239
Installation Guide, SC24-5237
Library Guide, Glossary, and Master Index, SC19-6207
Operator's Guide, SC19-6202
Planning Guide and Reference, SC19-6201
Problem Reporting Guide, SC24-5282
System Logic and Problem Determination Guide Volume 1 (CP), LY20-0892
System Logic and Problem Determination Guide Volume 2 (CMS), LY20-0893
System Messages and Codes, SC19-6204
System Messages Cross-Reference, SC24-5264
Service Routines Program Logic, LY20-0890

Transparent Services Access Facility Reference, SC24-5287

Virtual Machine/System Product High Performance Option (VM/SP HPO):

CP for System Programming, SC23-0341
Data Areas and Control Block Logic - CP, LY20-0896
Installation Guide, SC38-0107
Operator's Guide, SC19-6225
Planning Guide and Reference, SC19-6223
System Logic and Problem Determination Guide - CP, LY20-0897
System Messages and Codes, SC19-6226
System Messages Cross-Reference, SC20-0190

VM/SP Remote Spooling Communications Subsystem (RSCS) Networking Version 2:

Diagnosis Reference, LY24-5228
General Information, SH24-5055
Operation and Use, SH24-5058
Planning and Installation, SH24-5057
Program Reference and Operations Manual, SH24-5005

Virtual Machine/System (VM/SP) Product Pass-Through Facility:

Guide and Reference, SC24-5208
Logic, LY24-5208

Virtual Machine (VM):

CP Internal Trace Table (Poster), LX24-5202
Problem Determination Reference Information, LX23-0347
Running Guest Operating Systems, GC19-6212
System Facilities for Programming, SC24-5288

ACF/NCP V4, ACF/SSP V3 Diagnosis Guide, SC30-3255

IBM Field Engineering Programming System: General Information, G229-2228

IBM System/370 and 4300 Processor Bibliography, GC20-0001

IBM Vocabulary for Data Processing, Telecommunications, and Office Systems, GC20-1699

VSE/VSAM Programmer's Reference, SC24-5145

Notes:

1. References in text to titles of publications are given in abbreviated form.

2. The *VM/SP Library Guide, Glossary, and Master Index*, GC19-6207, describes all the VM/SP books and contains an expanded glossary and master index to all the books in the VM/SP library.

3. The *VM/SP HPO Library Guide, Glossary, and Master Index*, GC23-0187, describes all the VM/SP HPO books and contains an expanded glossary and master index to all the books in the VM/SP HPO library.

Index

International Business
Machines Corporation
P.O. Box 6
Endicott, New York 13760

File No. S370/4300-37
Printed in U.S.A.

LY24-5241-0

IBM®

VM Diagnosis Guide
Order No. LY24-5241-0

READER'S COMMENT FORM

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

__________ __________ ______ Help Information line ___ of ___

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? ___YES ___NO

Please print your name, company name, and address:

____

____

____

____

IBM Branch Office serving you: ________________________

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

Fold and tape Please Do Not Staple Fold and tape

IBM ®

VM Diagnosis Guide
Order No. LY24-5241-0

READER'S COMMENT FORM

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

______________ ______________ ________ **Help Information line ____ of ____**

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? ___YES ___NO

Please print your name, company name, and address:

____

____

____

____

IBM Branch Office serving you: ____

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape   Please Do Not Staple   Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL   PERMIT NO. 40   ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

Fold and tape   Please Do Not Staple   Fold and tape

IBM®

International Business
Machines Corporation
P.O. Box 6
Endicott, New York 13760

File No. S370/4300-37
Printed in U.S.A.

LY24-5241-0

LY24-5241-00