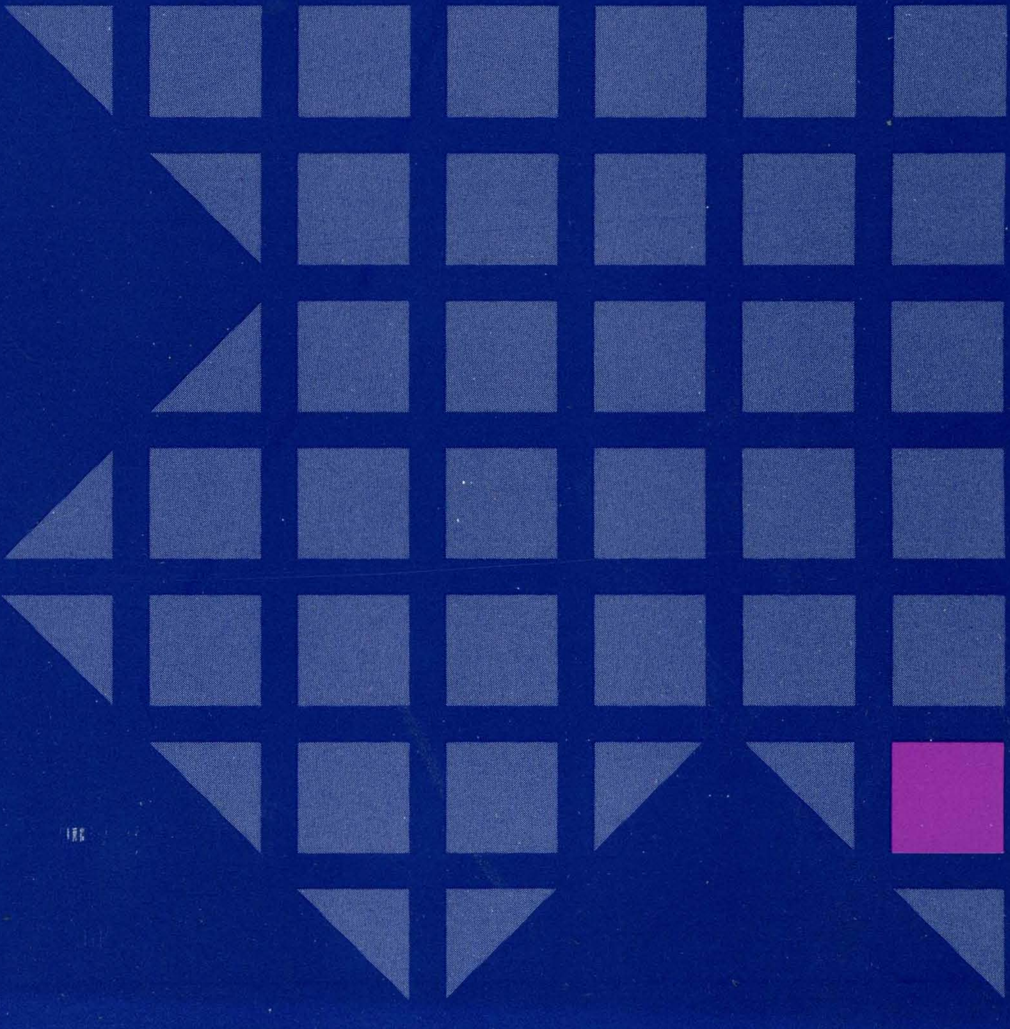
Virtual Machine/Extended Architecture™
System Product

SC23-0359-1

Dump Viewing Facility Operation Guide and Reference

VM/XA™ SP Release 2



IBM

Virtual Machine/Extended Architecture™
System Product

SC23-0359-1

Dump Viewing Facility Operation Guide and Reference

VM/XA™ SP Release 2

Second Edition (November 1988)

This is a major revision of, and obsoletes SC23-0359-0. See "Summary of Changes" on page 259 for a summary of the changes made to this manual.

This edition applies to Release 2 of the Virtual Machine/Extended Architecture System Product (VM/XA SP) Licensed Program 5664-308. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography,* GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the address given below. Requests for 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 International Business Machines, Department 52Q/MS 458, Neighborhood Road, Kingston, N.Y. 12401. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

Purpose

This publication presents a guide to the use of the Virtual Machine/ Extended Architecture System Product dump viewing facility, which is a program that allows the user to analyze and manage system software problems in an interactive mode under the conversational monitor system (CMS) for VM/XA SP. The dump viewing facility runs in System/370 mode.

Audience

This publication is for anyone who needs to use dump viewing facility to analyze dumps.

How to Use This Publication

Use the first part of the book, including Chapter 2, as a usage guide. The rest of the book is intended to be used as a reference manual. The reference manual contains commands, subcommands, and messages of dump viewing facility.

Related Publications

See the Bibliography at the back of this publication.

Contents

Chapter 1. Introduction

Major Functions

The dump viewing facility assists in the following tasks:

- Interactively analyzing dump data
- Formatting and printing dump data
- Reducing trace tables created by trace service tools.

Interactively Analyzing Dump Data

The dump viewing facility provides a variety of commands and subcommands that allow the user to interactively locate and display dump data. Using the dump viewing facility you can:

- Display real program status words, registers (including vector registers and designated elements), clocks, and the timer
- Display formatted data from any Virtual Machine/Extended Architecture™ System Product (VM/XA™ SP) control block or data area
- Display data in hexadecimal and EBCDIC
- Display data using 24- or 31-bit indirect addressing
- Display a chain of control block addresses in hexadecimal or display the data within the control blocks
- Locate a hexadecimal or EBCDIC string in the dump
- Print output from any DUMPSCAN subcommand
- Determine a module entry point and displacement, given an address
- Determine an address, given a module or entry point name
- Scroll forward or backward while viewing hexadecimal data
- Create a load map of module names and their entry points with addresses and displacements
- Assign symbolic names to subcommands
- Format, reduce and scroll through trace tables within a dump.

Formatting and Printing Dump Data

The dump viewing facility uses a dump file created by the CMS DUMPLOAD command to print summary reports. For CP dumps, these reports contain data from major CP control blocks and data areas. For VM dumps, there is limited information given. The summary reports available are:

- Virtual machines.

Virtual Machine/Extended Architecture and VM/XA are trademarks of the International Business Machines Corporation.

- Individual virtual machines with associated virtual I/O devices.
- Real I/O devices.
- Real storage frame tables.
- Trace table entries.
- Module load maps.
- Symptom records.
- General processor information.
- Dumpid (for virtual machine dumps only).
- For VM dumps of type GCS and RSCSV2, formatting of (VTAM type and VSCS type) control blocks, provided the formatting routine exit is taken. See Appendix E for more information.

Using the CMS DUMPLOAD command, you can print all pages of dumped storage in hexadecimal.

Reducing Trace Tables Created by Trace Service Tools

The dump viewing facility processes data trace information as well as CP trace table data. In addition, you can merge CP trace table information with data trace information and view it in chronological sequence. VM/XA SP records CP system trace table data in system trace files or to tape.

VM/XA SP provides three different types of data tracing. VM/XA SP records this trace information in system trace files. These three types are:

- **CP data trace**. Allows you to trace the most used paths in the CP code. You can use CP commands to specify what type of data you wish CP to record.
- **I/O data trace**. Allows you to trace I/O instruction streams, as well as data streams, coming from the I/O devices you select.
- **Guest data trace**. Allows you to trace data of virtual machines or groups of virtual machines. The virtual machine or group of virtual machines must use the MONITOR Call Class 10 instruction in application programs to actually send the trace data to CP. CP records the trace data in a system data file for the requesting virtual machine or group of virtual machines.

Types of Dumps the Dump Viewing Facility Processes

You can use the dump viewing facility to process any of the following kinds of dumps:

- CP (control program), including stand-alone dumps
- VM (virtual machine)

  The VM dumps you can use with the dump viewing facility are as follows:

  - CMS (conversational monitor system).
  - GCS (group control system).
  - Pass-Through. In this instance, Pass-Through refers to both the VM/Remote 3270 Display Option and Pass-Through Virtual Machine (PVM).

- RSCSNET (Remote Spooling Communications Subsystem) Version 1.
- RSCSV2 (Remote Spooling Communications Subsystem) Version 2.

Note: Throughout this document, for information on stand-alone dumps, see the sections that discuss CP dumps. For information on CMS, GCS, Pass-Through, RSCSNET and RSCSV2 dumps, see the sections that discuss VM dumps.

Requirements for Using the Dump Viewing Facility

To use all the functions the dump viewing facility provides to examine a dump, you need the following:

- The dump viewing facility installed on your system.

  For information about installing the dump viewing facility, see the *VM/XA System Product: Installation and Service* manual.

- A copy of the dump you want to examine loaded into a CMS file.

- In some instances, a copy of the load map that describes the system from which your dump was taken.

  For specific information on when the dump viewing facility requires a load map and how one is created and processed, see "Using Load Maps" on page 12 in this document.

Storage Requirements

The disk storage requirements for the dump viewing facility include space for the CP dump and the CP load map when it is used. A virtual machine load map and dump require less storage than a CP load map and dump. Therefore, the requirements given here should be viewed as a maximum. Figure 1 shows the space requirements in cylinders for the CMS file containing the load map:

BLKSIZE	3330	3340	3350	3375	3380
800	15	41	8	11	7
1024	15	36	8	10	7
2048	14	36	8	10	6
4096	14	32	8	9	6

Figure 1. Space Requirements in Cylinders for the Load Map File

After processing with the dump viewing facility MAP command, the dump viewing facility module map fits on 1 cylinder of disk space on any of the above devices.

Figure 2 shows the space requirements in cylinders for each 16 megabytes of dumped storage with the dump viewing facility module map appended dynamically or appended by the ADDMAP command.

BLKSIZE	3330	3340	3350	3375	3380
800	80	220	38	51	33
1024	80	197	38	48	31
2048	73	197	36	46	30
4096	73	172	36	43	28

Figure 2. Space Requirements in Cylinders per 16 Megabytes of Dumped Storage

Causes for a Dump

In VM/XA SP, dumps can be initiated by hardware, software, or a user. The cause of the dump determines the type of dump: CP, CMS, GCS, RSCSNET, RSCSV2, PVM, or stand-alone. For instance, if a machine check occurred, a CP dump would result. If a user issued the VMDUMP command in a CMS virtual machine, a CMS dump would result.

Note: If CP was unable to take an abend dump, you can initiate a stand-alone dump.

To initiate a stand-alone dump, use the stand-alone dump utility. See the *VM/XA SP: Planning and Administration* guide for information on creating the stand-alone dump utility. See the *VM/XA SP: Real System Operation* manual for information on running the stand-alone dump utility program.

Hardware-Initiated Dumps

Not all hardware errors result in a dump being taken. Some examples of hardware errors that do result in dumps are the following:

- Machine checks in the central processor
- Storage checks in main storage
- Channel checks in the I/O channels.

Some hardware errors cause a dump to be taken immediately as a result of a machine check condition. Other errors may alter the condition of the hardware (for example, a processor or main storage) in a manner that eventually will cause CP to detect an abnormal condition and take an abend dump.

If a dump is taken, it may contain symptoms of the hardware error. You may find additional symptoms by examining the hardware error log in the error recording cylinders, using the environmental recording, editing, and printing (EREP) facility, and by examining messages sent to the system operator's console.

Software-Initiated Dumps

Generally, a software error occurs when a sequence of instructions executed by a processor results in a condition that is incompatible with the design of the software system. Software errors have a variety of symptoms. Some typical symptoms are:

Loop A sequence of instructions is executed over and over again, infinitely.

Wait The software system cannot find work to do, or it encounters a condition that cannot be resolved.

Lockout A userid has become disabled, nondispatchable, or has no work-tasks to be run.

Storage overlay A program has stored data in the wrong location in real storage.

Invalid data A system control block or data area contains data that is inconsistent or erroneous.

Invalid address A control block or data area contains an address that is outside the storage areas to which the program has access. Or, a control block or data area points to a nonexistent control block. Since some addresses are generated by programs from data in control blocks, an invalid address can be generated if the data is in error.

Invalid instructions An instruction is found that does not conform to the system architecture, possibly because it has been modified in main storage.

Invalid addresses and instructions result in program checks. The other kinds of software errors are usually detected by CP as abnormal conditions. Both result in abnormal termination of CP and a dump being taken.

User-Initiated Dumps

System Restarts

A system restart is usually initiated by the system operator, and results in a dump being taken. The system is usually restarted when a problem exists in the hardware or software that results in a wait, loop, or lockout of one or more important userids, or in degraded performance. In these situations, it is usually desirable to reinitialize CP. System restart can perform this reinitializing and capture the circumstances of the problem in the dump.

VMDUMP Command

The CP command, VMDUMP, produces a dump of all or selected pages of storage that appear real to your virtual machine (second-level storage). In order for the resulting dump to be usable by the dump viewing facility, you must issue the CMS DUMPLOAD command to load the dump into a CMS file. The dump includes selected information for the virtual processor on which you issued the VMDUMP command. For more information on and the exact syntax of the VMDUMP command, see the *VM/XA SP: CP Command Reference.*

Location of a Dump

The dump taken is sent to a printer, a tape, or to the virtual reader of a specific virtual machine (userid). If a dump is to be analyzed using the dump viewing facility, it must be directed to tape or to a virtual reader. One userid commonly used for dumps is OPERATNS. The destination of the dump is determined by using the CP SET DUMP command. The SET DUMP command is described in the *VM/XA SP: CP Command Reference.*

Use of Dump Information

You can use data from a dump to assist in locating the source of the problem that caused the dump. As we have already discussed, in VM/XA SP, a dump may result from any of the following:

- A hardware error
- A software error
- The system operator initiating a system restart
- A user issuing the CP command VMDUMP.

The way you proceed to narrow your search for the cause of the dump is to use the following two main steps:

Problem determination Finding out whether the problem is caused by hardware or software.

Problem source identification Isolating the problem to a particular component of the software system.

Once you identify the cause of the error as being within the software, you can use error messages and the symptom record to refine your analysis of the problem further.

These steps are discussed below.

Problem Determination

The goal of problem determination is to discover whether the dump was the result of a hardware or software error. Sometimes the clues are obvious. For example, if a machine check, channel check, or storage check preceded the dump, the problem most likely is a hardware error. If the dump is a CP abend dump, the cause is very likely a software error. When a restart dump has been taken, you probably will have to examine the conditions of both the hardware and software to make the determination.

Once you have determined whether the problem is hardware or software, problem determination is complete. From this point on, the discussion focuses on performing problem source identification for software.

Performing Problem Source Identification

Problem source identification is the second step in analyzing a software problem. It consists of isolating the cause of a problem to a particular component of the software system. The VM/XA System Product has four components:

- The control program (CP)
- The conversational monitor system (CMS)
- The dump viewing facility
- The group control system (GCS).

In addition to these three components, there are several program products that can run in the VM/XA SP environment. Problem source identification is complete when you have determined the particular component or program in which the error occurred.

Using Error Messages

One method of determining where the problem occurred is to examine any error messages in the dump. These messages usually identify the immediate cause of the dump. For example, a CP abend dump is identified as such by a message. CMS and the dump viewing facility also issue messages when they detect errors. Similar messages may be sent to the system operator's console. You can look at these messages in the *VM/XA SP: System Messages and Codes Reference* and in the messages section, "Chapter 5. Messages and Message Summaries" on page 159, of this manual. The message descriptions tell which component failed and briefly describe the error conditions encountered.

If a message indicates that an error occurred in CP, you can use the message code to determine which module in CP encountered the error. The scenarios in this chapter describe this in more detail.

Using the Symptom Record to Identify Duplicate Problems

Whenever CP terminates abnormally, it creates a symptom record which is included in the dump. The symptom record summarizes data about the state of the system when the dump was taken. The dump viewing facility can format the symptom record for display and printing. See the SYMPTOM subcommand of DUMPSCAN in Chapter 4 of this manual for more information on symptom records.

You can use the keywords and formatted data from the symptom record to determine whether the problem has occurred on your system before. You can compare the symptom record data in a new dump to symptom records in dumps that already exist, keyword by keyword. A match on all the data indicates that the new problem may be a duplicate.

You can search for duplicate problems using the dumps on file at your installation. You can also ask the IBM Support Center to search the IBM data base. This data base contains information about all the problems reported to IBM by VM/XA SP users.

Commands of the Dump Viewing Facility

The dump viewing facility provides five commands. The commands are MAP, ADDMAP, DUMPSCAN, PRTDUMP, and TRACERED.

Note: The following information on the MAP and ADDMAP commands is required for non-CP dumps. For CP dumps, you can use the dynamic map building function of the DUMPSCAN and PRTDUMP commands. For more data on when you do and do not need to use the MAP and ADDMAP commands, see "Load Maps for CP Dumps" on page 12.

1. The MAP command compresses the VM/XA SP load map created at system generation time into a format that the dump viewing facility can process. The compressed module map is used to correlate module and entry point names with addresses in the dump. You do not have to use the MAP command for CP dumps. Both the DUMPSCAN and PRTDUMP commands dynamically create and append a module map to a CP dump.

2. Use the ADDMAP command to append the compressed module map to the CMS file containing the dump processed by the DUMPLOAD command. (For more information about the CMS DUMPLOAD command, refer to the *VM/XA System Product: CMS Command Reference* manual). ADDMAP processing also resolves the addresses of pageable CP modules that were in storage at the time of the dump. If you use the dynamic map build and append functions of the DUMPSCAN and PRTDUMP commands for a CP dump, you do not need to use the ADDMAP command.

3. Use the DUMPSCAN command to interactively view CP, stand-alone, and virtual machine dumps. The DUMPSCAN subcommands are described in Chapter 4 of this manual. For CP dumps, if a module map is not already appended to the dump, DUMPSCAN dynamically creates and appends one.

4. Use the PRTDUMP command to obtain a printed copy of any or all of the summary reports available through the dump viewing facility. Each summary report contains data the problem solver most frequently requires. For CP dumps, if a module map is not already appended to the dump, PRTDUMP dynamically creates and appends one.

5. Use the TRACERED command to read and create a CMS or print file containing the trace data saved to tape or DASD by the CP TRSAVE command. You may specify certain selection criteria be applied to the trace entries. And, you may designate whether the trace information is output as either hexadecimal or formatted data.

   You may also use TRACERED to collect GCS guest trace records in a simulated OS QSAM file. This allows programs, such as ACF/TAP licensed program, to process the GCS guest trace data.

Figure 3 illustrates the command structure for the ADDMAP, DUMPSCAN, MAP, and PRTDUMP commands. Figure 4 illustrates the command structure for the TRACERED command.

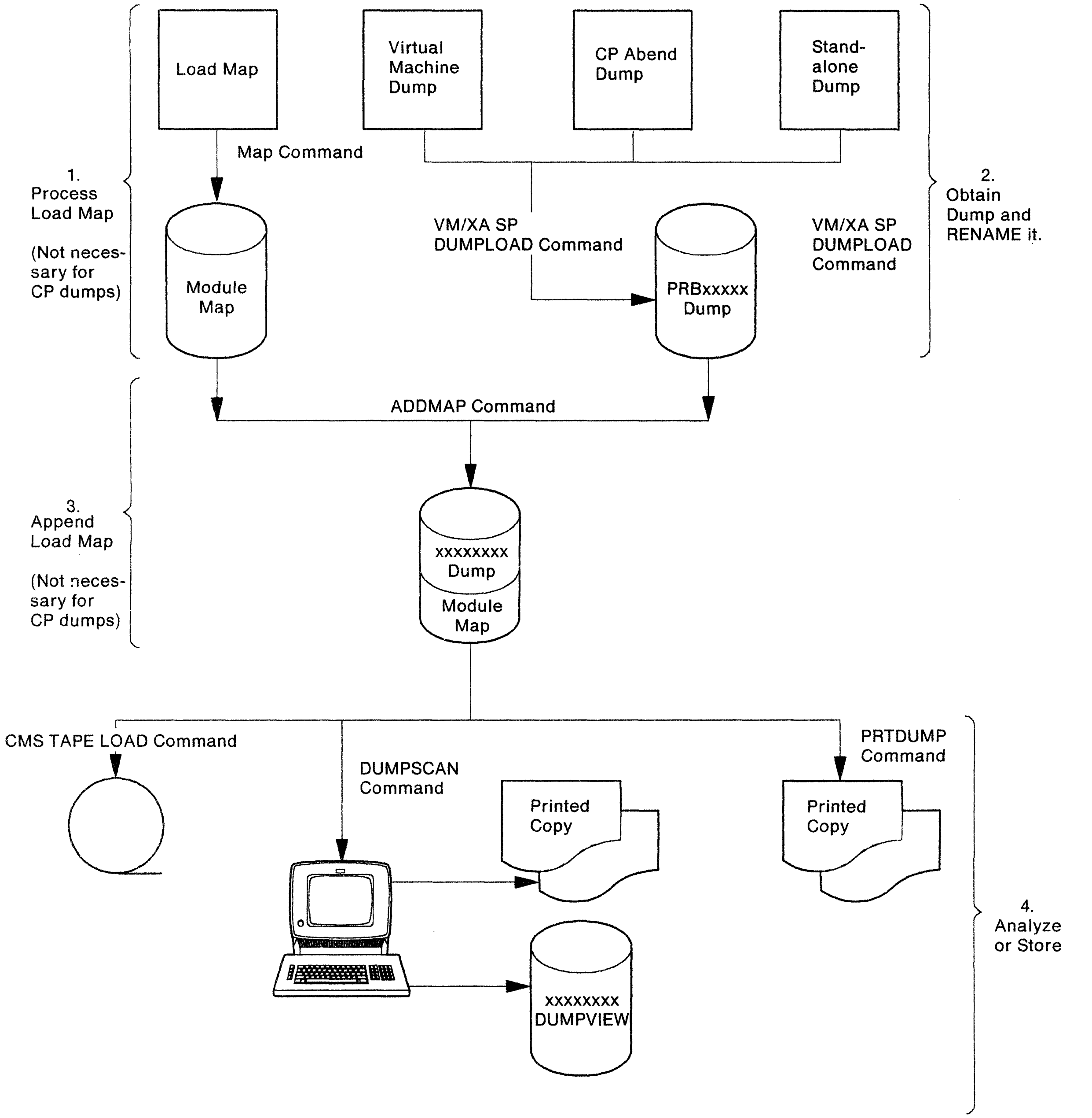


Figure 3. Command Structure for the Dump Viewing Facility's ADDMAP, DUMPSCAN, MAP, and PRTDUMP Commands

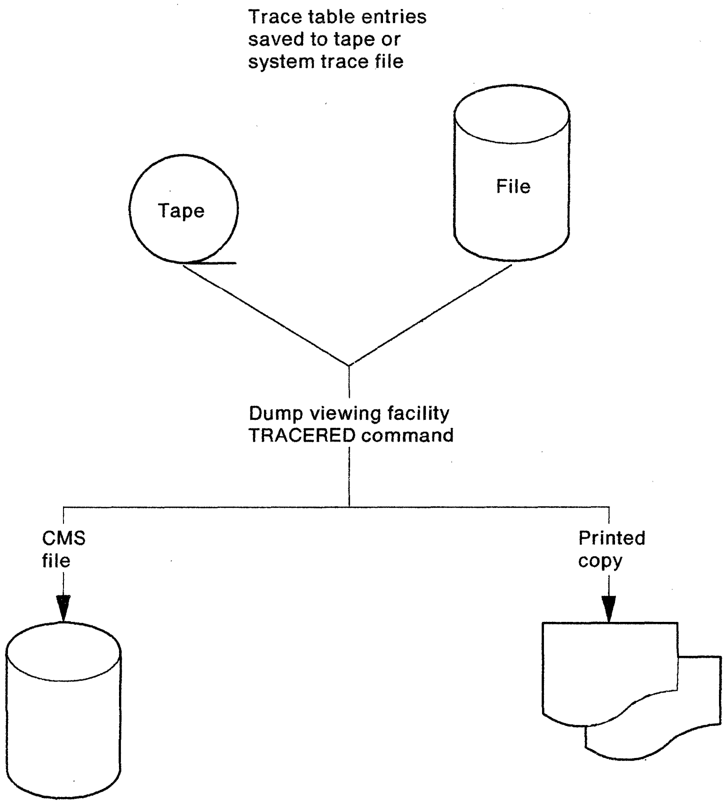

Trace table entries saved to tape or system trace file
Tape
File
Dump viewing facility TRACERED command
CMS file
Printed copy

Figure 4. Command Structure for the Dump Viewing Facility's TRACERED Command

Servicing the Dump Viewing Facility

Corrective and preventive maintenance of the dump viewing facility is performed using standard VM/XA SP procedures. Refer to the *VM/XA System Product: Installation and Service* manual for information on these maintenance procedures.

Writing Dump Data to Tape

To ship dump data to IBM service personnel or for offline storage, you can write dump data to tape using existing CMS commands.
Refer to Appendix D, "Dumping the Abend Dump to Tape" on page 233 for more information on dumping to tape.

Chapter 2. Usage Guide

Preparing a Dump for Use with the Dump Viewing Facility

If you wish to use the dump viewing facility to analyze a dump, you first have to prepare the dump by following these steps:

1. When VM/XA SP becomes available again, log on to a userid.

   The userid must:

   - Be the userid to which the dump was sent, if the dump was sent to a virtual reader
   - Have enough unused space on its A-disk to hold the dump file.

     Refer to the section "Storage Requirements" on page 3 for the amount of storage required for dumps.

2. Use the DUMPLOAD command to load the dump into a CMS file.

   Issuing DUMPLOAD puts the CMS file containing the dump onto the A-disk of the receiving userid. The file will be named PRB*xxxxx* DUMP A, where *xxxxx* is a number from 00000 to 00009, depending on the names of the dump files already on the A-disk.

   If you wish to print an unformatted dump, use the PRINT option of the DUMPLOAD command. For information on the VM/XA SP DUMPLOAD command, see the *VM/XA SP: CMS Command Reference* manual.

3. Rename the dump file by using the CMS RENAME command.

   When renaming the dump file, keep the general format of *filename* DUMP A. The filename can be from one to eight characters in length. For information on the VM/XA SP RENAME command, see the *VM/XA SP: CMS Command Reference* manual.

   We recommend you rename the file because the DUMPLOAD command restricts the file names to PRB00000 through PRB00009. The dump viewing facility allows file names in the form *xxxxxxxx* where *xxxxxxxx* is a 1- to 8-character string that may consist of the characters 0−9, A−Z, @, #, −, _, +, :, and $.

   A file that has a name of PRB00000 should be renamed before using the dump viewing facility. If dump files PRB00000 through PRB00009 already exist, DUMPLOAD erases PRB00000 before loading the current dump into a CMS file.

4. Create and append a module map to the dump you wish to examine.

   There are two ways to do so.

   - Use the MAP and ADDMAP commands.

     Using the MAP/ADDMAP method requires that a load map be created and then converted to a module map. See the sections below for more information on load maps and module maps.

   - For CP dumps hard abend and standalone dumps only, use the dynamic map build function of DUMPSCAN or PRTDUMP.

Using Load Maps

Load Maps for CP Dumps

In this section, any reference to a CP dump is to a hard abend or standalone dump.

There are certain instances when you need a copy of the CP load map created at system generation time:

- If you want to see certain data areas; for example, HCPSYSCM.
- If you want to see what modules existed at system generation time but were paged out when the dump was taken.

Load Maps for VM Dumps

The dump viewing facility does not dynamically generate module maps for VM dumps. Therefore, it requires a VM load map.

Creating Load Maps

In this section, any reference to a CP dump is to a hard abend or standalone dump.

As stated above, you need to create a load map in two instances for CP dumps, and in every instance for VM dumps.

Note: Create and save a new load map whenever a new system is generated in all cases for using a VM load map and in the two cases when you need a CP load map. If you don't, the module map that is converted from the load map will not match the dump to which it is appended, and incorrect data will be presented.

CP Load Maps

To create a CP load map, see the *VM/XA SP Installation and Service* manual.

VM Load Maps

To create a CMS or GCS load map, see the *VM/XA SP Installation and Service* manual. To create a PVM load map, see the *VM/SP Pass-Through Guide and Reference*. To create an RSCSNET load map, see *VM/SP RSCS Networking Reference and Operations* manual. To create an RSCSV2 load map, see the *VM/SP RSCS Networking Version 2 Planning and Installation* manual.

Creating Module Maps

In this section, any reference to a CP dump is to a hard abend or standalone dump. The following section explains how to create and append module maps for use with the dump viewing facility.

CP Module Maps

There are two methods by which you can convert a CP load map into a module map and append it to a CP dump:

1. Allow the dump viewing facility to create and append the module map dynamically by using the DUMPSCAN or PRTDUMP command
2. Use a two-step process that employs the MAP and ADDMAP commands with a CP load map as input.
   a. Use the MAP command to convert a load map into a module map which then can be appended to the dump.
   b. Use the ADDMAP command to append the module map to the dump you specify in the command. For CP dumps only, ADDMAP translates the addresses of pageable CP modules from virtual addresses to real addresses.

VM Module Maps

There is one method by which you can convert a VM load map into a module map and append it to a VM dump. That method is a two-step process that employs the MAP and ADDMAP commands. See the section above, on page 13, for a description of what the MAP and ADDMAP commands do. For more information on use of the commands, see "Chapter 3. Command Reference" on page 33.

The dump file is now ready for use with the dump viewing facility. One task you may wish to perform frequently is viewing the dump. The use of the DUMPSCAN command for viewing a dump is explained next.

Viewing Dumps

To view a dump, access it using the dump viewing facility DUMPSCAN command. For example, if the dump you wish to analyze is in the file named HUNGUSER DUMP A, you can enter the command:

```
dumpscan hunguser
```

When the dump file has been accessed, you are notified by the READY status message. You are in an XEDIT environment ready to enter DUMPSCAN subcommands on the command line.

All the commands used to view dump data in the dump file are subcommands of the DUMPSCAN command. They are explained in Chapter 4. Some of these subcommands are used in the dump analysis scenarios in this chapter. See "Scenario 1: Analyzing a CP STK017 Abend Dump" on page 20 and "Scenario 2: Analyzing a CP FRE016 Abend Dump" on page 24.

Using the Session File

The dump viewing facility makes use of full screen XEDIT functions so that you can scroll back and forth through dump data in full-screen mode. You can scroll through any data previously viewed without reissuing the command to display the data. You can edit your session file while in the dump viewing facility. This annotation feature allows you to make comments within a dump session file before passing the dump on to the next level of problem determination.

The dump viewing session can be filed by issuing the XEDIT subcommand FILE. The session is stored on the user's A disk. In the future, if you want to view the same dump, this file will be reactivated and this new session will then be appended to the *dumpname* DUMPVIEW file containing previous session or sessions.

Viewing Dumps of Licensed Programs

The dump viewing facility provides additional support for viewing data particular to specific virtual machine dumps. This support is comparable to that of VM/SP Interactive Problem Control System (IPCS). With this support a user is able to make use of IPCS formatting routines to find and format data in licensed programs′ virtual machine dumps and display it via the dump viewing facility.

The dump viewing facility provides this support for dumps taken from the following:

- Conversational monitor system (CMS)
- Group control system (GCS)
- Pass-Through Virtual Machine (PVM)
- Remote Spooling Communications System Version 1 (RSCSNET)
- Remote Spooling Communications System Version 2 (RSCSV2).

See Appendix E for more information regarding this support.

Analyzing Dumps

The discussion of software errors on page 4 pointed out that software problems have a variety of symptoms. Two examples of symptoms are used in this chapter′s scenarios to illustrate one way of analyzing CP abend dumps, using the dump viewing facility.

Analyzing Trace Tapes

The dump viewing facility also enables you to view trace tables saved to tape. The CP command TRSAVE saves the trace tables; the dump viewing facility command TRACERED formats the trace tables if you request that it be done and it formats the trace tables into a CMS or print file. See page 46 for more specific information on the use of the TRACERED command.

Analyzing Trace Data as System Files

The dump viewing facility processes data trace information as well as CP trace table data. In addition, you can merge CP trace table information with data trace information and view it in chronological sequence. VM/XA SP records CP system trace table data in system trace files or to tape.

Writing DUMPSCAN Macros

The DUMPSCAN subcommands help you to interactively analyze dump data, and in most situations there is a subcommand that will give you the needed data in a usable format. However, you can write macros to customize and automate the powers of DUMPSCAN, creating your own powerful tools to analyze dump data. By writing macros, you can:

- Expand the basic subcommand set
- Tailor the basic subcommand output for:
  - Dump data summary reports in a specific format
  - More readable format, such as changing technical jargon to English
  - Additional annotations
  - Additional time/date/tracking information.
- Eliminate repetitive tasks.

This section explains how to write XEDIT macros using DUMPSCAN subcommands and describes the DVFXEDIT profile. You should be familiar with the Restructured Extended Executor(REXX) language. See the *VM/XA System Product Interpreter User's Guide* and the *VM/XA System Product: System Product Interpreter Reference* for more information on REXX. You should also be familiar with XEDIT. See the *VM/XA System Product Editor User's Guide* and the *VM/XA System Product: System Product Editor Command and Macro Reference* for more information on XEDIT.

This section describes some usage of XEDIT subcommands, but it is not intended to give you a complete guide to writing macros.

What Is a DUMPSCAN Macro?

A DUMPSCAN macro is a EXEC file you invoke from the DUMPSCAN environment. This environment exists whenever DUMPSCAN is being used.

You execute a macro the same way you execute a DUMPSCAN subcommand. Type the macro name on the command line and press the ENTER key. A macro can be executed by entering only its name or its name and any parameters needed for its execution.

A macro file can contain:

- DUMPSCAN subcommands
- XEDIT subcommands
- REXX instructions
- Other REXX or EXEC2 EXECS
- CMS and CP commands.

A macro file should not contain:

- Any dump viewing facility command
- Any CMS command that does not run in subset mode or runs in the user area.

Generally it is safe to run any EXEC from within the DUMPSCAN environment. But, reexecuting DUMPSCAN from within DUMPSCAN or executing any other command that runs in the user area will have unpredictable results.

Creating a Macro File

A macro is a normal CMS file. It may be created in any of the ways that CMS provides for file creation. Like any CMS file, a macro is identified by filename, filetype, and filemode. Its creation must conform to the following rules:

- The filename may be 1-to-8 alphabetic characters.
- The filetype should be XEDIT or EXEC.
- The filemode can be any of the disks to which you have write access, usually your A-disk.

Using DUMPSCAN Subcommands in a Macro

A macro can contain any DUMPSCAN subcommand. Most subcommands look for specific data in the dump, format it, and write it to the session record. However, the DUMPSCAN macro subcommands are meaningful only from within a macro since they pass information to the system product interpreter.

What Is an Environment?

When you write a macro, you need to know which command processor will interpret your command. The system product interpreter looks at the instructions first within a macro. If the instructions are not interpreter instructions, they are passed to the environment for interpretation. The environment is the command processor that gets the instructions after the system product interpreter has done any symbolic substitution.

You can specify the environment with the REXX instruction ADDRESS:

- ADDRESS SCAN causes the macro instruction to be passed to DUMPSCAN.
- ADDRESS XEDIT causes the macro instruction to be passed to XEDIT.
- ADDRESS CMS causes the macro instruction to be passed to CMS.
- ADDRESS by itself causes the macro instruction to be passed to the default environment.

The default environment is created when you use XEDIT as the filetype for your macro. XEDIT is the default environment. When you use EXEC as the filetype for your macro, then CMS is the default environment.

DUMPSCAN Macro Subcommands FINDSTRG, READSTRG, and NOTE

FINDSTRG and READSTRG are subcommands that you can only use from within a macro. READSTRG allows you to read from the dump and place that data directly into a REXX variable. FINDSTRG allows you to locate data in the dump and put the address of the matching data into a REXX variable. With these two commands, you can extract and format any addressable data within the dump.

NOTE is a subcommand for annotating the session file. It can also be used to annotate the print file produced by DUMPSCAN subcommands when the PRINT ON subcommand is used.

The remaining DUMPSCAN subcommands can be used in a macro, but the output from those subcommands is placed in the session file. If you need this subcommand output, you must use XEDIT subcommands to get the records into the program stack or into REXX variables. See "DUMPSCAN Macro Example" on page 17 for an example of how the output of the CPU subcommand is taken from the session file. Once the needed data is in a macro variable, the subcommand is removed from the session file. Not all DUMPSCAN subcommands are used in the examples, but the technique is the same.

Communicating between the Editor and the Interpreter

The READ and EXTRACT subcommands of XEDIT can supply the macro with data from the DUMPSCAN command line or from the session file. The READ subcommand takes information from the screen and places it in the program stack. The information in the stack can be the command line, changed lines, prefix area, and program function key definitions (PF keys). The macro gets the information from the program stack with the REXX PULL instruction. The EXTRACT subcommand can supply a macro with information about internal XEDIT variables or about file data. The information is returned in one or more REXX variables, which can be examined or used by the macro.

READ Subcommand

When a READ subcommand is issued from a macro, the editor redisplays the current screen and waits for the user to press ENTER or a PF key. After a key is pressed, the requested data is placed in the program stack.

Operands of the READ subcommand can be used to specify how much information is placed in the program stack. A subsequent REXX PULL instruction assigns the data to program variables, and the macro continues.

EXTRACT Subcommand

EXTRACT is issued from a macro. The information is returned in REXX variables. The EXTRACT subcommand returns information about editor variable settings. These are needed when the macro changes an XEDIT variable and restores it before ending. The example that follows uses the EXTRACT subcommand to get records from the data file and to determine data file parameters such as the number of records in the file and the position of the current line. The EXTRACT command has many options. For a complete discussion of the options read the *VM/XA System Product: System Product Editor Command and Macro Reference.*

Displaying Data on the DUMPSCAN Screen

To move prompts and messages from the macro to the DUMPSCAN screen you can use the XEDIT subcommands MSG, EMSG, CMSG, and INPUT.

MSG	Displays a message on the message line
EMSG	Displays a message on the message line and sounds the alarm
CMSG	Displays a message on the command line
INPUT	Puts the message or text in the session file following the current line and makes the new line the current line.

DUMPSCAN Macro Example

The following example shows you how to use a combination of XEDIT subcommands and DUMPSCAN subcommands to create a new DUMPSCAN function. In this case, a new DUMPSCAN subcommand is created by a macro which displays 16 bytes from a specified offset within the prefix page. To get the same information using DUMPSCAN subcommands from the command line would require you to issue the CPU subcommand, write down the prefix page address, add the offset to that address, then issue the DISPLAY subcommand.

```
00001 /*DUMPSCAN Subcommand Macro                        */
00002 /* Display the data at the specified offset from */
00003 /* the failing processor prefix page.            */
00004 /*                                               */
00005 Parse arg offset .
00006 Parse source . . macroname .
00007 'EXTRACT/LINE/MSGMODE'
00008 If offset = '' Then Do
00009   Emsg 'Command format is:' macroname 'offset'
00010   Cmsg macroname
00011   exit
00012   end
00013 'EXTRACT/SIZE/'
00014 Address SCAN 'CPU'
00015 ':'size.1+1 'EXTRACT/CURLINE/'
00016 parse var curline.3 . . . . . . . pfxpgad .
00017 'SET MSGMODE OFF'
00018 ':'size.1+1 'DEL *'
00019 'SET MSGMODE' MSGMODE.1 MSGMODE.2
00020 ':'line.1
00021 Address SCAN 'Display' d2x(x2d(pfxpgad)+x2d(offset)) 'F OFFSET'
00022 exit
```

Figure 5. A DUMPSCAN Macro Example

00001 /* DUMPSCAN Subcommand Macro */

An interpreter comment. The first line must be a comment.

00002 /* Display the data at the specified offset from */
00003 /* the failing processor prefix page. */
00004 /* */
00005 Parse arg offset .

This instruction says to take the first operand and assign it to the variable *offset*. The period says to disregard any other operands.

00006 Parse source . . macroname .

Here the macro gets its name from CMS. Again the periods mean that any other file information is not needed.

00007 'EXTRACT/LINE/MSGMODE'

The EXTRACT XEDIT subcommand will return the line number at the XEDIT current line and the XEDIT message settings.

00008 If offset = ″ Then Do

Check for the required operand and if it was not entered, then issue an error message to the user. DO is the first statement of a series of instructions. This set of statements ends at line 00012.

00009 Emsg 'Command format is:' macroname 'offset'

Use XEDIT EMSG to tell the user the correct way to use this macro.

00010 Cmsg macroname

Use XEDIT CMSG to put the macro name back on the command line in case the user wants to try the macro again.

00011 exit

Since the macro cannot do anything, the EXIT instruction will cause control to return to the caller.

00012 end
END is the last instruction of a DO statement, this one started at line 00008.

00013 'EXTRACT/SIZE/'
Use the XEDIT EXTRACT subcommand to find the size of the current file.

00014 Address SCAN 'CPU'
The ADDRESS command changes the environment to SCAN for the following subcommand. SCAN is the name of the DUMPSCAN environment. CPU is a DUMPSCAN subcommand that lists the CPU names and the prefix page address of all CPUs contained in the dump.

00015 ':'size.1 + 1 'EXTRACT/CURLINE/'
Reset the current line to the CPU subcommand first output line and EXTRACT the line into the CURLINE.3 variable.

00016 parse var curline.3 . . . . pfxpgad .
Break the curline variable into its parts and ,ignoring some parts, assign the prefix page address to the *pfxpgad* variable.

00017 'SET MSGMODE OFF'
Use XEDIT SET to suppress any messages.

00018 ':'size.1 + 1 'DEL *'
Remove the CPU subcommand output from the file.

00019 'SET MSGMODE' MSGMODE.1 MSGMODE.2
Turn the messages back to their previous settings.

00020 ':'line.1
Reset the current line to its position when we started.

00021 Address SCAN 'Display' d2x(x2d(pfxpgad) + x2d(offset)) 'F OFFSET'
Switch to the DUMPSCAN environment, and after the interpreter works through the calculation, issue the DUMPSCAN DISPLAY subcommand with the OFFSET operand for 15 bytes.

00022 exit
EXIT indicates the macro is finished.

DVFXEDIT Profile

DVFXEDIT XEDIT is a macro distributed with the dump viewing facility that modifies a standard XEDIT session to the special DUMPSCAN format. This macro can be modified to change the session format to your preference. You should not change the definition of the ENTER key.

Assigning Program Function Keys to DUMPSCAN Subcommands

The PF keys default to XEDIT functions in the DUMPSCAN environment. The PF keys are assumed to be for XEDIT functions. You can assign keys to DUMPSCAN subcommands or DUMPSCAN subcommand macros by putting an XEDIT SET command in a copy of the DVFXEDIT XEDIT macro that you keep on your A-disk.

For example, you may want to assign PF keys 4 and 5 to the DUMPSCAN FORWARD and BACKWARD subcommands. You would put the XEDIT subcommands SET PF04 SCAN BACKWARD and SET PF05 SCAN FORWARD in your DVFXEDIT XEDIT macro. SET is an XEDIT subcommand name that changes XEDIT system variables. PF04 is the name of an XEDIT system variable. SCAN is the CMS command that gives you access to DUMPSCAN subcommand processing. BACKWARD is the name of the DUMPSCAN subcommand that displays addresses lower than the last addresses displayed.

Scenario 1: Analyzing a CP STK017 Abend Dump

In this scenario, CP terminated abnormally with an STK017 abend code and a dump. The steps that follow suggest one way this problem could be analyzed. The analysis is designed to meet three objectives: problem determination, problem source identification, and information gathering.

Step 1: Checking the Error Message and Symptom Record

There are two methods for determining the reason for an abend:

- Checking the error and informational messages displayed on your screen, or
- Using the SYMPTOM subcommand of the DUMPSCAN command in the VM/XA SP dump viewing facility.

Checking the Error and Informational Messages

The system operator was notified of the problem by the message:

```
HCPDMP908I  SYSTEM FAILURE ON CPU 0002, CODE - STK017
```

HCP indicates that the message is from the control program (CP). **DMP** indicates that module DMP wrote the message. **908I** is the message number. Using this information, look up the message in the *VM/XA System Product: System Messages and Codes Reference*. If you're unsure of where to look, check the Preface for the organization of the book. For this scenario, the explanation in *Messages and Codes* indicates that CP encountered a severe software failure that caused a dump to be taken.

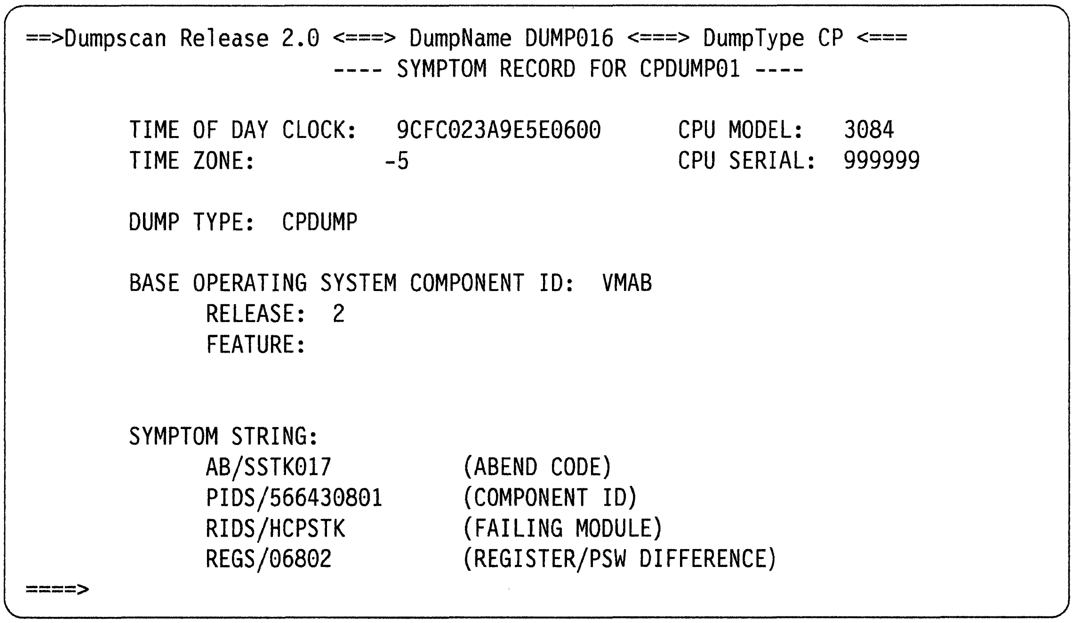
The message text also indicates that an abend occurred. The CP abend code in the message is STK017.

You can find the explanation for this particular abend code in, Section 1: CP Abend Codes of the *VM/XA System Product: System Messages and Codes Reference* manual. The explanation for a STK017 code is that the value for the VMDBK scheduler list identifier is unrecognizable and the probable cause for the abend is that the VMDBK has logged off.

Using the SYMPTOM Subcommand of DUMPSCAN in the Dump Viewing Facility

In order to use any commands or subcommands in the dump viewing facility, you first have to load the dump you wish to view into a CMS file by using the CMS command DUMPLOAD. Then you can proceed to use the dump viewing facility's other subcommands.

In this example, assume you already have loaded the dump you want to examine in a CMS file and assume you have issued the DUMPSCAN command for the dump. Proceed by entering the subcommand SYMPTOM to display formatted symptom record information. Figure 6 shows an example of the output you receive if you enter the SYMPTOM subcommand for an STK017 abend.



```
==>Dumpscan Release 2.0 <===> DumpName DUMP016 <===> DumpType CP <===
                        ---- SYMPTOM RECORD FOR CPDUMP01 ----

        TIME OF DAY CLOCK:   9CFC023A9E5E0600      CPU MODEL:   3084
        TIME ZONE:          -5                     CPU SERIAL:  999999

        DUMP TYPE:  CPDUMP

        BASE OPERATING SYSTEM COMPONENT ID:  VMAB
              RELEASE:  2
              FEATURE:


        SYMPTOM STRING:
              AB/SSTK017          (ABEND CODE)
              PIDS/566430801      (COMPONENT ID)
              RIDS/HCPSTK         (FAILING MODULE)
              REGS/06802          (REGISTER/PSW DIFFERENCE)
====>
```

Figure 6. Output Format of the SYMPTOM Subcommand

The symptom string information is interpreted as follows:

- The module that issued the abend is **STK** and the reason code for the abend is **017** as indicated by the **ABEND CODE** field.
- The component identifier for this product is **566430801** as indicated by the **COMPONENT ID** field.
- The identifier of the failing module is **HCPSTK** as indicated by the **FAILING MODULE** field.
- **Register 6** had an offset of **802** from the current PSW at the time the dump was taken as indicated by the **REGISTER/PSW DIFFERENCE** field.

At this point, problem determination and problem source identification are complete. The error has been identified as a software error, and it occurred in the CP component VM/XA SP. Your next step is to gather more information about the abend.

Step 2: Analyzing the Trace Table

Aside from its SYMPTOM subcommand, the dump viewing facility can help you gather additional data about the circumstances of this failure.

The CP trace table contains information about activity in the system. The processing flow in CP just prior to the abend is in the most recent trace entries. Use the TRACE subcommand of DUMPSCAN to view these entries on your display screen. If you enter TRACE FORMAT, the subcommand formats and displays the last trace entries merged from all processors.

The last trace entry made before CP abended is displayed at the bottom of your screen. As Figure 7 shows, the trace code is 0200.

```
 •
 •
 •
0200 CPU 0002  SVC INTERRUPTION              TOD FF40386DC400  ADR 01F3A520
     ABEND CODE    ILC      INT CODE    SVC OLD PSW
     STK017        0004     0002        000C1000 8099460E
```

Figure 7. Last Trace Entry in the CP Trace Table with Abend STK017

This is the trace entry for the CP abend. You can see that the information in the entry corresponds to the primary symptom in the dump: the STK017 abend.

The next-to-last trace entry is of greater interest since it shows what CP processing occurred just prior to the abend; see Figure 8. It is trace code 3300, STACK CPEBK:

```
 •
 •
 •
3300 CPU 002  STACK CPEBK                 TOD FF40386DC400 ADR 01F3A500

     STATE   SLIST   CPEXSCHC   VMDBK ADDR  CPEBK ADDR   MOD/DISP ADDR
     2C      00      80         01BD7000    01F5AD10     LCK/1D0

0200 CPU 0002  SVC INTERRUPTION             TOD FF40386DC400  ADR 01F3A520
     ABEND CODE     ILC     INT CODE  SVC OLD PSW
     STK017         0004    0002      000C1000 8099460E
```

Figure 8. Second-to-Last Trace Entry from the CP Trace Table

Module HCPLCK requested that a CPEBK be stacked for eventual execution by CP. The call to HCPSTK occurred at hexadecimal 1D0 bytes from the beginning of HCPLCK.

Step 3: Finding the VMDBK

Recall from the description of the STK017 abend that the VMDBK may not exist because the user has logged off. You can find the address of a VMDBK in the 3300 STACK CPEBK trace entry. It represents the userid for which the CP execution task (CPEBK) was to be performed. The VMDBK's address is 1BD7000, and you can display it using the VMDBK subcommand:

```
vmdbk at 1bd7000
```

This subcommand should display a summary of the VMDBK at the address 1BD7000. But in this case, because the user has logged off, the response is the following message:

```
THE VMDBK PAGE WITH ADDRESS 01BD7000 IS NOT IN THE DUMP
```

Although the address of the VMDBK points to a page that isn't in the dump, there is a way to check the status of the page. The frame table (FRMTE) contains the status of every page frame in real storage and the FRMTE is always in storage. You can determine the status of the VMDBK page with the FRAMETBL subcommand:

```
 frametbl 1bd7000
```

This subcommand causes the following information to be displayed on your screen:

```
FRAME TABLE ENTRY 303000            FRAME USE = AVAILABLE
PAGE TABLE ENTRY = 00000000
FORWARD POINTER = 01BD8C70     BACKWARD POINTER = 01AE7F90
STATIC FLAGS    = 00  NO FLAGS SET
DYNAMIC STATES  = 00  NO FLAGS SET
SERIALIZE FLAGS = 80  ON AVAILABLE QUEUE
```

Figure 9. Determining the Status of the VMDBK Page with the FRAMETBL Subcommand

You can see that the frame is on the available list for the dynamic paging area. It is possible that the user did log off, and that the page was returned to the dynamic paging area for use as a user page. This would account for the frame flag settings and the AVAILABLE frame use.

You can see that the possibility that the user logged off is supported by the value of SLIST displayed in the 3300 STACK CPEBK trace entry. SLIST is the VMDBK data element named VMDSLIST, and it identifies the scheduler list to which the VMDBK is currently assigned. You can locate the definition of VMDSLIST by examining the CP COPY blocks (the listing that contains the definitions of CP control blocks), and by locating the VMDBK COPY file. Locate VMDSLIST in the VMDBK. The valid values for VMDSLIST are:

```
USER IS IN THE DISPATCH LIST
USER IS IN THE ELIGIBLE LIST
USER IS IN THE DORMANT LIST
VIRTUAL MACHINE IS NOT IN A LIST
```

The VMDBK must be on one of the first three lists if it is logged on. The only time it is not on a scheduler list is when it is logged off.

Since the SLIST value indicates the virtual machine is not in a list, the user must have been logged off.

Step 4: Summarizing the Dump Analysis

By examining these two trace entries and the frame table, you have found the immediate reason for the STK017 abend. You can summarize the diagnostic information in the dump as follows:

1. CP terminated with an STK017 abend.
2. Immediately before the abend, HCPLCK called HCPSTK to stack a CPEBK for the VMDBK at location hex 1BD7000.
3. The SLIST value in the VMDBK contains zeros, which indicates the VMDBK (userid) was not logged on to the system when HCPSTK was called.
4. The page that contained the VMDBK has been released for use in the dynamic paging area.

You now know enough information to begin searching for duplicate problems at your installation. If no duplicates are found, your next step is to contact the IBM Support Center to report the problem and its symptoms.

Scenario 2: Analyzing a CP FRE016 Abend Dump

In this scenario, CP terminated abnormally with a FRE016 abend code and a dump. The steps that follow suggest one way this problem could be analyzed. The analysis is designed to meet three objectives: problem determination, problem source identification, and information gathering.

Step 1: Checking the Error Message and Symptom Record

There are two methods for determining the reason for an abend:

- Checking the error and informational messages displayed on your screen, or
- Using the SYMPTOM subcommand of the DUMPSCAN command of VM/XA SP dump viewing facility.

Checking the Error and Informational Messages

The system operator was notified of the problem by the message:

```
HCPDMP908I SYSTEM FAILURE ON CPU 0000, CODE - FRE016
```

HCP indicates the message is from the control program (CP). **DMP** indicates that module HCPDMP wrote the message. **908I** is the message number. Using this information, look up the message in the *VM/XA System Product: System Messages and Codes Reference*. If you're unsure of where to look, check the Preface for the organization of the book. For this scenario, the explanation in *Messages and Codes* indicates that CP encountered a severe software failure that caused a dump to be taken.

The message text also indicates that an abend occurred. The CP abend code in the message is FRE016.

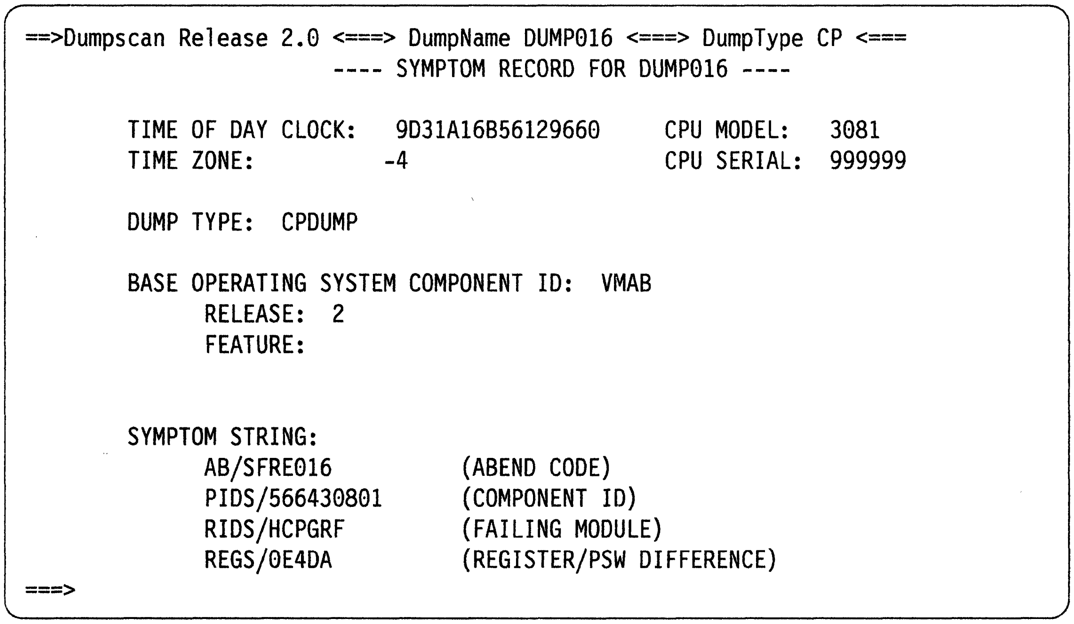
You can find the explanation for this particular abend code in, Section 1: CP Abend Codes of the *VM/XA System Product: System Messages and Codes Reference*. The explanation for a FRE016 code is that the requester of the free storage block has written beyond the free storage block. The free storage TRAILER is destroyed.

Using the SYMPTOM Subcommand of DUMPSCAN in the Dump Viewing Facility

In order to use any commands or subcommands in the dump viewing facility, you first have to load the dump you wish to view into a CMS file by using the CMS command, DUMPLOAD. Then you can proceed to use the dump viewing facility's DUMPSCAN subcommands.

In this case, we have renamed the file, DUMP016. For this example, assume you already have loaded the dump you want to examine in a CMS file and proceed to use the SYMPTOM subcommand to display formatted symptom record information.

An example of the output you receive if you enter the SYMPTOM subcommand for an FRE016 abend follows:



```
==>Dumpscan Release 2.0 <===> DumpName DUMP016 <===> DumpType CP <===
                       ---- SYMPTOM RECORD FOR DUMP016 ----

        TIME OF DAY CLOCK:   9D31A16B56129660     CPU MODEL:   3081
        TIME ZONE:          -4                    CPU SERIAL:  999999

        DUMP TYPE:  CPDUMP

        BASE OPERATING SYSTEM COMPONENT ID:  VMAB
              RELEASE:  2
              FEATURE:


        SYMPTOM STRING:
              AB/SFRE016          (ABEND CODE)
              PIDS/566430801      (COMPONENT ID)
              RIDS/HCPGRF         (FAILING MODULE)
              REGS/0E4DA          (REGISTER/PSW DIFFERENCE)
===>
```

Figure 10. Output Format of the SYMPTOM Subcommand

The symptom string information is interpreted as follows:

- The module that issued the abend is **FRE** and the reason code for the abend is **016** as indicated by the **ABEND CODE** field.
- The component identifier for this product is **566430801** as indicated by the **COMPONENT ID** field.
- The identifier of the failing module is **HCPGRF** as indicated by the **FAILING MODULE** field.
- **Register 14** had an offset of **4DA** from the current PSW at the time the dump was taken as indicated by the **REGISTER/PSW DIFFERENCE** field.

At this point, problem determination and problem source identification are complete. The error has been identified as a software error, and it occurred in the CP component of VM/XA SP. Your next step is to gather more information about the abend.

Step 2: Analyzing the Trace Table

Use the dump viewing facility to gather additional data about the circumstances of the failure. You can use the TRACE subcommand of DUMPSCAN to view these entries on your display screen. You first must enter the DUMPSCAN environment by entering the dump viewing facility DUMPSCAN command with the dump file name. You can now enter TRACE FORMAT, causing TRACE to format and display the last trace entries merged from all processors. The last trace entry (the one displayed at the bottom of the screen) is trace code 0200.

```
  •
  •
  •
0200 CPU 0000 SVC INTERRUPTION             TOD A16875BF5640   ADR 00FF1F60
     ABEND CODE     ILC     INT CODE   SVC OLD PSW
     FRE016         0002    0000       000C2000 80366514
```

Figure 11. Last Trace Entry in the CP Trace Table with Abend Code FRE016

This is the trace entry for the CP abend. You can see that the abend code is FRE016. The address at which the abend was issued is in the second word of the SVC OLD PSW. You can obtain the name of the module that issued the abend by entering the FINDMOD subcommand with the address:

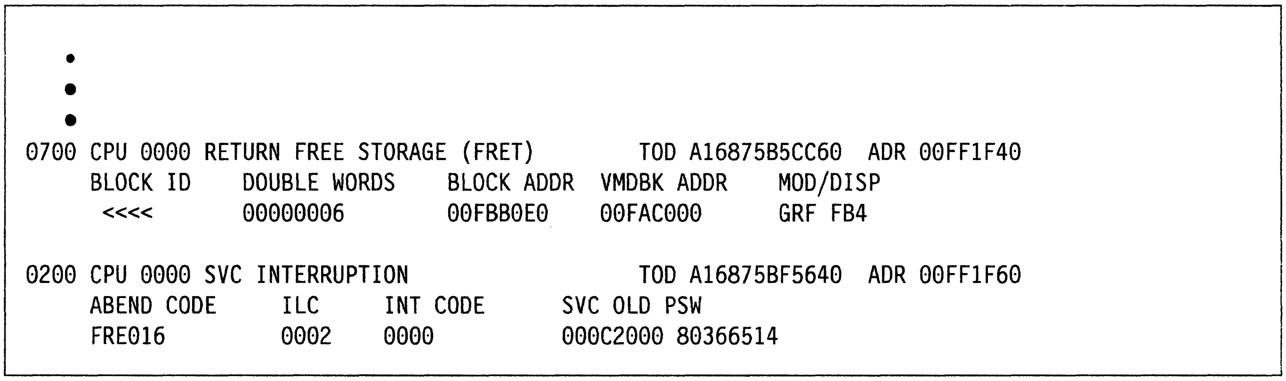
```
findmod 366514
```

```
   00366514 IS CE4 BYTES INTO MODULE HCPFRE   AT 00365830
   00366514 IS 69C BYTES INTO ENTRY  HCPFRET  AT 00365E78
```

Figure 12. Finding the Name of a Calling Module with the FINDMOD Subcommand

The name of the module is HCPFRE, and the abend was issued at the instruction located at hex CE4 bytes from the beginning of the module.

The information in this trace entry corresponds to the primary symptom in the dump: the FRE016 abend. The next-to-last trace entry is of greater interest, since it shows what CP processing step occurred just prior to the abend. To view the trace entries again, use XEDIT scrolling capabilities to scroll to previous subcommand output, or reenter the TRACE subcommand. Now you can examine the next-to-last trace entry. It is trace code 0700, RETURN FREE STORAGE (FRET):



```
    •
    •
    •
0700 CPU 0000 RETURN FREE STORAGE (FRET)        TOD A16875B5CC60  ADR 00FF1F40
     BLOCK ID    DOUBLE WORDS    BLOCK ADDR  VMDBK ADDR    MOD/DISP
      <<<<       00000006        00FBB0E0    00FAC000      GRF FB4

0200 CPU 0000 SVC INTERRUPTION                  TOD A16875BF5640  ADR 00FF1F60
     ABEND CODE     ILC     INT CODE      SVC OLD PSW
     FRE016         0002    0000          000C2000 80366514
```

Figure 13. Second-to-Last Trace Entry from the CP Trace Table

You can see in the trace entry that 6 doublewords at address 00FBB0E0 were returned to free storage by the module HCPGRF at displacement FB4.

Immediately prior to the FRE016 ABEND, HCPGRF returned 6 doublewords of storage at location 00FBB0E0.

Step 3: Analyzing Free-Storage Trace Fields

Module HCPFRE builds trace fields to aid in debugging problems associated with FREE/FRET storage requests. These trace fields are a part of the storage allocated by HCPFRE to satisfy a storage request.

HCPFRE adds 1 doubleword to the front of each request (free-storage header) and 2 doublewords at the end (free-storage trailer) as shown in Figure 14.

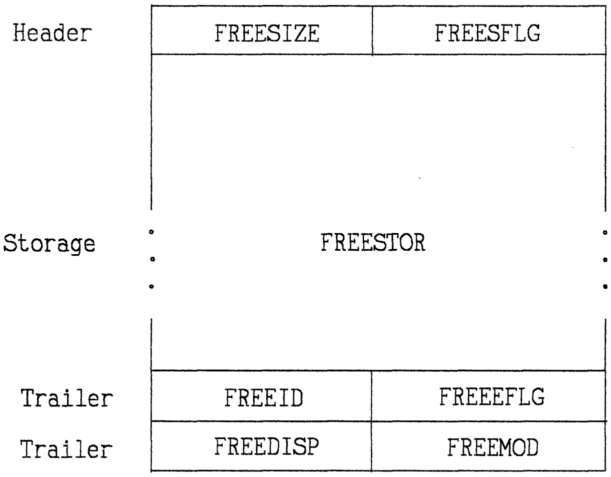


Figure 14. Free-Storage Header and Trailers Added to Free Storage

The contents of these doublewords are as follows:

FREESIZE The size, in doublewords, of the original free-storage request. FREESIZE is located at X'0' in the header.

FREESFLG Four greater-than signs (>>>>) indicating the beginning of the free-storage area. FREESFLG is located at X'4' in the header.

FREESTOR The free-storage area, which is variable in size, depending on the value in FREESIZE.

FREEID The control block identifier. EBCDIC representation of the name of the control block occupying the free storage. Note that four less-than signs (<<<<) means a variable length block without a specific identifier. FREEID is located at X'0' in the trailer.

FREEEFLG Four less-than signs (<<<<) indicating the end of the free-storage area. FREEEFLG is located at X'4' in the trailer.

FREEDISP The offset, or displacement, of the free-storage request into the calling module. FREEDISP is located at X'08' in the trailer.

FREEMOD The last three characters of the module ID requesting free storage. FREEMOD is located at X'C' in the trailer.

Use the DISPLAY subcommand to examine the storage HCPGRF was trying to return to free storage (FRET):

```
display 00FBB0E0
```

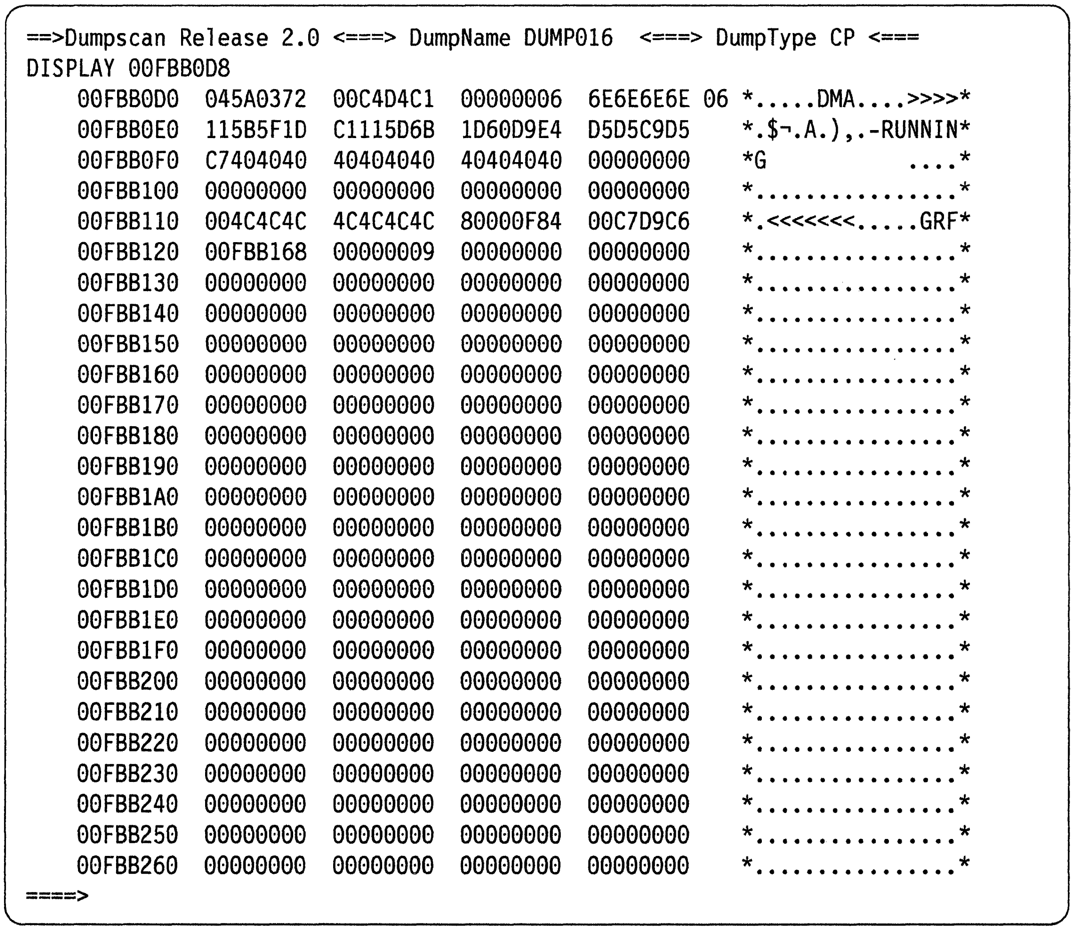
This subcommand displays a full screen of storage beginning at address 00FBB0E0. After looking at module HCPGRF using a source listing or microfiche, and examining the hexadecimal data beginning at 00FBB0E0, we determined that HCPGRF was trying to return to free storage (FRET) a 6-doubleword block.

In-depth examination of the header and trailer doublewords enables you to determine which free-storage trace fields were the source of the FRE016 abend.

Since the free-storage header is 1 doubleword long, in order to determine the header's address you must subtract 8 bytes from the address contained in the trace table entry. Use the subcommand DISPLAY 00FBB0D8 (00FBB0E0 minus X'8') to look at the storage header:

```
display 00fbb0d8
```

This subcommand displays a full screen of storage beginning at address FBB0D0:



```
==>Dumpscan Release 2.0 <===> DumpName DUMP016  <===> DumpType CP <===
DISPLAY 00FBB0D8
    00FBB0D0  045A0372  00C4D4C1  00000006  6E6E6E6E 06 *.....DMA....>>>>*
    00FBB0E0  115B5F1D  C1115D6B  1D60D9E4  D5D5C9D5    *.$¬.A.),.-RUNNIN*
    00FBB0F0  C7404040  40404040  40404040  00000000    *G           ....*
    00FBB100  00000000  00000000  00000000  00000000    *................*
    00FBB110  004C4C4C  4C4C4C4C  80000F84  00C7D9C6    *.<<<<<<<.....GRF*
    00FBB120  00FBB168  00000009  00000000  00000000    *................*
    00FBB130  00000000  00000000  00000000  00000000    *................*
    00FBB140  00000000  00000000  00000000  00000000    *................*
    00FBB150  00000000  00000000  00000000  00000000    *................*
    00FBB160  00000000  00000000  00000000  00000000    *................*
    00FBB170  00000000  00000000  00000000  00000000    *................*
    00FBB180  00000000  00000000  00000000  00000000    *................*
    00FBB190  00000000  00000000  00000000  00000000    *................*
    00FBB1A0  00000000  00000000  00000000  00000000    *................*
    00FBB1B0  00000000  00000000  00000000  00000000    *................*
    00FBB1C0  00000000  00000000  00000000  00000000    *................*
    00FBB1D0  00000000  00000000  00000000  00000000    *................*
    00FBB1E0  00000000  00000000  00000000  00000000    *................*
    00FBB1F0  00000000  00000000  00000000  00000000    *................*
    00FBB200  00000000  00000000  00000000  00000000    *................*
    00FBB210  00000000  00000000  00000000  00000000    *................*
    00FBB220  00000000  00000000  00000000  00000000    *................*
    00FBB230  00000000  00000000  00000000  00000000    *................*
    00FBB240  00000000  00000000  00000000  00000000    *................*
    00FBB250  00000000  00000000  00000000  00000000    *................*
    00FBB260  00000000  00000000  00000000  00000000    *................*
====>
```

Figure 15. Hexadecimal Data Displayed as a Result of the DISPLAY Subcommand

The header and trailer data included in the displayed information are listed below. At hexadecimal address FBB0D8, the free-storage header contains the following values:

```
00000006 6E6E6E6E
```

At hexadecimal address FBB110, the free-storage trailer contains the following values:

```
004C4C4C 4C4C4C4C 80000F84 00C7D9C6
```

Use the data in the header and trailer fields to perform the following analysis:

1. Examine the contents of FREESIZE (the size, in doublewords, of the requested block of storage) and compare it with the size of the free storage returned (FRET) in the CP FRET trace table entry. The sizes should be identical.

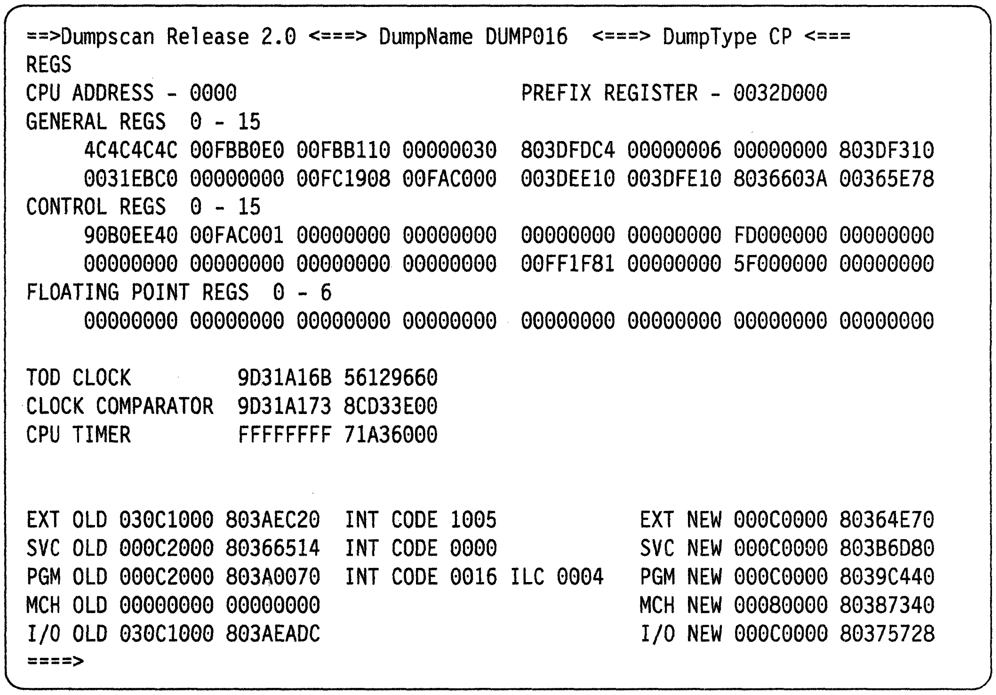
2. Examine the contents of FREESFLG and verify that it contains four greater-than signs (>>>>).

3. Locate the trailer by converting the FREESIZE from doublewords to bytes. In this case, X'6' doublewords = X'30' bytes. Add this to the beginning address of the block (not to the address of the free-storage header):

   ```
   X'00FBB0E0' + X'30' = X'00FBB110'
   ```

4. Examine the FREEID and verify that it contains the valid control block identifier. This can be done by issuing the REGS subcommand and comparing the contents of Register 0 with FREEID.

   An example of the output you receive if you enter the REGS subcommand follows:



```
==>Dumpscan Release 2.0 <===> DumpName DUMP016  <===> DumpType CP <===
REGS
CPU ADDRESS - 0000                          PREFIX REGISTER - 0032D000
GENERAL REGS  0 - 15
     4C4C4C4C 00FBB0E0 00FBB110 00000030  803DFDC4 00000006 00000000 803DF310
     0031EBC0 00000000 00FC1908 00FAC000  003DEE10 003DFE10 8036603A 00365E78
CONTROL REGS  0 - 15
     90B0EE40 00FAC001 00000000 00000000  00000000 00000000 FD000000 00000000
     00000000 00000000 00000000 00000000  00FF1F81 00000000 5F000000 00000000
FLOATING POINT REGS  0 - 6
     00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000

TOD CLOCK         9D31A16B 56129660
CLOCK COMPARATOR  9D31A173 8CD33E00
CPU TIMER         FFFFFFFF 71A36000


EXT OLD 030C1000 803AEC20  INT CODE 1005          EXT NEW 000C0000 80364E70
SVC OLD 000C2000 80366514  INT CODE 0000          SVC NEW 000C0000 803B6D80
PGM OLD 000C2000 803A0070  INT CODE 0016 ILC 0004   PGM NEW 000C0000 8039C440
MCH OLD 00000000 00000000                         MCH NEW 00080000 80387340
I/O OLD 030C1000 803AEADC                         I/O NEW 000C0000 80375728
====>
```

Figure 16. Checking the Control Block Identifier with the REGS Subcommand

5. Examine the FREEEFLG and verify that it contains four less-than signs (<<<<).

In the header, the contents of FREESIZE is 00000006, which matches the length of the free storage returned (FRET) in the trace table entry, and FREESFLG contains (>>>>).

In the trailer, FREEEFLG contains (<<<<).

The contents of FREEID in the trailer is 004C4C4C. It does not match the contents of register 0 (4C4C4C4C). This is the cause of the FRE016 abend.

Step 4: Summarizing the Dump Analysis

You have now determined the immediate cause of the FRE016 abend. Your next step is to search for duplicate problems at your installation to make sure the problem has not already been resolved. If no duplicates are found, contact the IBM Support Center to report the problem and its symptoms. You can summarize the symptoms in this dump as follows:

1. Immediately prior to the FRE016 abend, HCPGRF + X'FB4' was trying to FRET a variable length control block.
2. HCPFRE checked the free-storage trace fields and determined that the control block identifier in the free-storage trailer did not match the one from the control block identifier table (R0).
3. The free storage header and free-storage trailer doublewords were all valid except for FREEID.

Chapter 3. Command Reference

This chapter contains reference information for the dump viewing facility commands. The description of each command includes format, operands, return codes, options, and responses, if any. Where applicable, usage notes further describe the characteristics of the command. For information on messages, see Chapter 5 of this manual. You enter dump viewing facility commands from a terminal attached to a CMS virtual machine.

Notational Conventions

The notations used to define the command syntax in this publication are as follows:

- Commands and subcommands are shown in uppercase and lowercase; the uppercase letters represent the absolute minimum truncation or abbreviation of the command or keyword operand that the dump viewing facility accepts.
- An operand that contains all lowercase characters indicates a variable value you supply. For example, for the operand "fn" you replace "fn" with a filename. For the operand "raddr" you replace "raddr" with the desired real address.
- Where multiple operands are shown between braces { }, you must select one operand.
- Where operands are shown between brackets [ ], you *may* choose one unless otherwise specified.
- An operand that is both enclosed in brackets and underscored indicates that the system will default to the bracketed underscored operand unless you specify another one.
- In all syntax descriptions, fm and filemode are interchangeable.

Command Reference

The dump viewing facility commands are described here in alphabetical order. For further information on messages issued by the various commands, refer to Chapter 5 of this manual.

ADDMAP Command

Use the ADDMAP command to resolve the address of each pageable CP module in real storage at the time of the dump, and to append the resolved module map to the dump file. For virtual machine dumps, use the ADDMAP command to append virtual machine module maps.

The format of the ADDMAP command is as follows:

```
+-------------+---------------------------------------------------+
|             |          [                       ]                |
|             |          [              [ fm ]   ]                |
| ADDMAP      |  fileid  [ xxxxxxxx     [ *  ]   ]                |
|             |          [              [ -  ]   ]                |
|             |          [                       ]                |
+-------------+---------------------------------------------------+
```

Where:

fileid
is the filename, filetype, and filemode of the input CMS file containing the dump viewing facility module map.

xxxxxxxx
is the name of the CMS file containing the dump to be processed, where *xxxxxxxx* is a 1- to 8-character string containing any combination of the characters 0-9, A-Z, @, #, -, _, +, :, and $.

fm
*****
is the filemode of the CMS file containing the dump to which the dump viewing facility module map will be appended.

Usage Notes:

1. Due to the limited amount of modules in a soft abend dump, the ADDMAP command is not supported for soft abend dumps.
2. The ADDMAP command will resolve pageable addresses for CP hard abend and standalone dumps only.
3. In case you add an incorrect map to a dump, the ADDMAP command can be invoked again to add the correct map to the dump.
4. In order to use the ADDMAP command, the load map must first have been processed by the MAP command. If this has not been done, refer to the MAP command description in this chapter.
5. The dump file *(xxxxxxxx)* must have a filetype of DUMP; if it does not, you may rename the file appropriately using the CMS RENAME command. If the dump filename is not specified, you are prompted to enter one.

   If the filemode is not specified, or an asterisk (*) is specified, the system uses the standard CMS search sequence.

Examples:

You want to resolve and append to dump ABEND003 DUMP A1 a processed module map called CPLOAD MAPA A1. To do this, enter the following:

```
addmap cpload mapa a1 ABEND003 a1
```

The result is the ABEND003 dump with the CPLOAD module map appended to it.

To resolve and append to dump DUMPUSER DUMP A1 a processed module map with the default name for a CMS dump, enter:

```
addmap cmsdvf map a1 DUMPUSER a1
```

For more information on the default map names, refer to the MAP command.

Return Codes:

RC	**Explanation**
0	Successful completion
20	Invalid fileid
24	Command line error
28	CMS file does not exist
32	Invalid data in file
36	Disk not accessed
41	Insufficient storage
100	FSREAD/PRINTL error
104	Internal processing error.

DUMPSCAN Command

Use the DUMPSCAN command to initiate a session to interactively analyze and debug problems in a dump. Once you start a session, use the various DUMPSCAN subcommands as described in Chapter 4. The DUMPSCAN command also dynamically creates and appends a CP module map to a CP dump.

The format of the DUMPSCAN command is as follows:

DUMPSCAN	[*xxxxxxxx* [*fm* *_*_]]

Where:

xxxxxxxx

is the name of the CMS dump file to be processed, where xxxxxxxx is a 1- to 8-character string containing any combination of the characters 0-9, A-Z, @, #, -, _, +, :, and $.

fm
*****

is the positional operand specifying the filemode of the desired CMS file containing the dump.

Usage Notes:

1. The dump file must have a filetype of DUMP; if it does not, you must rename the file appropriately using the CMS RENAME command.

   If the filemode is not specified or an asterisk (*) is specified, the system uses the standard CMS search sequence.

2. If the DUMPSCAN command is entered with no operands, you are prompted to enter the desired dump filename (and optionally the filemode).

3. In order for DUMPSCAN to build and append a CP module map, the dump to which you want the module map appended must be on a disk to which you have write access. Furthermore, the dump must not have a map already appended to it.

4. The DUMPSCAN subcommands FRAMEtbl (FRMtbl), RIOblok, VIOblok, FINDUSER, and FINDMod are not supported for soft abend dumps.

Return Codes:

RC	Explanation
0	Successful completion
20	Invalid fileid
24	Command line error
28	CMS file does not exist
32	Invalid data in file
36	Disk not accessed
41	Insufficient storage
100	FSREAD/PRINTL error
104	Internal processing error.

MAP Command

Use the MAP command to convert a CP nucleus load map or a virtual machine load map into a format that the dump viewing facility can process. The converted load map, which is called the dump viewing facility module map, will serve as input to the ADDMAP command. ADDMAP appends the converted load map to the dump file, which is created by the VM/XA System Product DUMPLOAD command.

The MAP command can be invoked in three ways:

1. You can specify the files to be processed.
2. You can specify that the default files are to be processed.
3. You can specify that you be prompted for the files to be processed.

The format of the MAP command is as follows:

```
MAP     [ infile1 [ infile2 ] outfile   [ ( [ CP      ] ] ]
        [                               [   [ CMS     ] ] ]
        [                               [   [ GCS     ] ] ]
        [ NOPrompt                      [   [ PVM     ] ] ]
        [                               [   [ RSCSnet ] ] ]
        [                               [   [ RSCSV2  ] ] ]
        [                                                 ]
        [ PROmpt                                          ]
```

Where:

infile1
is the filename, filetype, and filemode of the input CMS file containing the primary load map.

infile2
is the filename, filetype, and filemode of the input CMS file containing the secondary VM/Pass-Through load map required for PVM dumps. Once the primary CMS nucleus load map (*infile1*) has been processed, the secondary map is processed and included in the module map.

outfile
is the filename, filetype, and filemode of the resulting CMS file containing the dump viewing facility module map, which is used as input to the ADDMAP command.

CP
specifies that a CP load map is to be processed. This is the default.

CMS
specifies that a CMS load map is to be processed.

GCS
specifies that a GCS load map is to be processed.

PVM
specifies that VM/Pass-Through load map and CMS load map are to be processed.

RSCSnet
specifies that an RSCSNET load map is to be processed.

RSCSV2
specifies that an RCSCV2 load map is to be processed.

NOPrompt
specifies that prompting will not occur under any circumstances. The default input and output fileids will be used as indicated by the type specified on the command line. See usage note 3.

PROmpt
is the default. It specifies that the user wants to be prompted for the following:

1. The type of maps to process. The acceptable reply is:

   CP
   specifies that a CP load map is to be processed.

   CMS
   specifies that a CMS load map is to be processed.

   GCS
   specifies that a GCS load map is to be processed.

   PVM
   specifies that VM/Pass-Through and CMS load maps are to be processed.

   RSCSnet
   specifies that an RSCSNET load map is to be processed.

   RSCSV2
   specifies that an RSCSV2 load map is to be processed.

   null line
   specifies that a CP load map is to be processed.

   SUBSET
   enter CMS subset mode.

   HX
   terminate the MAP command.

2. The fileids of the input load maps and the output module map. The acceptable reply is:

   filename [filetype [filemode]]
   filetype and filemode of the input files default to MAP and *, respectively if not specified. The filetype and filemode of the output file default to MAP and A if not specified.

   null line
   use the predefined fileid. See usage note 3.

   SUBSET
   enter CMS subset mode.

   HX
   terminate the MAP command.

Usage Notes:

1. If file ids are used on the MAP command line, VM/Pass-Through requires that *infile2* be specified, or an error message is issued and command processing terminates. CMS, CP, RSCSNET, GCS, and RSCSV2 have no required secondary maps. Therefore, if *infile2* is specified, an error message is issued and command processing terminates.

2. Prompting messages will not be issued when the user specifies fileids or NOPROMPT on the command line. If an error occurs or a required fileid is not specified, an error message is issued and command processing terminates.

3. The following are the default map names used when fileids are not specified:

Type	Input Map Names	Output Map Names
CP	CPLOAD MAP * (nucleus)	HCPXADVF MAP A
CMS	CMSNUC MAP * (nucleus)	CMSDVF MAP A
GCS	GCSNUC MAP * (nucleus)	GCSDVF MAP A
PVM	CMSNUC MAP * (primary)	PVMDVF MAP A
	PVM MAP * (secondary)	
RSCSNET	RSCSNET MAP *	RSCSDVF MAP A
RSCSV2	RSCSV2 MAP *	RSV2DVF MAP A

Return Codes:

RC	Explanation
0	Successful completion
20	Invalid fileid
24	Command line error
28	Missing files or file already exists
32	Invalid data in file
36	Disk not accessed
41	Insufficient storage
100	FSREAD/PRINTL failure
104	Internal processing error.

PRTDUMP Command

Use the PRTDUMP command to print summary information about the major system control blocks and data areas of the dumped system. The PRTDUMP command also dynamically creates and appends a CP module map to a CP dump.

Use the summary reports to assist in problem analysis. Each summary report is designed as a reference to information in the dump. You refer to it when interactively viewing a dump using the DUMPSCAN command, or when examining a hexadecimal dump printed by the VM/XA SP DUMPLOAD command.

Note: For more information on obtaining a printed dump in hexadecimal and EBCDIC, see the *VM/XA SP CMS Command Reference.*

The printed summary information for control blocks includes key fields rather than the entire block. The data and flags are interpreted and a text description provided wherever possible. Only control block data that is used most often for debugging is presented.

Using the dump viewing facility PRTDUMP command, you can print these summary reports:

- Symptom Records

  The dump symptom record contains information that indicates the state of the system when the dump was taken. The printed information is in the same format as output obtained through the SYMPTOM subcommand of DUMPSCAN.

- General Processor Information

  This report contains information that describes the processor(s) associated with the dump. This report includes:

  - Registers
  - Clocks and timer
  - CPU address and prefix register
  - Program status words (PSWs).

  For System/370 virtual machine dumps, the general processor summary report will also include:

  - Channel status word (CSW)
  - Channel address word (CAW)
  - Interval timer
  - Current PSW.

- Dump Viewing Facility Module Map

  Module and entry point names and their addresses in the dump are printed for all modules that were in real storage when the dump was taken.

- Frame Table

  The frame table summary report describes the use and status of all pages of real storage at the time the dump was taken.

  The frame table is a series of contiguous 4K-byte pages containing one 16-byte entry for every 4K-byte page of real storage. The frame table summary report describes the use and status of all pages of real storage at the time the dump was taken. Each printed summary report contains:

  - Real storage page number
  - Address of the frame table entry
  - Contents of the frame table entry
  - Frame use.

- DUMPID

  - Only valid for VM dumps.

- Real I/O Control Blocks

  This report summarizes all real devices in the system. The report includes device and subchannel numbers, device class and type, and key status indicators.

- Trace Table

  This report contains trace table entries in the order of creation from the oldest to the most recent. If the dump is of a multiple processor system, the trace tables of the different processors are merged according to the time-of-day (TOD) clock values.

- Virtual Machine List

  A summary of the information in the virtual machine definition block (VMDBK) is printed for each virtual machine that was logged on (that is, on the global cyclic list) to the system at the time of the dump. The printed summary includes the virtual machine's mode (370, XA, V=R, V=F), the virtual machine's userid, and various status indicators for each virtual machine.

- Virtual Machine User Summary

  This summary report is printed for each virtual machine requested through the PRTDUMP command. This summary includes virtual device data, active I/O indicators, and additional information from the VMDBK.

- Virtual Machine Dump Specific Summary Reports.

  See Appendix E for more information.

The format of the PRTDUMP command is as follows:

```
PRTDUMP  [xxxxxxxx [fm]]  [( [FRMtb   ] [USER userid...]]
                   [* ]      [        ]
                             [FRAMEtbl]
                             [MAP     ]
                             [RIOblok ]
                             [Trace   ]
                             [Vmdbk   ]
                             [DUMPID  ]
                             [ALL     ]
```

Where:

xxxxxxxx
is the positional operand specifying the filename of the CMS file containing the dump to be processed, where *xxxxxxxx* is a 1- to 8-character string containing any combination of the characters 0-9, A-Z, @, #, -, _, +, :, and $.

fm
*
is the positional operand specifying the filemode of the CMS dump file. If the filemode is not specified, or an asterisk (*) is specified, the file is searched for using the standard CMS search sequence.

(
indicates that PRTDUMP operands will follow. If the (is specified with no operands following, ALL is the default. If (is not specified and operands follow the filemode, they are treated as invalid operands. If nothing follows the filemode, ALL is the default.

FRAMEtbl/FRMtb
specifies that the frame table summary should be printed.

MAP
specifies that the dump viewing facility module map that is appended to the dump should be printed. For CP dumps only, if a load map is not currently appended to the dump, one is built dynamically and appended. However, the dump file must be on a CMS disk to which you have write access.

RIOblok
specifies that the real I/O control blocks summary report should be printed.

Trace
specifies that the trace table summary report should be printed.

Vmdbk
specifies that the VMDBK summary report should be printed.

DUMPID
specifies that the dumpid of the virtual dump should be printed. This option only applies to virtual machine dumps.

ALL
specifies that all applicable reports should be printed for the dump type. For CP dumps, these include the frame table, module map, real I/O control blocks, trace table, and VMDBK summary reports. This is equivalent to specifying the operands FRMTB, MAP, RIOBLOK, TRACE, and VMDBK on the command line. If no summary reports, including a USER summary report, are requested by their keyword operands, ALL is the default.

For virtual machine dumps, ALL specifies that symptom record data, general processor information, DUMPID, and the module map be printed. ALL is equivalent to specifying MAP and DUMPID on the command line. If no operands are specified, ALL is the default.

USER userid
specifies that the virtual machine summary report should be printed for the virtual machine (userid) specified. Each userid must be preceded by the USER keyword.

Usage Notes:

1. The FRAMEtbl (FRMtb), RIOblok, Trace, USER, and Vmdbk operands of the PRTDUMP command are not supported for virtual machine dumps. The FRAMEtbl (FRMtb), RIOblok, and MAP operands of the PRTDUMP command are not supported for soft abend dumps.

2. The dynamic map build capabilities for the PRTDUMP command apply to CP hard abend and standalone dumps. Dynamic map build capabilities are not supported for soft abend dumps.

3. In order for PRTDUMP to build and append a CP module map, the dump to which you want the module map appended must be on a disk to which you have write access.

   Furthermore, the dump must not have a map already appended to it.

4. If you enter the PRTDUMP command without any operands, you will be prompted to enter the dump filename and the filemode, or HX to end processing. After you respond to the prompt, the summary reports will be printed.

   This is equivalent to entering the ALL operand, which is the default.

5. If the filemode is not specified, or an asterisk (*) is specified, the system uses the standard CMS search sequence.

6. The symptom record summary and general processor information summary reports are printed for every valid invocation of the PRTDUMP command. They are the first reports to be printed.

7. If you want USER summary reports for more than one virtual machine, the USER operand may be specified more than once. The VIO portion of the USER summary report is supported.

   You can use the USER operands with other keyword operands. For example, the following is valid:

   ```
   prtdump 12345 (user userid1 frmtb user userid2 map
   ```

This will result in the following summary reports being printed:

- The symptom record summary
- The general processor information summary
- The frame table summary
- The dump viewing facility module map
- The user summaries for userid1 and userid2.

If a user summary report is desired in addition to all other summary reports, you must specify both the USER and ALL operands. For example, if you enter:

```
prtdump dumpuser (user userlawc all
```

You will get:

- The symptom record summary
- The general processor information summary
- The frame table summary
- The dump viewing facility module map
- Real I/O control blocks
- The trace table
- VMDBK summary reports.

in addition to a summary report for user USERLAWC.

8. FRAMETBL is an equivalent operand to the FRMTB. The FRAMETBL form of this operand may be truncated to a minimum of FRAME.
9. The heading "FRAME TABLE SUMMARY FOR DUMP xxxxxxxx" appears at the top of each page of the summary report.
10. If the dump symptom record is missing or not readable, an error message is issued and you will be prompted to continue dump processing.

Return Codes:

RC	**Explanation**
0	Successful completion
20	Invalid fileid
24	Command line error
28	CMS file does not exist
32	Invalid data in file
36	Disk not accessed
41	Insufficient storage
100	FSREAD/PRINTL error
104	Internal processing error.

TRACERED Command

Use this command to process trace data recorded by the TRSOURCE command in system trace files as well as CP system trace tables records on tape or system trace files. TRSOURCE command data can be merged with CP data or with other TRSOURCE command trace data to produce CP data or with TRSOURCE command trace data to produce a consolidated output file in chronological sequence.

The format of the command is as follows:

```
            ┌                    ┐
            │ id1              ) │ [id2...id5]
 TRACERED   │ NAME fn1[,id1]  (  │ [{NAME fn2[,id2]}..{NAME fn5[,id5]}]
            │ DEV vdev         ( │
            │ TAPn             ) │
            └                    ┘

             [ CMS fname [ ftype [ fmode ] ] ]
             [ GTRACE                        ]
             [ PRINT      [ A ]              ]

             OPTIONS:

             [ HEX    ] [ ALL    ] [ LRECL { 80  } ]
             [ FORMAT ] [ SELECT ] [       { 132 } ]
```

Where:

id1

specifies the primary input to TRACERED from an input system trace file containing CP or TRSOURCE data. The spool ID, *id1*, is a decimal number from 1 to 9 999.

NAME fn1 [,id1]

specifies the primary input to TRACERED from the filename containing CP or TRSOURCE data. The TRSAVE command creates and names a file called fn1. The parameter, id1, identifies an input system trace file created for fn1. The parameter, id1, is specified as a spoolid, a decimal number from 1 to 9 999.

DEV vdev

specifies the primary input to TRACERED from the virtual device address of a tape containing CP trace table data. See the *VM/XA SP CMS Command Reference* for a list of supported virtual tape devices.

TAP*n*

specifies the primary input to TRACERED from the symbolic tape identification (TAPn) or the virtual device address of a tape containing CP trace table data. See the *VM/XA SP CMS Command Reference* for a list of supported TAPn values.

[id2..id5]
NAME fn2[,id2]..NAME fn5 [,id5]

identifies up to four additional system trace files or filenames containing CP or TRSOURCE trace data that are to be merged with data from the primary file. The output is merged by time-of-day and formatted as it is written to a CMS file or the virtual printer as designated by this command.

id2..id5 identifies a spoolid or spoolids which must be decimal numbers from 1 to 9 999.

NAME fn2[,id2] ...NAME fn5 [,id5]
identifies the filenames created by the TRSAVE command. The identification ,id2..,id5 is for input system trace files created for fn2..fn5 and is specified as a spool ID, a decimal number from 1 to 9 999.

CMS fname [ftype [fmode]]
indicates that the output of the TRACERED command should be directed to a CMS file of the filename, filetype, and filemode you specify. If you do not specify a filetype, the default is CPTRACE. If you do not specify a filemode, the default is A.

GTRACE
specifies that the output will be in the form of an OS QSAM file. Only GTRACE records recorded by GCS are output to this file.

PRINT
designates the output of the TRACERED command to go to your virtual printer. This is the system default.

HEX
stipulates that each trace entry selected will appear in hexadecimal format. CP trace table data output is one line per trace entry. The number of lines used for TRSOURCE data is variable depending on the entry's record size. You may need to use this option if you want to direct the output to a CMS file and limited DASD space precludes using the FORMAT option.

FORMAT
specifies that each selected trace entry will appear in a format applying to either CP trace table or TRSOURCE recorded data. This is the system default. Three lines per trace entry are formatted for CP trace table entries. The number of lines of output for TRSOURCE recorded entries is determined by a formatting routine and the length of the trace record which the originator supplies. When the formatting routine is not supplied, the data is in formatted hex. This is the system default.

ALL
indicates that all trace entries from the specified input tape or system trace file are to be selected for output. You are not prompted for selection criteria.

SELECT
signifies that you want selection criteria to be applied to the trace entries. You are prompted for the selection criteria you want. This is the system default.

LRECL $\left\{\begin{matrix}80\\132\end{matrix}\right\}$
specifies the record length maximum for the output file. When not specified, CMS files default to 80 bytes and print files default to 132 bytes.

Selection Criteria for the SELECT Option

If you choose, or allow the system to default to, the SELECT option for the TRACERED command, you receive the following prompt:

```
ENTER TRACE ENTRY SELECTION CRITERIA, NULL LINE TO END
          SELECTION, OR QUIT TO END TRACERED COMMAND
```

If you enter a null line, *no* selection criteria are applied. As a result, everything is traced.

TIME is the only applicable selection criterion for TRSOURCE data.

If you enter selection criteria, you will continue to receive this prompt until you enter a null line, thus indicating that entry of selection criteria is complete. Acceptable replies to the prompt 010A are:

1. Any combination of the following keywords and values:

   CODE xxxx xxxx . . .
   You may enter up to sixteen 4-digit hexadecimal values representing the trace table codes to be selected during data reduction of the input tape or system trace file. Leading zeros are required. The last two digits of any code may be specified as the character string ** or *. This indicates that you want a range of trace table codes. For instance, 15** or 15* would select all trace table entries beginning with the hexadecimal digits 15. CODE is valid for CP trace table data only.

   VMDBK xxxxxxxx xxxxxxxx . . .
   You may enter up to 16 hexadecimal values that represent the VMDBK addresses to be used during data reduction of the input tape or system trace file. Leading zeros are not required. Each address specified must be in the range of hex 0 to 7FFFF000 and on a page (4K-byte) boundary. VMDBK is valid only for CP trace table data.

   RDEV xxxx xxxx . . .
   You may specify up to 16 hexadecimal values that represent the RDEV numbers to be used during data reduction of the input tape or system trace file. Leading zeros are not required. Each device number specified must be in the range of hex 0 to FFFF. RDEV is valid only for CP trace table data.

   VDEV xxxx xxxx . . .
   You may specify up to 16 hexadecimal values that represent the VDEV numbers to be used during data reduction of the input tape or system trace file. Leading zeros are not required. Each device number specified must be in the range of hex 0 to FFFF. VDEV is valid only for CP trace table data.

   CPU xxxx xxxx . . .
   You may specify up to 16 hexadecimal values that represent the processor addresses to be used during data reduction of the input tape or system trace file. Leading zeros are not required. Each processor address specified must be in the range of hex 0 to 3F. CPU is valid only for CP trace table data.

   TIME mm/dd/yy hh:mm:ss mm/dd/yy hh:mm:ss
   You may designate a start/stop date/time for the data reduction of the input trace entries. If this option is selected, only trace entries created within the designated date and time range will be processed. You may select only one date/time range. All the TIME parameters may be input on the same line. TIME is the only valid selection criteria if TRSOURCE entries are being processed.

2. CMS – To enter the CMS subset mode.

3. HELP – To invoke the help panel for the TRACERED command.

4. QUIT – To terminate the TRACERED command.

5. A null line – To indicate that you are done entering selection criteria and want to end prompting.

   If your immediate response to the prompt is a null line, then all trace entries are selected.

Usage Notes:

1. The *filename* designates a set of system trace files created by one invocation of the TRSAVE command. Multiple system trace files may exist with the same filename. If you specify a filename as NAME *fn*, TRACERED processes all system trace files with that filename, beginning with the oldest. If you specify NAME *fn1* ,*id1* , TRACERED processes all system trace files with that filename beginning with the spoolid *id1*. If you specify *id*, TRACERED processes that system trace file only.

2. TRACERED uses the first 2 bytes of each TRSOURCE created entry record as the length of that entry.

3. When more than one tape created by the same TRSAVE command is to be mounted and processed, they must be processed in chronological order.

4. Use the QUERY TRFILES command to display information about system trace files you own, including spool ID, filename, and the time of creation.

5. A total of five system trace files can be merged. However, only one CP trace table file or tape may be included. Therefore, you may specify:

   - One CP system trace table file with up to four TRSOURCE trace files

   - One CP tape with up to four TRSOURCE trace files, or

   - Up to five TRSOURCE trace files.

   You cannot merge a CP tape or tapes with a CP system trace file.

6. The following guidelines apply to your responses to the prompt for selection criteria when CP trace table data is processed:

   - Each time the prompt appears, you may specify any or all of the six selection types: CODE, VMDBK, RDEV, VDEV, CPU, and TIME.

   - You do not have to input these values on one line (with the exception of TIME) because the prompt for selection criteria reappears until you enter a null line or QUIT.

   - You cannot specify more than a total of 16 values per selection type in all of your response lines combined, with the exception of TIME.

   - An individual trace entry may or may not contain a field of the same type specified in your reply to the prompt for selection criteria. All trace entries have a trace code, processor address, and time-of-day field; not all trace entries have a VMDBK address, real device number, or virtual device number. As a result, there are two main rules that apply to TRACERED data reduction:

     – If you specify just one selection type in response to the prompt (CODE, VMDBK, RDEV, VDEV, CPU, or TIME) then the trace entries chosen for output will have a field for the selection type you indicated *and* a value in the field that matches the value you indicated in your response.

       For instance, your response to the selection criteria prompt may have been the following:

       ```
       code 0c32 1000 0100
       ```

       In this case, all trace entries with the code 0C32, 1000, or 0100 are selected.

- If you specify more than one selection type in response to the prompt (some combination of CODE, VMDBK, RDEV, VDEV, CPU, and TIME) then the trace entries chosen for output depend on the selection type fields a given trace entry contains. Trace entries that contain fields for only some of the selection types you indicated, but whose values in those fields match the values you designated, are chosen for output. A trace entry does not have to have all the selection types you specified in order to be chosen for output.

  For example, your response to the selection criteria prompt may be the following:

  ```
  code 0c32 rdev 399 vdev 191 code 1000 0100
  ```

  In this case, trace entries to be selected for output depend on the following:

  - Trace entries with code 0C32 contain both an RDEV and VDEV value. Therefore, a trace entry of 0C32 must have an RDEV field value of 399 and a VDEV field value of 191 in order to be selected for output.
  - Trace entries with code 1000 contain only an RDEV value. The VDEV selection value does not apply in this situation. Therefore, all trace entries of code 1000 with an RDEV field value of 399 are chosen for output.
  - Trace entries with code 0100 have neither an RDEV nor VDEV value. Therefore all trace entries of code 0100 are output.

7. The following guidelines apply to your responses to the prompt for selection criteria when CP and/or TRSOURCE data is requested:

   - If you immediately enter a null line when prompted for the selection criteria, this indicates that no data reduction is requested. As a result, all trace entries will be selection.
   - You can specify a time range only once.

8. If you own more than one set of files with different originators, you must specify NAME *fn* , *id* for the file you want.

9. All files being processed must have been created in a VM/XA SP environment. Previous trace files not created under VM/XA SP cannot be processed.

10. The CMS return code explanation for return code 28 is:

    ```
    Tape not attached or output file already exists or system trace
    file not found.
    ```

11. The format option is supported for CP trace table records, CP records of type DATA and IO, and GCS guest trace records. Other guest trace records will be formatted in hex output.

12. When a new system trace file or tape is being processed, a message will be issued to the screen indicating which file or tape is being processed.

13. When you indicate VMDBK in your response, data reduction takes place by VMDBK address. Should you want to trace a specific user or users, you need to find out the VMDBK address for the user in question *before* invoking the TRACERED command.

14. For TIME selection criteria, the year YY=00 defaults to 1900.

15. If the HEX option is chosen, all entries will be formatted in hex.

16. When processing system trace files which contain GCS trace entries, it is advisable to use the LRECL 132 option for formatted CMS output.

Examples:

1. TRACERED 190 CMS TRACE1 TABLES A

   TRACERED is specified for one system trace file by spoolid 190. The output goes into CMS file TRACE1 TABLES A. Selection criteria are prompted.

2. TRACERED 183 194 172

   TRACERED is specified for three system trace files by spoolids 183, 194, and 172. All defaults are taken. That is, the output is formatted and goes to a printer. Selection criteria are prompted.

3. TRACERED 0183 NAME PVMUSER 194 CMS TRACE2 OUTPUT A (SELECT

   TRACERED is specified for two system trace files by spoolids 183 and 194 and one filename, PVMUSER. Output goes to CMS file TRACE2 OUTPUT A and selection criteria is prompted.

4. TRACERED NAME PVMUSER 194 CMS TRACE2 OUTPUT A

   TRACERED is specified for one system trace file by spoolid 194 and file name PVMUSER. Output goes to CMS file TRACE2 OUTPUT A and selection criteria are prompted.

5. TRACERED DEV 182 NAME PVMUSER 194

   TRACERED is specified for one system trace file by spoolid 194 and filename PVMUSER. One CP trace tape on virtual device 182 is input. All defaults are taken. That is, the output is formatted and goes to a printer. Selection criteria are prompted.

6. TRACERED NAME PVMUSER ,100

   TRACERED is specified for one filename, PVMUSER. Processing will start with spoolid 100, which identifies a system trace file within filename PVMUSER. All defaults are taken. That is, the output is formatted and goes to a printer. Selection criteria are prompted.

7. TRACERED NAME PVMUSER ,100 RSCS ,2000

   TRACERED is specified for two filenames, PVMUSER amd RSCS. Processing for PVMUSER will start with spoolid 100 and processing for RSCS will start with spoolid 2000. 100 and 2000 identify system trace files within filenames PVMUSER and RSCS. All defaults are taken. That is, the output is formatted and goes to a printer. Selection criteria are prompted.

8. TRACERED NAME RSCS ,2000 500

   TRACERED is specified for one filename, RSCS, and for one system trace file by spoolid 500. Processing for RSCS will start with spoolid 2000. All defaults are taken. That is, the output is formatted and goes to a printer. Selection criteria are prompted.

9. TRACERED 190 CMS TRACE1 (LRECL 132

   TRACERED is specified for one system trace file by spoolid 190. The output goes into CMS file TRACE1 CPTRACE A and the record length for the output file will be 132 bytes. Selection criteria are prompted.

10. TRACERED 190 (LRECL 80

    TRACERED is specified for one system trace file by spoolid 190. The output is formatted and goes to a printer. The record length for the output file will be 80 bytes. Selection criteria are prompted.

TRACERED Output

There are four types of output you may generate from the TRACERED command: a formatted CMS file, a formatted print file, an unformatted CMS file, or an unformatted print file.

Note: You may encounter storage constraints if you select a CMS file for the output. The more trace entries that meet the selection criteria, the larger the storage requirements.

One way to alleviate storage constraints is to designate more stringent selection criteria.

Formatted CMS File

When you select the FORMAT (the default) option and designate a CMS file in the TRACERED command, the default setting of the output to the CMS file consists of 80-character fixed-length records. You can reset the default record using the LRECL option.

CP Trace Table Output: Each CP trace entry selected comprises three output lines:

- The first line contains the trace code, the CPU address, a description of the type of entry, and the time the entry was created.
- The second and third lines describe the trace entry data contents, which vary according to the type of entry.

```
0600 CPU 0000 OBTAIN FREE STORAGE (FREE)                     TIME 19:26:31
     BLOCK ID    DOUBLE WORDS    BLOCK ADDR    VMDBK ADDR    RETURN ADDR
     <IOR        00000020        00FC5EA8      00FB1000      004CF922
```

Figure 17. Formatted CP Trace Entry

Each time TRACERED encounters a missing CP data record, the following line appears in the CMS file:

```
CPU xxxx - TRACE DATA HAS BEEN LOST
```

A blank line precedes and follows the lost data line in the CMS file. Processing continues with the next available trace entry.

Processing Input Tapes: Each time a new input tape is processed, a header record is written that contains the volume sequence number and the creation date and time of the TRSAVE tape. Also at the time a new input tape is processed, that same information is displayed at your terminal screen. Informational message 030I (see page 163) indicates the volume sequence number of the current TRSAVE input trace tape and the date the tape was created.

Note: This message does not indicate that the dump viewing facility has finished processing. It continues processing the input tape.

Guest Trace Output (GCS): Each GCS trace entry processed by TRACERED is sent to a GCS routine, CSIYTD, to be formatted for output. An example of formatted GCS output follows:

```
3D 0E NET    VM/GCS USER REQUESTED GTRACE
    TIME OF DAY CLOCK = 9A6266195A8E2C00
    LENGTH OF GTF HEADER AND TRACE DATA =0017
    FORMAT ROUTINE ID = FD
    EVENT IDENTIFICATION = EFE1
```

Figure 18. Guest Trace Data of Type GCS

Other Types of Guest Trace Output: TRACERED formats other guest trace data and produces the following information per trace record:

- The time the entry was created.
- The contents of the trace entry in hexadecimal (up to 20 bytes per line).
- The string SPID followed by a decimal number. This identifies the system trace file from which the entry was obtained.

An example of other types of guest trace output formatted by TRACERED follows:

```
     16:23:10    0201AED0 085F7400 000C1000 00025BA2 0001BDE0 SPID 1000
                 02025BA2 0C5F7400
```

Figure 19. Other Types of Guest Trace Output Formatted by TRACERED

I/O Trace Entry Output: TRACERED recognizes I/O data trace information. An example of a formatted I/O data trace entry follows:

```
TRACE TYPE IO, CPU 0000                            TIME 01:00:00
   TRACEID - I01, TRACESET = IOSET, IODATA = 12
   USER - TIMR, I/O OLD PSW = 033E0000 803973C8
   DEVICE = 0570, SCSW = 00404007 005D10B0 0C000001
  ->CCW(1) = 4B000000 60000001, CCW ADDRESS = 005D1088
  ->CCW(2) = 015D1050 A0000001, CCW ADDRESS = 005D1090
   DATA = C2
  ->CCW(3) = 005CE2F5 A0000047, CCW ADDRESS = 005D1098
   DATA = 1101E028 41002842 00F1F57A    *.....15:*
  ->CCW(4) = 005D1058 6000002D, CCW ADDRESS = 005D10A0
   DATA = 29034200 4100C0C1 11E76B29    *.....A.X,.*
  ->CCW(5) = 03000000 20000001, CCW ADDRESS = 005D10A8
```

Figure 20. I/O Data Formatted by TRACERED

Data Trace Entry Output: TRACERED recognizes CP data trace information.

An example of a CP data trace entry output formatted by TRACERED follows:

```
TRACE TYPE DATA, CPU 0000                        TIME 16:06:07
    TRACEID = Q, TRACESET = NULL, ADDRESS = 0061A058
  -> DATALINK STRING   1 = G11.090=VMDBK
    DATA = 00BDC00F 00A9E00F 00A9D00F 00A9C00F  *.....z...z...z..*
           00A9B00F 00A9A00F 00A9900F 00A9800F  *.z...z...z...z..*
           00A9700F 00A9600F 00A9500F 00A9400F  *.z...z-..z&..z .*
           00A9300F 00AB500F 00BC800F 00AC700F  *.z....&.........*
           80000020 80000020 80000020 80000020  *................*
           80000020 80000020 80000020 80000020  *................*
           80000020 80000020 80000020 80000020  *................*
           80000020 80000020 80000020 00AA0000  *................*
           D6D7C5D9 C1E3D6D9 D6D7C5D9 C1E3D6D9  *OPERATOROPERATOR*
  -> DATALINK STRING   2 = G10.100=HCPTXCCM
    DATA = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF  *................*
           030C0000 803A30F2 000C2000 80DEB05A  *.......2.......!*
           000C1000 8047865C 00000000 00000000  *......f*........*
           030C1000 804631F0 00000000 00000000  *.......0........*
           00000000 00000000 00000000 00000000  *................*
           000C0000 803B1060 000C0000 80478060  *.......-.....,.-*
           000C0000 804A0230 00080000 80401170  *.....¢....... ..*
           000C0000 803DCCF8 0033E000 00001004  *.......8........*
           0002001C 00040016 00DE5888 00000000  *...........h....*
           00000000 00000000 00000000 00000000  *................*
           00000000 00000000 00000000 00000000  *................*
           00010001 00327240 00000000 00000000  *....... ........*
           00000000 00000000 00000000 00000000  *................*
           00000000 00000000 00000000 00000000  *................*
           00000000 00000000 00000000 00000000  *................*
           00000000 00000000 00000000 00000000  *................*
```

Figure 21. Data Trace Entry Output Formatted by TRACERED

Formatted Print File

TRACERED formatted print output is identical to formatted CMS file output except that the default record length is 132 bytes. The TRACERED output is directed to your virtual printer rather than to a CMS file.

Unformatted CMS File

When you select the HEX option and designate a CMS file in the TRACERED command, the default setting of the output to the CMS file consists of 80-character fixed-length records. You can reset the default record using the LRECL option.

CP Trace Table Output: Each CP trace entry selected comprises one line. The line contains the processor address, the time the entry was created, the trace code, and the trace entry data contents. These are in hexadecimal format.

```
CPU     TIME     CODE  *********** TRACE ENTRY CONTENTS ***********

   05/31/88

0000  11:51:21  3310  E4C34040 00000001 00FAC000 00FFBA80 005311E0 SPID 0003
```

Figure 22. CP Trace Table Output

Other Trace Types: TRACERED processes other traces (I/O, CP data, guest trace) and produces the following information per trace record:

- The time the entry was created.
- The contents of the trace entry in hexadecimal (up to 20 bytes per line).
- The string SPID followed by a decimal number. This identifies the system trace file from which the entry was obtained.

An example of unformatted trace data output follows:

```
    16:23:10    0201AED0 085F7400 000C1000 00025BA2 0001BDE0 SPID 1000
                02025BA2 0C5F7400
```

Figure 23. Guest Trace Output

Unformatted Print File

TRACERED hexadecimal print output is identical to hexadecimal CMS file output except that the default record length is 132 bytes. The TRACERED output is directed to your virtual printer rather than to a CMS file.

CMS Return Codes

When the trace reduction process is complete, the dump viewing facility returns one of the following codes to CMS:

Return Codes:

RC	Explanation
0	Successful completion
4	No trace entries meet the selection criteria
20	Invalid character in output fileid
24	Command syntax error
28	Tape not attached or output file already exists or system trace file is not found
36	Disk not accessed
41	Insufficient virtual storage
100	I/O error.

Chapter 4. DUMPSCAN Reference

This chapter contains reference information for the DUMPSCAN subcommands used to interactively view data from a dump. It also contains XEDIT services that can only be used within an EXEC.

XEDIT Services

The following table lists all the XEDIT services that can be used only from within a macro. The syntax for these services is documented first in this chapter.

Subcommand	Description
FINDStrg	Search for a particular string of data in the dump while you are still in an EXEC
NOTE	Send text output to the dump viewing facility to be displayed on the terminal, printer or both
READStrg	Read data from the dump, via a direct or indirect address, while in an EXEC
SCAN	Process a PF key assignment or command string to the system product interpreter.

Figure 24. XEDIT Services Only Used within a Macro

DUMPSCAN Subcommands

The following table lists all the subcommands for DUMPSCAN. The minimum truncation for each command is indicated by the uppercase letters in the subcommand column. All subcommands are valid for CP hard abend and stand-alone dumps except DUMPID, VPAIR, VREG, and VSTAT. Reference information on these subcommands follows XEDIT services in this chapter.

Subcommand	Description
(null line)	Enter a null line to continue the previous CHAIN, LOCATE, LOCATEUP, BACKWARD, or FORWARD subcommands
+ (plus symbol) - (minus symbol)	Increase forward through hexadecimal data or trace entries; decrease backward through hexadecimal data or trace entries
& (ampersand) or &name	Assign symbolic names to subcommands
? (question mark)	Display the last subcommand entered
= (equal symbol)	Reexecute the previous DUMPSCAN subcommand
Backward	Scroll backward through hexadecimal data or trace entries
BLock	Format control blocks within a dump
CHain	Display the chain of control block addresses
CMS	Enter the CMS subset mode
CPU	Display the address and prefix register values for each processor
Cregs	Display the control registers for a specific processor
Display	Display dump data in both hexadecimal and EBCDIC
DUMPID	Display the dump identifier assigned on a VMDUMP command
END	Exit from the DUMPSCAN command and return to CMS
FINDMod	Display the displacement and module name or entry point, given an address; or display an address, given a module name or entry point
FINDUSER	Display the active save areas and VMDBK status for a specified userid
Forward	Scroll forward through hexadecimal data or trace entries
FRAMEtbl or FRMtbl	Display the frame table entry for the specified address
Gregs	Display general purpose registers for the specified processor
HELP	Display individual DUMPSCAN help files or the menu that lists all the DUMPSCAN help files
HX	Exit from the DUMPSCAN command and return to CMS
Locate	Locate the next occurrence of a hexadecimal or character string in a dump
LocateUp	Locate the previous occurrence of a hexadecimal or character string in a dump

Figure 25 (Part 1 of 2). Subcommands for DUMPSCAN

Subcommand	Description
Print or PRT	Direct output of the subcommand to the printer
QUIT	Exit from the DUMPSCAN command and return to CMS
Regs	Display registers, clocks, timer, and program status words for a specific processor
RIOblok	Display data about real I/O devices
SELect	Specify trace table display criteria
SNAPlist	Display a summary snap list in a soft abend dump
SYMPtom	Display symptom record data
Trace	Display trace table entries
VIOblok	Display data about virtual I/O devices for a specific virtual machine
Vmdbk	Display data about one or more virtual machines
VPair	Display the contents of a vector register pair or a designated number of vector elements
VReg	Display the contents of a vector register or a designated number of vector elements
VStat	Display Vector Facility status information, if applicable, for any processor found in a VM dump
XEDIT	Pass the command line to XEDIT for execution.

Figure 25 (Part 2 of 2). Subcommands for DUMPSCAN

Each of the following subcommand descriptions include the subcommand syntax, keywords, operands, and options. Where applicable, the descriptions also include usage notes, examples, or samples of the output to be expected from each subcommand.

The same notational conventions apply to the DUMPSCAN subcommands as described in "Notational Conventions" in Chapter 3.

General Usage Notes:

1. All addresses in this chapter refer to real addresses unless otherwise specified.
2. All dump storage addresses are 31-bit, 4-byte addresses containing 1-to-8 hexadecimal characters unless otherwise specified. Leading zeros can be omitted. For example, if you want to enter address 00012F31, you may enter 00012F31 or 12F31.
3. All processor addresses (*cpuaddr*) are 1- to 4-byte hexadecimal digits.
4. All device numbers are 1-to-4 hexadecimal digits.
5. Logical real device numbers are 1-to-4 hexadecimal digits and are prefixed with the letter L.

FINDSTRG Subcommand

Use the FINDSTRG subcommand to search for a particular string of data in the dump while you are still in an EXEC. If found, the address of the string is returned in a REXX variable.

The format of the FINDSTRG subcommand is as follows:

```
FINDStrg   string  ┌                                          ┐
                   │           ┌                           ┐  │
                   │ fromaddr  │  toaddr    ┌ increment ┐  │  │
                   │    0      │  7FFFFFFF  │     1     │  │  │
                   │           └            └           ┘  ┘  │
                   └                                          ┘

                   ┌                    ┐
                   │ ( VAR  ┌ name   ┐  │
                   │        │ RESULT │  │
                   │        └        ┘  │
                   └                    ┘
```

Where:

string
is a 1- to 8-byte hexadecimal string for which you are searching.

fromaddr
is the 31-bit (1-to-4 byte) hexadecimal starting address for the search. If not specified, this defaults to start at location 0. Leading zeros are not required.

toaddr
is the 31-bit (1-to-4 byte) hexadecimal ending address for the search. If not specified, this defaults to end at location 7FFFFFFF. Leading zeros are not required.

increment
is a 1- to 4-digit hexadecimal number to change the current address after each match attempt.

VAR
is a keyword operand indicating that the following is the user specified REX variable name.

name
is a 1- to 8-character user-specified name of a REXX variable where the results of the FINDSTRG subcommand will be placed.

RESULT
is the default name of the REXX variable if you do not specify a name.

Usage Notes:

1. This subcommand can only be executed from an EXEC. An error message is issued if entered from the command line.
2. Unlike the LOCATE subcommand which accepts either up to 8 EBCDIC characters or 16 hexadecimal digits, the FINDSTRG subcommand only accepts up to 8 hexadecimal digits (4 bytes). If EBCDIC data such as a userid needs to be located, it must be converted to hex first.
3. If the "from" and "to" addresses are not specified, they will default to the beginning and ending of the dump.
4. In order to specify an *increment* both the "from" and "to" addresses must be specified.
5. The address of the first byte of the string, if found, is placed in a REXX variable (RESULT or the user-specified name).
6. If the user wants to look for multiple occurrences of a string within a dump they must update the *fromaddr* after each match. The reuse or = subcommands do not apply to this subcommand.
7. The valid increment range is from X'1' to X'1000'.
8. The following return codes will be returned to the EXEC:

0	=	Successful execution
8	=	String not found
16	=	Invalid operands
20	=	Internal error.

NOTE Subcommand

Use the NOTE subcommand to send text output to the dump viewing facility to be displayed on the terminal, the printer, or both.

The format of the NOTE subcommand is as follows:

NOTE	[*text*]	[PRINT <u>NOPRINT</u>]	[NOTERMINAL <u>TERMINAL</u>]

Where:

text
is the output to be displayed. This includes any leading blanks. The maximum length of the text is 80 bytes. If no text is specified, a blank line is printed or displayed according to the options selected or defaulted. Beginning and ending quotes are mandatory if text is specified.

PRINT
indicates that the text should be sent to the virtual printer. This operand may not be specified in conjunction with the NOPRINT operand.

NOPRINT
indicates that the text should not be sent to the virtual printer. This operand may not be specified in conjunction with the PRINT or NOTERMINAL operands. This operand is the default.

TERMINAL
indicates that the text should be displayed on the terminal. This operand may not be specified with the NOTERMINAL operand. This operand is the default.

NOTERMINAL
indicates that the text should not be displayed on the terminal. This operand may not be specified in conjunction with the TERMINAL or NOPRINT operands.

Usage Notes:

1. This subcommand is only valid when issued from an EXEC.
2. Text which is longer than 80 bytes is truncated to 80 bytes prior to being printed or displayed.
3. The following return codes are returned to the EXEC:

   0 - Successful execution.

   8 - Invalid condition such as conflicting operands, or missing end quote.

   500 - Virtual printer error. (A message indicating this error is displayed on the terminal.)

READSTRG Subcommand

Use the READSTRG subcommand to read data from the dump while you are in an EXEC. You can specify the actual or an indirect address. The data at that address is returned in a REXX variable.

The format of the READSTRG subcommand is as follows:

```
READStrg     address [ % ]   [ length ]   [ ( VAR [ name   ] ]
                     [ ? ]   [   4    ]   [       [ RESULT ] ]
```

Where:

address
is the 31-bit (1-to-4 byte) hexadecimal address from which the data is to be retrieved in the dump. Leading zeros are not required.

%
specifies a 24-bit indirect address. A word (4 bytes) of storage at the specified address is read from the dump and used as the basis for a second read. The data at the second address is returned to the EXEC.

?
specifies a 31-bit indirect address. A word (4 bytes) of storage at the specified address is read from the dump and used as the basis for a second read. The data at the second address is returned to the EXEC.

length
is an optional operand. It is a 1- to 4-digit nonzero hexadecimal number indicating the length in bytes to be returned to the EXEC. The valid range is from X'1' to X'1000'.

4
is the default length of the data to be returned.

VAR
is a keyword operand indicating that the following is the user-specified REXX variable name.

name
is a 1- to 8-character user-specified name of a REXX variable where the results of the READSTRG subcommand are placed.

RESULT
is the default name of the REXX variable that is used if the user does not specify a name.

Usage Notes:

1. This subcommand can only be executed from an EXEC. An error message is issued if entered from the command line.
2. The dump data is translated to EBCDIC and then returned to the EXEC in a REXX variable (that is, in the parameter RESULT or the user-specified name).

3. If only partial data is available in the dump, READSTRG returns only the available data in which case the user should check the length of REXX variable being used.

4. The following return codes will be returned to the EXEC:

   0 = Successful execution
   4 = Partial data returned
   8 = Page not dumped
   16 = Invalid operand
   20 = Internal error.

SCAN Subcommand

Use the SCAN subcommand to process a PF-key assignment or a command string from the system product interpreter.

The format of the SCAN subcommand is as follows:

SCAN	*subcommand* [*operands*]

Where:

subcommand
is any valid DUMPSCAN subcommand

operands
includes any operands for the DUMPSCAN subcommand you requested.

Usage Notes:

1. Use the SCAN service to assign dump viewing facility functions to PF keys. For more information, see "Assigning Program Function Keys to DUMPSCAN Subcommands" on page 19
2. Use the system product interpreter command ADDRESS to effect a temporary or permanent change to the destination of commands. The SCAN environment is addressable from any EXEC2, REXX EXEC, or XEDIT macro during the DUMPSCAN session. For more information, see "What Is an Environment?" on page 16.

Null Line Subcommand

Use the "null line" subcommand to reissue the previous CHAIN, LOCATE, LOCATEUP, BACKWARD, or FORWARD subcommands.

(null line)	

Usage Notes:

1. Pressing the ENTER key with no data entered repeats the previous CHAIN, LOCATE, LOCATEUP, FORWARD, or BACKWARD subcommands with an updated address.
2. Entering a null line is valid for the CHAIN subcommand only if the number of control blocks exceeds 4096. A message is issued when there are more than 4096 control blocks. Entering a null line continues the chain presentation starting with the last address displayed.
3. The running total of all members found are displayed in the output of the reissued CHAIN subcommand.

Response:

Using the null line command re-executes the CHAIN, LOCATE, LOCATEUP, BACKWARD, or FORWARD commands; a full screen of the appropriate data is displayed.

+ and - Subcommands

Use the " + " and "-" subcommands to adjust the address pointer and reissue the DISPLAY subcommand.

+	*increment*

-	*decrement*

Where:

increment
is the hexadecimal number to be added to the address pointer of the last displayed subcommand entered.

For the TRACE subcommand, *increment* is the decimal number of entries you want to move forward.

decrement
is the hexadecimal number to be subtracted from the address pointer of the last displayed subcommand entered.

For the TRACE subcommand, *decrement* is the decimal number of entries you want to move backward.

Usage Notes:

1. The " + " and "-" subcommands can be used after the displaying and scrolling subcommands of the dump viewing facility are issued. These subcommands are: BACKWARD, DISPLAY, FINDMOD, FORWARD, LOCATE, LOCATEUP, and TRACE.
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.
3. These subcommands do not wrap the screen.
4. If the OFFSET operand was specified on the previous DISPLAY subcommand, then issuing the " + " or "-" subcommand continues to display offsets.
5. When scrolling through trace entries, use the increment or decrement variable to specify the number of entries to move forward or backward.

Response:

Provided there is a full screen of data left in the dump, an entire screen of dump data is displayed with the current line position at the calculated address.

&name Subcommand

Use the "&name" subcommand to create a table of frequently used subcommands that may be invoked by another name, or to invoke a subcommand by its other name.

&name &	[*subcommand*]

Where:

name
is the symbolic name you give to the subcommand expression entered in the &name table. The name portion of this subcommand may be 1-to-7 characters in length and must be preceded by the ampersand.

subcommand
is the entire syntax of the subcommand including any operands specified. Entering "&name" without any operands invokes the subcommand.

&
Entering an ampersand (&) alone lists all the table entries.

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 a &name that is not in the table, DUMPSCAN issues an error message.

3. The subcommand in the table is not checked for validity until it is invoked by entering "&name." Only then are errors detected by the appropriate subcommand processor.

4. The PRINT subcommand is not allowed in the &name table.

5. All entries into the &name table are limited to 8 characters for each operand. The "&name" subcommand plus eight operands may be entered, that is, &VM VMDBK *operand2 operand3...... operand8*.

6. Up to 64 operands, including the symbolic names (*&name*), may be contained in the &name table at any one time. The number of symbolic names you are limited to is determined by the number of operands used per subcommand. You can assign more subcommands with three operands (21) than you can with seven operands (nine).

7. If the LOCATE subcommand is placed in the &name table, the maximum string of 8 characters includes the hexadecimal identifier X, the hexadecimal characters, and the quotes (for example, X'13AB4').

Responses:

1. If `&name` is entered, the response is from the subcommand executed.
2. If `&` is entered, a list of the current entries in the &name table is displayed.
3. If `&name subcommand` is entered, the ready response indicates the subcommand has been added to the &name table.

Example:

The "&name" subcommand is useful for command strings that are used constantly. It allows you to shorten a command string to one symbolic name.

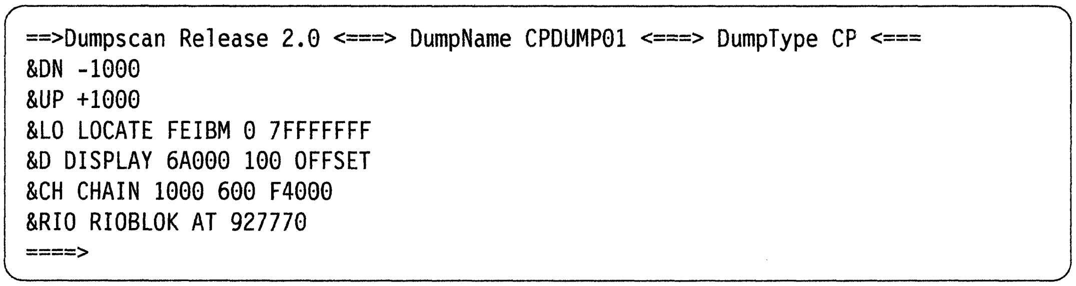
Figure 26 illustrates a sequence of six &name entries being made in the &name table from the command line.

```
= = = = > &dn -1000
= = = = > &up +1000
= = = = > &lo locate feibm 0 7fffffff
= = = = > &d display 6a000 100 offset
= = = = > &ch chain 1000 600 f4000
= = = = > &rio rioblok at 927770
```

Figure 26. A Sequence of &name Commands

After all six entries are made, you can check the &name table by entering the &name table list subcommand "&". This displays the &name table as shown in Figure 27. The subcommand entered is:

```
&
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
&DN -1000
&UP +1000
&LO LOCATE FEIBM 0 7FFFFFFF
&D DISPLAY 6A000 100 OFFSET
&CH CHAIN 1000 600 F4000
&RIO RIOBLOK AT 927770
====>
```

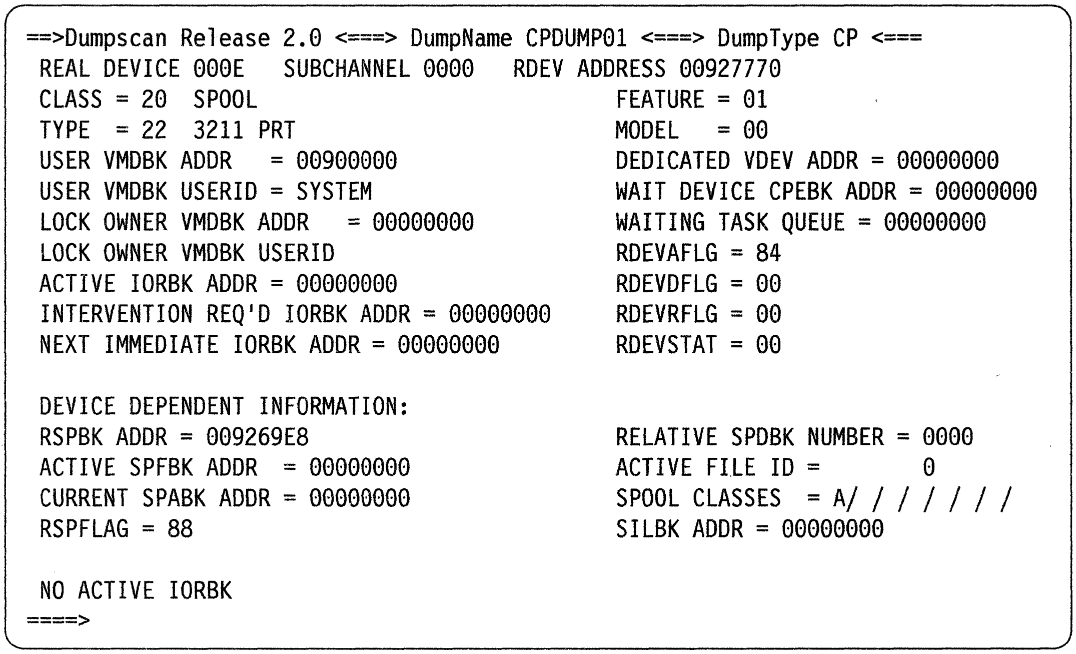
Figure 27. Listing the &name Table

Any time you wish to execute a command string in the &name table, enter the symbolic name corresponding to the desired command. The command in the &name table is processed as if it were just entered manually.

Figure 28 shows the symbolic command *&rio* being entered, and the resulting display of the corresponding subcommand, RIOBLOK AT 927770.

The subcommand entered is:

```
&rio
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 REAL DEVICE 000E   SUBCHANNEL 0000   RDEV ADDRESS 00927770
 CLASS = 20  SPOOL                           FEATURE = 01
 TYPE  = 22  3211 PRT                        MODEL   = 00
 USER VMDBK ADDR   = 00900000                DEDICATED VDEV ADDR = 00000000
 USER VMDBK USERID = SYSTEM                  WAIT DEVICE CPEBK ADDR = 00000000
 LOCK OWNER VMDBK ADDR   = 00000000          WAITING TASK QUEUE = 00000000
 LOCK OWNER VMDBK USERID                     RDEVAFLG = 84
 ACTIVE IORBK ADDR = 00000000                RDEVDFLG = 00
 INTERVENTION REQ'D IORBK ADDR = 00000000    RDEVRFLG = 00
 NEXT IMMEDIATE IORBK ADDR = 00000000        RDEVSTAT = 00

 DEVICE DEPENDENT INFORMATION:
 RSPBK ADDR = 009269E8                       RELATIVE SPDBK NUMBER = 0000
 ACTIVE SPFBK ADDR  = 00000000               ACTIVE FILE ID =        0
 CURRENT SPABK ADDR = 00000000               SPOOL CLASSES  = A/ / / / / / /
 RSPFLAG = 88                                SILBK ADDR = 00000000

 NO ACTIVE IORBK
====>
```

Figure 28. Executing the RIOBLOK Subcommand Using a Symbolic Name

? Subcommand

Use the "?" subcommand to display the last subcommand entered.

?	

Usage Notes:

1. The subcommand that is displayed as a result of a question mark (?) can be reexecuted by pressing the ENTER key. You can also modify the command before entering it again.
2. Successive execution of the "?" subcommand will display the previous subcommands.
3. A synonym cannot be defined for the ? subcommand.
4. The "?" subcommand can be assigned to a PF or a PA key.
5. Anything following a ? is ignored except another ?. Multiple question marks can be specified to retrieve previous subcommands.
6. The results of the execution of the equal (=) subcommand may not be identifical to the results of combining the "?" subcommand and the ENTER key. The "=" subcommand executes the last valid dump viewing facility subcommand.

Response:

The system displays the last command line entered from the terminal.

= Subcommand

Use the "=" subcommand to reexecute the last DUMPSCAN subcommand entered.

=	

BACKWARD Subcommand

The BACKWARD subcommand scrolls backward toward the lowest address in the dump.

The format of the BACKWARD subcommand is as follows:

Backward	

Usage Notes:

1. The BACKWARD subcommand can be used after you issue the DISPLAY, LOCATE, FINDMOD, TRACE, or other scrolling subcommands.
2. The BACKWARD subcommand may be reissued by pressing the ENTER key ("null line" subcommand).
3. ScrollUp and Scroll Up are functionally equivalent synonyms for the BACKWARD subcommand. These are VM/SP Interactive Problem Control System (IPCS) subcommands.
4. For scrolling forward to the highest address in the dump, see the FORWARD subcommand later in this chapter.
5. If you entered the OFFSET operand on the previous DISPLAY subcommand, then the BACKWARD subcommand continues to display data using the specified offsets. Your terminal screen continues to display the original storage address requested.
6. The BACKWARD subcommand does not wrap the screen.
7. You cannot display data with offsets below 0.

Responses:

One full screen of data is presented in both hexadecimal and EBCDIC. For example, if your screen displays 19 lines, the data at the top of the screen (first line) is hex 130 bytes from the last address displayed.

When scrolling after the TRACE subcommand, the format of the next screen is identical to the screen when TRACE was entered. For example, if the previous TRACE subcommand was for FORMAT output, scrolling continues with formatted output.

BLOCK Subcommand

Use the BLOCK subcommand to format control blocks within a dump. You can format the entire block or only selected fields. You can also request that a predefined subset of high interest fields be formatted.

The format of the BLOCK subcommand follows:

```
BLock      { name } address [ BITS ] [ OFFSET ] ┌      ┐
           { *    }                             │PROMPT│
                                                │ALL   │
                                                └      ┘
```

Where:

name
is the 1- to 8-character name of the control block to be formatted.

*****
is an indication to the dump viewing facility that you want it to identify the block using the control block ID (CBID).

Note: See the second usage note for information on address and name verification.

address
is a 1- to 8-digit hexadecimal address indicating the storage location of the control block.

BITS
is a keyword indicating that the bits within a byte should be formatted when possible.

Note: If not specified, then only information down to the byte level is formatted.

OFFSET
is a keyword indicating that the display should be formatted using relative offsets from the start of the control block instead of actual addresses.

ALL
is a keyword indicating that all fields within the control block are to be formatted.

PROMPT
is a keyword indicating that the user would like to be prompted for the field names to be displayed.

Usage Notes:

1. If you do not specify either ALL or PROMPT, only fields that are marked as "default" in the table for the control block are displayed.
2. BLOCK does not verify that the control block name provided is valid for the address given. If the user gives BLOCK the wrong address, BLOCK simply maps the storage into the control block definition as if the address were correct.
3. When using the PROMPT function of BLOCK, you can display a selected group of fields and then discover that additional fields need to be displayed. In this instance you need not retype all of the fields entered previously. When prompted for the fields to be displayed, you have the option of reusing the old

fields and having the new fields added to the display. This is accomplished by entering an equal sign (=) followed by the name of the new fields. BLOCK redisplays the previous fields followed by the new ones. The following is an example of the use of PROMPT with an equal sign.

On a prompt for field names to be displayed, you have entered the following:

```
field1 field2 field3 field4
```

You now want to add fields 5 and 6 to the display. After reentering the BLOCK command with the PROMPT keyword, you are again prompted for the field names to be displayed, and enter the following:

```
= field5 field6
```

BLOCK displays the fields in the order that you entered them. The program does not try to order the fields nor does it check for duplicate fields. BLOCK only verifies that a name entered by the user actually exists in the control block definition.

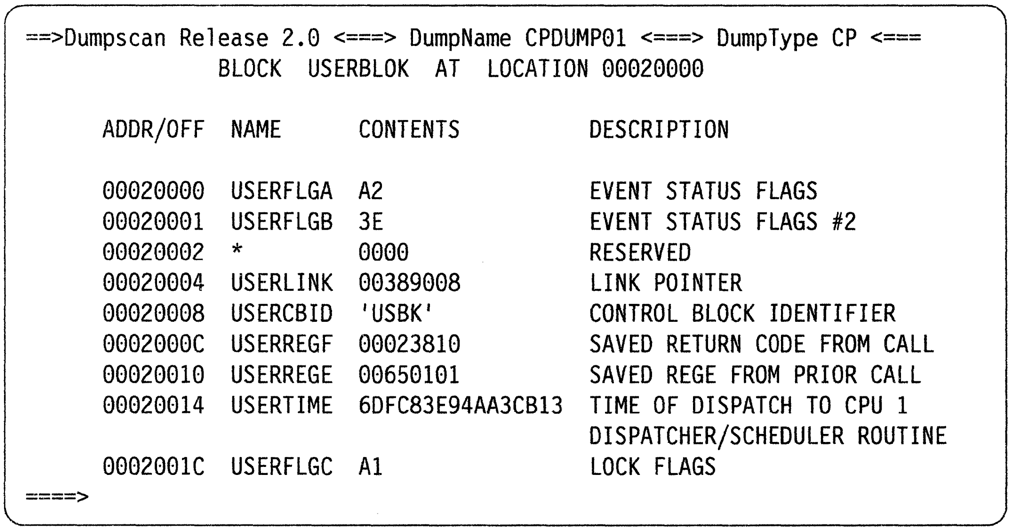
4. You can flag any field as a default field with the exception of BIT subrecord fields. If a BIT field is flagged as a default field and the BITS keyword was specified, the bits are displayed. If just the BIT subrecord field is flagged, the flag is ignored.

Examples:

The following examples demonstrate how the various BLOCK operand and keyword combinations provide you with the control block data you need.

The user enters the following:

```
block userblok 20000 all
```

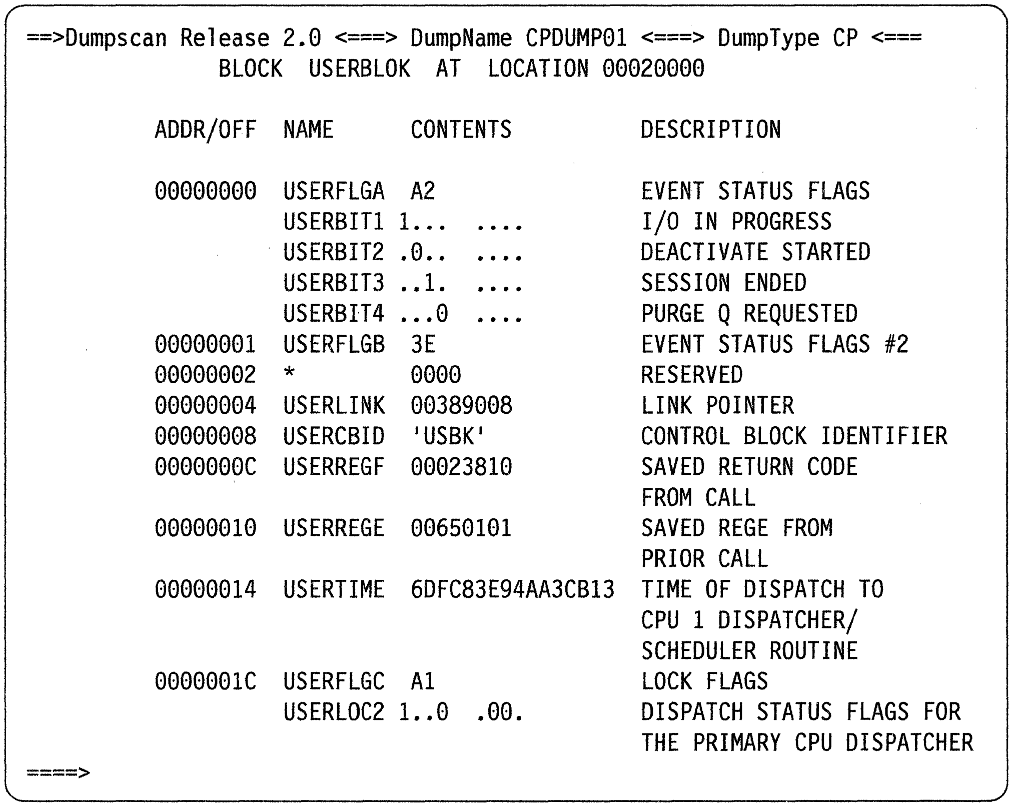


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
               BLOCK  USERBLOK  AT  LOCATION 00020000

      ADDR/OFF  NAME      CONTENTS          DESCRIPTION

      00020000  USERFLGA  A2                EVENT STATUS FLAGS
      00020001  USERFLGB  3E                EVENT STATUS FLAGS #2
      00020002  *         0000              RESERVED
      00020004  USERLINK  00389008          LINK POINTER
      00020008  USERCBID  'USBK'            CONTROL BLOCK IDENTIFIER
      0002000C  USERREGF  00023810          SAVED RETURN CODE FROM CALL
      00020010  USERREGE  00650101          SAVED REGE FROM PRIOR CALL
      00020014  USERTIME  6DFC83E94AA3CB13  TIME OF DISPATCH TO CPU 1
                                            DISPATCHER/SCHEDULER ROUTINE
      0002001C  USERFLGC  A1                LOCK FLAGS
====>
```

Figure 29. BLOCK with the ALL Keyword

The user enters the following:

```
block userblok 20000 bits offset
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
               BLOCK  USERBLOK  AT  LOCATION 00020000

         ADDR/OFF  NAME      CONTENTS          DESCRIPTION

         00000000  USERFLGA  A2                EVENT STATUS FLAGS
                   USERBIT1 1...  ....         I/O IN PROGRESS
                   USERBIT2 .0..  ....         DEACTIVATE STARTED
                   USERBIT3 ..1.  ....         SESSION ENDED
                   USERBIT4 ...0  ....         PURGE Q REQUESTED
         00000001  USERFLGB  3E                EVENT STATUS FLAGS #2
         00000002  *         0000              RESERVED
         00000004  USERLINK  00389008          LINK POINTER
         00000008  USERCBID  'USBK'            CONTROL BLOCK IDENTIFIER
         0000000C  USERREGF  00023810          SAVED RETURN CODE
                                               FROM CALL
         00000010  USERREGE  00650101          SAVED REGE FROM
                                               PRIOR CALL
         00000014  USERTIME  6DFC83E94AA3CB13  TIME OF DISPATCH TO
                                               CPU 1 DISPATCHER/
                                               SCHEDULER ROUTINE
         0000001C  USERFLGC  A1                LOCK FLAGS
                   USERLOC2 1..0  .00.         DISPATCH STATUS FLAGS FOR
                                               THE PRIMARY CPU DISPATCHER
====>
```

Figure 30. BLOCK with the BITS and OFFSET keywords

Block Tables

Block Definitions: The BLOCK subcommand displays a control block by mapping the contents of storage into a predefined format. This is accomplished by using a previously defined description of the control block. The control block descriptions are formatted as follows:

- Each field within a control block is described to the BLOCK subcommand in a single record called a block definition record.
- The entire control block description is made up of groups of these definition records and is called a block definition.
- The block definitions are stored in large files and are called block table files.

Here are examples of the above.

```
040        8     (8) CHARACTER      4  USERCBID         CONTROL BLOCK
0A0                                                     IDENTIFIER
```

Figure 31. A Block Definition Record: A Single Field within a Control Block

Note: See "Block Descriptor Record Format" on page 79 for record layout.

```
*** USER BLOCK NONDISPLAYABLE COMMENT
000        0     (0) STRUCTURE    29  USERBLOK
010        0     (0) BIT           1  USERFLGA       EVENT STATUS FLAGS
011              1... ....            USERBIT1       I/O IN PROGRESS
012              .1.. ....            USERBIT2       DEACTIVATE STARTED
012              ..1. ....            USERBIT3       SESSION ENDED
012              ...1 ....            USERBIT4       PURGE Q REQUESTED
012              .... 1...            USERBIT5       PURGE Q COMPLETED
012              .... .1..            USERBIT6       DEACTIVATE COMPLETE
012              .... ..1.            USERBIT7       I/O COMPLETE
013              .... ...1            USERBIT8       PURGE Q I/O WAIT
010        1     (1) BIT           1  USERFLGB       EVENT STATUS FLAGS #2
020        2     (2) FIXED(U)      2  *              RESERVED
040        4     (4) POINTER       4  USERLINK       LINK POINTER
030        8     (8) CHARACTER     4  USERCBID       CONTROL BLOCK
0A0                                                  IDENTIFIER
060       12     (C) OFFSET        4  USERREGF       SAVED RETURN CODE
0A0                                                  FROM CALL
080       16    (10) FIXED(S)      4  USERREGE       SAVED REGE FROM
0A0                                                  PRIOR CALL
090       20    (14) AREA          8  USERTIME       TIME OF DISPATCH TO
0A0                                                  CPU 1 DISPATCHER/
0A0                                                  SCHEDULER ROUTINE
010       28    (1C) BIT           1  USERFLGC       LOCK FLAGS
011              1..1 .11.            USERLOC2       DISPATCH STATUS FLAGS
0A1                                                  FOR THE PRIMARY CPU
0A1                                                  DISPATCHER
```

Figure 32. A Block Definition: The Group of Records Defining a Single Block

```
*** FIRST BLOCK DEFINITION
000       0    (0) STRUCTURE   29  USERBLOK
010       0    (0) BIT          1  USERFLGA       EVENT STATUS FLAGS
011            1... ....           USERBIT1       I/O IN PROGRESS
012            .1.. ....           USERBIT2       DEACTIVATE STARTED
012            ..1. ....           USERBIT3       SESSION ENDED
013            ...1 ....           USERBIT4       PURGE Q REQUESTED
010       1    (1) BIT          1  USERFLGB       EVENT STATUS FLAGS #2
020       2    (2) FIXED(U)     2  *              RESERVED
040       4    (4) POINTER      4  USERLINK       LINK POINTER
030       8    (8) CHARACTER    4  USERCBID       CONTROL BLOCK
0A0                                               IDENTIFIER
060      12    (C) OFFSET       4  USERREGF       SAVED RETURN CODE
0A0                                               FROM CALL
080      16   (10) FIXED(S)     4  USERREGE       SAVED REGE FROM
0A0                                               PRIOR CALL
090      20   (14) AREA         8  USERTIME       TIME OF DISPATCH TO
0A0                                               THE CPU 1 DISPATCHER/
0A0                                               SCHEDULER ROUTINE
010      28   (1C) BIT          1  USERFLGC       LOCK FLAGS
011            1..1 .11.           USERLOC2       DISPATCH STATUS FLAGS
0A1                                               FOR THE PRIMARY CPU
0A1                                               DISPATCHER
*** SECOND BLOCK DEFINITION
000       0    (0) STRUCTURE   34  BLOKBLOK
030       0    (0) CHARACTER    4  BLOKCBID       THE BLOCKS' ID FIELD
040       4    (4) POINTER      4  BLOKLINK       LINK POINTER
040       8    (8) POINTER      4  BLOKSTAT       ACTIVATE STATUS
040      12    (C) POINTER      4  BLOKWAIT       WAIT EVENT BLOCK
0A0                                               POINTER
090      24   (18) AREA         8  BLOKTIME       TIME OF DISPATCH
010      32   (20) BIT          1  BLOKWFLG       WAIT FLAGS
010      33   (21) BIT          1  *              RESERVED
```

Figure 33. A Block Table File Containing Multiple Block Definitions

Block Descriptor Record Format: You can build a block definition record by using the following format:

```
040      8    (8) CHARACTER     4  USERCBID        CONTROL BLOCK
0A0                                                IDENTIFIER
```

Figure 34. The Block Descriptor Record

Where:

Table 1 (Page 1 of 2). Block Descriptor Record Format

Columns	Field Name	Field Description
01-03	Key field	A 3-digit value that provides information about the field to BLOCK. The digits are used as follows: Column 1 Default character field. Column 2-3 Data type indicator. The possible combinations are: **00** STRUCTURE - Start of definition. **10** BIT - BIT data. **11** BIT - Bit subrecord start (on byte boundary). **12** BIT - Bit subrecord. **13** BIT - Bit subrecord end. **20** FIXED(U) - Fixed unsigned data. **30** CHARACTER - Character data. **40** POINTER - Pointer data. **50** Reserved for future IBM use. **60** OFFSET - Offset data. **70** Reserved for future IBM use. **80** FIXED(S) - Fixed signed data. **90** AREA - Area data. **A0** Independent record comment. **A1** Independent subrecord comment. ** Nondisplayable comment
07-11	Decimal offset	The fields' offset within the block. This field is right-justified.
13-18	Hexadecimal offset	The fields' offset within the block. This field is right-justified and bracketed by parentheses.

Table 1 (Page 2 of 2). Block Descriptor Record Format

Columns	Field Name	Field Description
20-28	Data type	The type of data contained in the field.
30-34	Field/block size	The decimal size of the field in bytes This field is right-justified.
37-50	Field/block name	The name of the field. The name field can also contain an array element count value. This count must be bracketed by parentheses; for example, TESTDATA(1024). This indicates that TESTDATA is an array of 1024 elements.
52-79	Field comments	The comments describing the field

Note: Columns not specifically assigned to a field must contain blanks.

Block Descriptor Record - BIT Subrecord Format: There is an alternate format for bit subrecords format records. See the following example:

```
011            1... ....           USERBIT1       I/O IN PROGRESS
```

Figure 35. Alternate Format for Bit Subrecord

Where:

Table 2. Block Descriptor Record – BIT Subrecord Format

Columns	Field Name	Field Description
01-03	Key field	A 3-digit value that provides information about the field to BLOCK. The digits are used as follows: Column 1 Default character field. Column 2-3 Data type indicator. The possible combinations are: **10** BIT - BIT data. **11** BIT - Bit subrecord start (on byte boundary). **12** BIT - Bit subrecord. **13** BIT - Bit subrecord end.
16-25	Bit map field	The bit placement map used to format bit display.
37-50	Field/block name	The name of the bits.
52-79	Field comments	The comments describing the field.

Note: Columns not specifically assigned to a field must contain blanks.

Default Display Fields: You can get a display of the default display fields by omitting the ALL and PROMPT options on the BLOCK subcommand. You select the fields to be displayed as default fields. This is accomplished by specifying a default field indicator in the definition header record of a block definition.

```
000        0     (0) STRUCTURE    29  USERBLOK
```

Figure 36. Block Header Record with No Default Settings

Note: If you want to use a nondisplayable comment, you cannot use an asterisk for the default character within that block.

The default character for a block is set by altering the first character of the KEY field. For example, we will use the letter D to signify that a field is a default field. The user may specify any valid EBCDIC character as the default character.

```
D00        0     (0) STRUCTURE    29  USERBLOK
```

Figure 37. Block Header Record Set to Default of D

Altering a block definition header record KEY field indicates to the BLOCK subcommand that any field record within the definition that has the same character in the first character position of the its own KEY field is to be considered a default field. For example, the following block field would be considered a default field.

```
D30        8     (8) CHARACTER     4  USERCBID       CONTROL BLOCK
DA0                                                  IDENTIFIER
```

Figure 38. Field record - Set to Default of D

Tailoring a Block Table File: You can customize an existing definition by:

- Changing the control block name in the header record
- Changing the names of various fields
- Deleting fields of no interest
- Adding new fields, such as bit subrecords
- Changing comments for the field.

Essentially, you can modify anything in the definition with the following cautions:

- The control block and field size values. The field size value tells BLOCK how much data to map into that particular field. The block size value tells BLOCK how much data to get from the dump to map into the block.
- The field offset values. These fields tell block exactly where, within the storage of the control block, the data to be mapped is located.

In this case, the control block name is changed.

```
D00        0     (0) STRUCTURE    29  USERBLOK
```

is changed to:

```
D00        0     (0) STRUCTURE    29  MYBLOCK1
```

In this case, the field name and the comment are changed.

```
D30        8     (8) CHARACTER     4  USERCBID        CONTROL BLOCK
DA0                                                   IDENTIFIER
```

is changed to:

```
D40        8     (8) CHARACTER     4  BLOCKID         THE BLOCKS'
DA0                                                   ID FIELD
```

In this case, a bit breakdown is added to a definition:

```
000      0    (0) STRUCTURE    29  USERBLOK
010      0    (0) BIT           1  USERFLGA       EVENT STATUS FLAGS
011           1... ....            USERBIT1       I/O IN PROGRESS
012           .1.. ....            USERBIT2       DEACTIVATE STARTED
012           ..1. ....            USERBIT3       SESSION ENDED
013           ...1 ....            USERBIT4       PURGE Q REQUESTED
010      1    (1) BIT           1  USERFLGB       EVENT STATUS FLAGS #2
020      2    (2) FIXED(U)      2  *              RESERVED
040      4    (4) POINTER       4  USERLINK       LINK POINTER
030      8    (8) CHARACTER     4  USERCBID       CONTROL BLOCK
0A0                                               IDENTIFIER
060     12    (C) OFFSET        4  USERREGF       SAVED RETURN CODE
0A0                                               FROM CALL
080     16   (10) FIXED(S)      4  USERREGE       SAVED REGE FROM
0A0                                               PRIOR CALL
090     20   (14) AREA          8  USERTIME       TIME OF DISPATCH TO
0A0                                               THE CPU 1 DISPATCHER/
0A0                                               SCHEDULER ROUTINE
010     28   (1C) BIT           1  USERFLGC       LOCK FLAGS
011           1..1 .11.            USERLOC2       DISPATCH STATUS FLAGS
0A1                                               FOR THE PRIMARY CPU
0A1                                               DISPATCHER
```

is changed to:

```
000      0    (0) STRUCTURE    29  USERBLOK
010      0    (0) BIT           1  USERFLGA       EVENT STATUS FLAGS
011           1... ....            USERBIT1       I/O IN PROGRESS
012           .1.. ....            USERBIT2       DEACTIVATE STARTED
012           ..1. ....            USERBIT3       SESSION ENDED
012           ...1 ....            USERBIT4       PURGE Q REQUESTED
012           .... 1...            USERBIT5       PURGE Q COMPLETED
012           .... .1..            USERBIT6       DEACTIVATE COMPLETE
012           .... ..1.            USERBIT7       I/O COMPLETE
013           .... ...1            USERBIT8       PURGE Q I/O WAIT
010      1    (1) BIT           1  USERFLGB       EVENT STATUS FLAGS #2
020      2    (2) FIXED(U)      2  *              RESERVED
040      4    (4) POINTER       4  USERLINK       LINK POINTER
030      8    (8) CHARACTER     4  USERCBID       CONTROL BLOCK
0A0                                               IDENTIFIER
060     12    (C) OFFSET        4  USERREGF       SAVED RETURN CODE
0A0                                               FROM CALL
080     16   (10) FIXED(S)      4  USERREGE       SAVED REGE FROM
0A0                                               PRIOR CALL
090     20   (14) AREA          8  USERTIME       TIME OF DISPATCH TO
0A0                                               THE CPU 1 DISPATCHER/
0A0                                               SCHEDULER ROUTINE
010     28   (1C) BIT           1  USERFLGC       LOCK FLAGS
011           1..1 .11.            USERLOC2       DISPATCH STATUS FLAGS
0A1                                               FOR THE PRIMARY CPU
0A1                                               DISPATCHER
```

In this case, USERBIT5 through USERBIT8 are being deleted.

```
000       0     (0) STRUCTURE    29  USERBLOK
010       0     (0) BIT           1  USERFLGA        EVENT STATUS FLAGS
011             1... ....            USERBIT1        I/O IN PROGRESS
012             .1.. ....            USERBIT2        DEACTIVATE STARTED
012             ..1. ....            USERBIT3        SESSION ENDED
012             ...1 ....            USERBIT4        PURGE Q REQUESTED
012             .... 1...            USERBIT5        PURGE Q COMPLETED
012             .... .1..            USERBIT6        DEACTIVATE COMPLETE
012             .... ..1.            USERBIT7        I/O COMPLETE
013             .... ...1            USERBIT8        PURGE Q I/O WAIT
010       1     (1) BIT           1  USERFLGB        EVENT STATUS FLAGS #2
020       2     (2) FIXED(U)      2  *               RESERVED
040       4     (4) POINTER       4  USERLINK        LINK POINTER
030       8     (8) CHARACTER     4  USERCBID        CONTROL BLOCK
0A0                                                  IDENTIFIER
060      12     (C) OFFSET        4  USERREGF        SAVED RETURN CODE
0A0                                                  FROM CALL
080      16    (10) FIXED(S)      4  USERREGE        SAVED REGE FROM
0A0                                                  PRIOR CALL
090      20    (14) AREA          8  USERTIME        TIME OF DISPATCH TO
0A0                                                  THE CPU 1 DISPATCHER/
0A0                                                  SCHEDULER ROUTINE
```

is changed to:

```
000       0     (0) STRUCTURE    29  USERBLOK
010       0     (0) BIT           1  USERFLGA        EVENT STATUS FLAGS
011             1... ....            USERBIT1        I/O IN PROGRESS
012             .1.. ....            USERBIT2        DEACTIVATE STARTED
012             ..1. ....            USERBIT3        SESSION ENDED
013             ...1 ....            USERBIT4        PURGE Q REQUESTED
010       1     (1) BIT           1  USERFLGB        EVENT STATUS FLAGS #2
020       2     (2) FIXED(U)      2  *               RESERVED
040       4     (4) POINTER       4  USERLINK        LINK POINTER
030       8     (8) CHARACTER     4  USERCBID        CONTROL BLOCK
0A0                                                  IDENTIFIER
060      12     (C) OFFSET        4  USERREGF        SAVED RETURN CODE
0A0                                                  FROM CALL
080      16    (10) FIXED(S)      4  USERREGE        SAVED REGE FROM
0A0                                                  PRIOR CALL
090      20    (14) AREA          8  USERTIME        TIME OF DISPATCH TO
0A0                                                  THE CPU 1 DISPATCHER/
0A0                                                  SCHEDULER ROUTINE
```

Creating Block Files: You can alter the block definitions and create new definitions in user block table files. You must observe the following limitations, however:

- A block table file must have a logical record size of 80.
- The record format must be fixed.
- The file cannot exceed 32 656 records in length.

- A maximum of 2048 blocks can be defined per block table.
- The filename of the file must be unique. This is necessary to ensure the right block definitions are loaded into BLOCK during initialization. If the name is not unique, BLOCK loads the first occurrence of the name based on the CMS minidisk search order.

Creating BLOCK Control Files: Once the BLOCK table file is built, you must determine in which of the BLOCK control files the new definition filename should reside, and what order position the file will occupy.

A maximum of four block table files can be used in a single block session.

The block control file is a file that tells BLOCK which block table files to load during initialization. The definition files loaded depend on the type of dump the user is currently examining. When BLOCK initialization starts, BLOCK uses the dump type to search a table called HCSTBL. See Appendix E for more information on HCSTBL. When a match is found on the DUMPTYPE, the entry in the HCSTBL is checked for a block control file. If that exists, the suffix is appended to the characters HCS$ to form the filename of the a control file name (the filetype is always TABLE) for that dump type. BLOCK then searches for that control file and extracts the names of the block definition files for the dump. Once you know the name of the control file for the dump being examined, you simply put the name of the new file in the order position desired. The position of the file is important. BLOCK searches for control block names sequentially. This means that the first definition file is searched first, the second searched second, and so on. Once BLOCK finds a match to the name entered on BLOCK invocation, the file search stops and BLOCK maps the data based on that definition. BLOCK does not recognize duplicate names within a definition file.

Once you have added the new definition filenames, the file should be saved on a minidisk that is ahead of the original control file disk in the CMS search order.

The following is an example of a simple BLOCK control file. The file has records 80 bytes long. In this example, MYFILE CBMAP is added to a BLOCK control file.

```
**************************************************************
* The base CP control blocks for VM/SP                       *
**************************************************************


CPBLOCK1 CBMAP
CPBLOCK2 CBMAP
```

is changed to:

```

**************************************************************
* The base CP control blocks for VM/SP                       *
**************************************************************


MYFILE CBMAP
CPBLOCK1 CBMAP
CPBLOCK2 CBMAP
```

CHAIN Subcommand

Use the CHAIN subcommand to do any of the following:

- Display the addresses for the control blocks on a chain
- Display the data in the control blocks on a chain
- Display a count of the number of control blocks on a chain
- Detect any loops in a chain of control blocks.

This subcommand accepts a 24- or 31-bit qualifier for the address.

The format of the CHAIN subcommand is as follows:

```
CHain    address linkdisp [ ? ]  endval  [ LIST   [ 0         ] ]
                          [ % ]          [        [ offset    ] ]
                                         [                      ]
                                         [ COUNT  [ 500       ] ]
                                         [        [ increment ] ]
                                         [                      ]
                                         [ length               ]
```

Where:

address
is a 1- to 8-digit hexadecimal address specifying the starting address of the first control block on the chain.

linkdisp
is a 1- to 6-digit hexadecimal operand specifying the displacement into the current control block where a pointer to the next control block in the chain is located. The default is ? (question mark), which is 31-bit addressing. If you wish to override the default addressing, enter the % operand for 24-bit addressing. The valid range of the linkdisp operand is from hex 0 to FFFFFF.

endval
is a 1- to 8-digit hexadecimal operand specifying the value of the pointer in the last block of the chain.

LIST
indicates that only a list of control block addresses and a decimal count of the control blocks be displayed. The control blocks themselves are not to be displayed. LIST is the default for the command.

offset
is a 1- to 6-digit hexadecimal offset indicating the starting address from which 4 contiguous bytes of data are to be displayed. The valid range of the offset operand is from hex 0 to FFFFFF. The default is 0.

COUNT
is a keyword specifying that only a count of the total number of control blocks on the chain and not the addresses or control blocks themselves be displayed.

increment

is a 1- to 4-digit decimal number designating how often the following message is issued: "nnnn ENTRIES - PROCESSING CONTINUES." The default is 500 entries.

length

is a 1- to 4-digit hexadecimal operand indicating the number of bytes to be displayed. The valid range of the length operand is from hex 0 to 1000. However, the control block itself may be larger.

Usage Notes:

1. The default indirect addressing mode is ? (question mark), which is 31-bit addressing mode. You can override the default addressing by specifying a percent sign (%) for 24-bit addressing mode.
2. If the number of control blocks on the chain exceeds 4096, a message is issued. Entering a null line continues the chain presentation starting with the last address displayed.
3. If you restart chain processing with the null line, the last address displayed becomes the first on the new chain.
4. If the address of the next control block in the chain has already been found in the current group of 4096 blocks, an error message is issued and processing ends.
5. If a loop of greater than 4096 entries exists it is not detected.

Responses:

The hexadecimal address of each block found in the chain and a decimal count of the number of blocks found is displayed.

Example:

Figure 39 shows an example of a chain of control blocks.

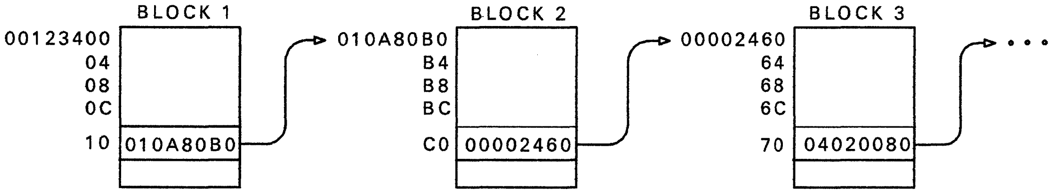


Figure 39. Example of a Chain of Control Blocks

Assume you know the following about the chain of control blocks:

1. All the blocks in the chain have the same format.
2. The address of the first block is 00123400.
3. The pointer to the next block in the chain is at offset hex 10 into the block.
4. The last block in the chain contains a pointer value of zero.

If you wanted to view the addresses of the control blocks, you would enter this subcommand:

```
chain 123400 10 0
```

The output displayed appears in a list format:

```
CHAIN 123400 10 0
CB # 0001 AT 00123400
CB # 0002 AT 010A80B0
CB # 0003 AT 00002460
CB # 0004 AT 04020080

0004 ENTRIES WERE FOUND IN THE CHAIN
```

If you wanted to view the addresses of a chain of control blocks, but with 24-bit addressing specified, you would enter this command:

```
chain 203010 4% 0
```

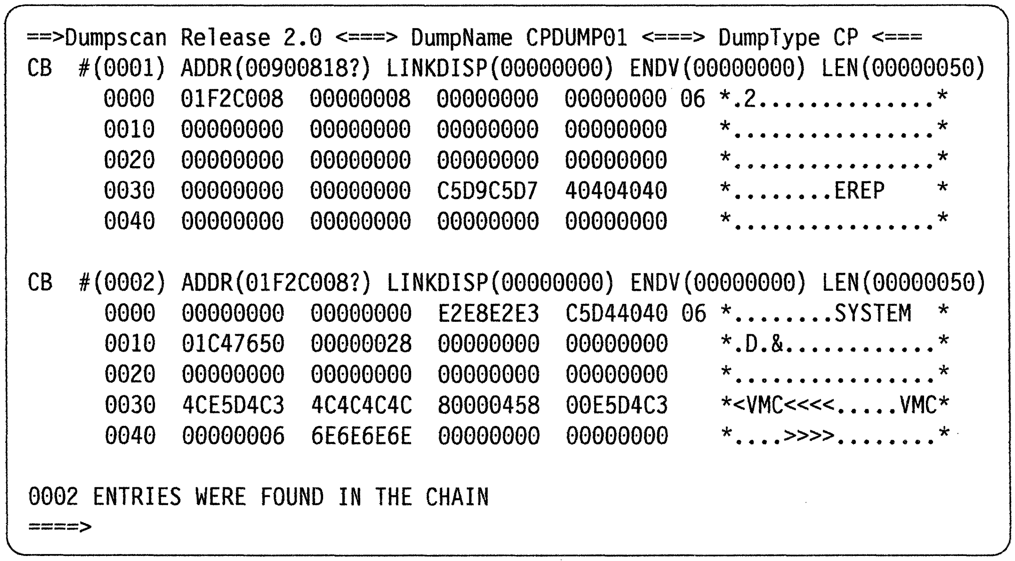
The output displayed appears in a list format:

```
CHAIN 203010 4% 0
CB # 0001 AT 00203010
CB # 0002 AT 00203330
CB # 0003 AT 00203120
CB # 0004 AT 00203110
CB # 0005 AT 00203100
CB # 0006 AT 002030F0
CB # 0007 AT 002030E0
CB # 0008 AT 002030D0
         •
         •
```

The list would continue until all the blocks on the chain are listed. As with the example for 31-bit addressing, the total number of blocks on the chain are listed for 24-bit addressing requests. Figure 40 shows an example of the output from the CHAIN subcommand when the length operand is specified.

The subcommand entered is:

```
chain 900818 0 0 50
```

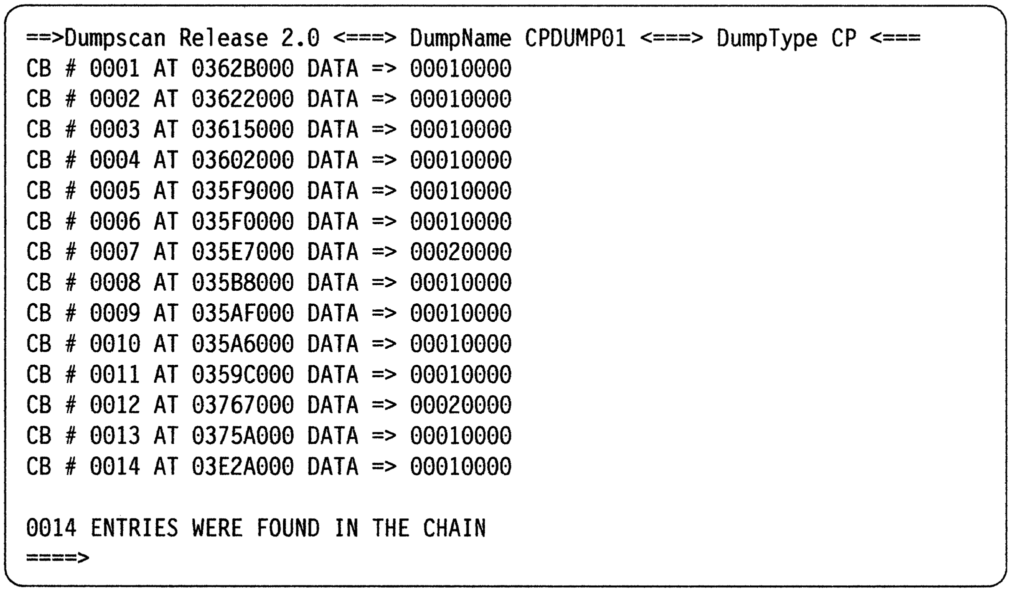


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
CB  #(0001) ADDR(00900818?) LINKDISP(00000000) ENDV(00000000) LEN(00000050)
     0000  01F2C008  00000008  00000000  00000000 06 *.2..............*
     0010  00000000  00000000  00000000  00000000    *................*
     0020  00000000  00000000  00000000  00000000    *................*
     0030  00000000  00000000  C5D9C5D7  40404040    *........EREP    *
     0040  00000000  00000000  00000000  00000000    *................*

CB  #(0002) ADDR(01F2C008?) LINKDISP(00000000) ENDV(00000000) LEN(00000050)
     0000  00000000  00000000  E2E8E2E3  C5D44040 06 *........SYSTEM  *
     0010  01C47650  00000028  00000000  00000000    *.D.&............*
     0020  00000000  00000000  00000000  00000000    *................*
     0030  4CE5D4C3  4C4C4C4C  80000458  00E5D4C3    *<VMC<<<<.....VMC*
     0040  00000006  6E6E6E6E  00000000  00000000    *....>>>>........*

0002 ENTRIES WERE FOUND IN THE CHAIN
====>
```

Figure 40. Sample Output of a CHAIN Subcommand with a Length Specified

Figure 41 shows an example of the output from the CHAIN subcommand when LIST and an offset are specified. The subcommand entered is:

```
chain 362b000 600 0 list 540
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
CB # 0001 AT 0362B000 DATA => 00010000
CB # 0002 AT 03622000 DATA => 00010000
CB # 0003 AT 03615000 DATA => 00010000
CB # 0004 AT 03602000 DATA => 00010000
CB # 0005 AT 035F9000 DATA => 00010000
CB # 0006 AT 035F0000 DATA => 00010000
CB # 0007 AT 035E7000 DATA => 00020000
CB # 0008 AT 035B8000 DATA => 00010000
CB # 0009 AT 035AF000 DATA => 00010000
CB # 0010 AT 035A6000 DATA => 00010000
CB # 0011 AT 0359C000 DATA => 00010000
CB # 0012 AT 03767000 DATA => 00020000
CB # 0013 AT 0375A000 DATA => 00010000
CB # 0014 AT 03E2A000 DATA => 00010000

0014 ENTRIES WERE FOUND IN THE CHAIN
====>
```

Figure 41. Sample Output of a CHAIN Subcommand with LIST and an Offset Specified

CMS Subcommand

Use the CMS subcommand to enter the CMS subset environment.

The format of the CMS subcommand is as follows:

CMS	[*commandline*]

Where:

commandline
is any valid CMS command.

Usage Notes:

1. If you enter the CMS subcommand without an operand, you enter CMS subset mode.
2. If you try to execute a CMS command that terminates abnormally, changes during the dump viewing session can be lost. You should try to save the current session file before using the CMS subcommand.
3. Any CMS command should be prefaced with CMS to prevent the dump viewing facility or XEDIT from decoding the subcommand. This should be done to prevent cases where a CMS command can be interpreted as a dump viewing facility or XEDIT subcommand.
4. Using CMS commands that run in the user area has unpredictable effects on DUMPSCAN, usually resulting in termination.

CPU Subcommand

Use this subcommand to display the CPU address and the prefix register value for each processor in the dump.

The format of the CPU subcommand is as follows:

CPU	

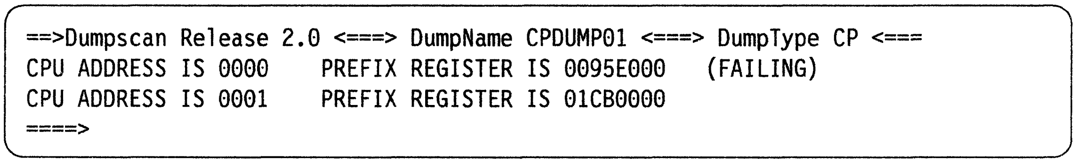
Usage Notes:

1. This subcommand does not clear the screen prior to displaying information.
2. The failing processor is always listed first.
3. For more information about any processor in the system, use the REGS, CREGS, or GREGS subcommands.

Sample Output:

Figure 42 illustrates the output you receive after entering the CPU subcommand. The subcommand entered is:

```
cpu
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
CPU ADDRESS IS 0000    PREFIX REGISTER IS 0095E000   (FAILING)
CPU ADDRESS IS 0001    PREFIX REGISTER IS 01CB0000
====>
```

Figure 42. Sample Output of a CPU Subcommand

CREGS Subcommand

Use the CREGS subcommand to display the control registers for a specified CPU address.

The format of the CREGS subcommand is as follows:

Cregs	[*cpuaddr*]

Where:

cpuaddr
is a 1- to 4-digit hexadecimal number specifying the processor address for which the information is to be displayed.

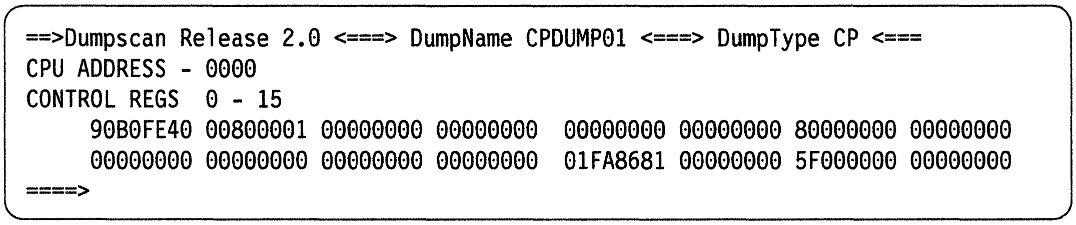
Usage Notes:

1. If the *cpuaddr* operand is not specified, it defaults to the failing processor for a CP abend dump, the IPL processor for a stand-alone dump, or the processor on which the CP VMDUMP command was issued for the virtual machine.
2. Use the CPU subcommand to obtain the processor addresses in the dump.

Sample Output:

Figure 43 illustrates the output of the CREGS subcommand for the failing processor. The subcommand entered is:

```
cregs
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
CPU ADDRESS - 0000
CONTROL REGS  0 - 15
     90B0FE40 00800001 00000000 00000000  00000000 00000000 80000000 00000000
     00000000 00000000 00000000 00000000  01FA8681 00000000 5F000000 00000000
====>
```

Figure 43. Sample Output of a CREGS Subcommand

DISPLAY Subcommand

Use the DISPLAY subcommand to display areas of the dump. You can specify an address, an indirect address, or request data to be displayed by offsets from an address.

The format of the DISPLAY subcommand is as follows:

```
  Display    address ┌ % ┐  [length]  [OFFSET]
                     │ ? │
                     └   ┘
```

Where:

address
is the 31-bit (4-byte) hexadecimal address from which the data is to be displayed.

%
specifies a 24-bit indirect address. A word (4 bytes) of storage at the specified address is read from the dump. The low-order 24 bits are used to compute the address that is displayed.

?
specifies a 31-bit indirect address. A word (4 bytes) of storage at the specified address is read in from the dump. The low-order 31 bits are used to compute the address displayed.

length
is an optional operand. It is a 1- to 4-digit nonzero hexadecimal number indicating the length in bytes to be displayed. The valid range is from hex 1 to FFFF. If this is specified, the screen is **not** cleared. If it is not specified, the screen is cleared and the output is displayed.

One screen of dump date is presented in both hexadecimal and EBCDIC.

OFFSET
is an optional operand. If you specify it, the leftmost column of the output contains the offsets from the input address instead of the storage address of the data. The data is displayed to the right of the column of offsets.

Usage Notes:

1. A period (.) used as a delimiter between the address and the length is acceptable. If the indirect addressing qualifier is specified, the delimiter should follow the qualifier.
2. A "T" preceding the address operand (for example, Taddress) is used to provide compatibility with the CP DISPLAY command. The DISPLAY subcommand always provides the EBCDIC translation whether "T" is specified or not.
3. If you specified an indirect address, the resulting address appears in parentheses in the output as part of the command line.
4. The minimum output is one 16-byte line with EBCDIC translation.
5. If you specify the length operand, the resulting display is rounded to start and end on a hex 10 boundary (see Figure 47).
6. Each line includes 16 bytes of hexadecimal data with the EBCDIC translation. In addition, the key of the page is displayed on the first line and subsequent

page boundaries. It appears between the hexadecimal data and its EBCDIC translation.

7. OFFSET and length are operands that may be specified in any order.

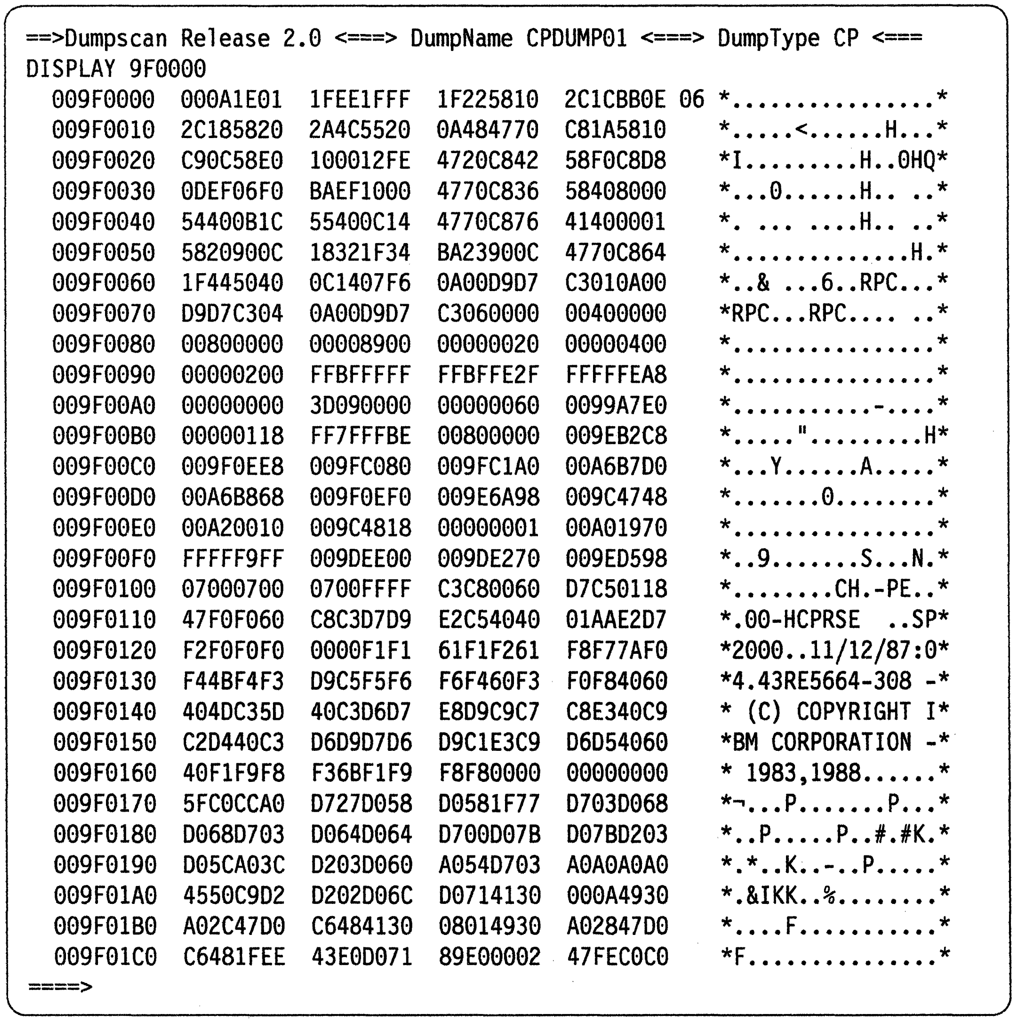
8. If you specify the OFFSET keyword operand, the leftmost column of the output contains the hexadecimal offset from the starting address instead of the 31-bit storage address.

9. If only partial data is available in the dump, DISPLAY presents all of the available data, then an error message is issued.

Sample Output:

Figure 44 illustrates the results of the command DISPLAY 9F0000. Notice that the left-most column contains the storage address of the data. The subcommand entered is:

```
display 9f0000
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
DISPLAY 9F0000
  009F0000  000A1E01  1FEE1FFF  1F225810  2C1CBB0E 06 *................*
  009F0010  2C185820  2A4C5520  0A484770  C81A5810    *.....<......H...*
  009F0020  C90C58E0  100012FE  4720C842  58F0C8D8    *I.........H..0HQ*
  009F0030  0DEF06F0  BAEF1000  4770C836  58408000    *...0......H.. ..*
  009F0040  54400B1C  55400C14  4770C876  41400001    *. ... ....H.. ..*
  009F0050  5820900C  18321F34  BA23900C  4770C864    *..............H.*
  009F0060  1F445040  0C1407F6  0A00D9D7  C3010A00    *..& ...6..RPC...*
  009F0070  D9D7C304  0A00D9D7  C3060000  00400000    *RPC...RPC.... ..*
  009F0080  00800000  00008900  00000020  00000400    *................*
  009F0090  00000200  FFBFFFFF  FFBFFE2F  FFFFFEA8    *................*
  009F00A0  00000000  3D090000  00000060  0099A7E0    *...........-....*
  009F00B0  00000118  FF7FFFBE  00800000  009EB2C8    *.....".........H*
  009F00C0  009F0EE8  009FC080  009FC1A0  00A6B7D0    *...Y......A.....*
  009F00D0  00A6B868  009F0EF0  009E6A98  009C4748    *.......0........*
  009F00E0  00A20010  009C4818  00000001  00A01970    *................*
  009F00F0  FFFFF9FF  009DEE00  009DE270  009ED598    *..9.......S...N.*
  009F0100  07000700  0700FFFF  C3C80060  D7C50118    *........CH.-PE..*
  009F0110  47F0F060  C8C3D7D9  E2C54040  01AAE2D7    *.00-HCPRSE  ..SP*
  009F0120  F2F0F0F0  0000F1F1  61F1F261  F8F77AF0    *2000..11/12/87:0*
  009F0130  F44BF4F3  D9C5F5F6  F6F460F3  F0F84060    *4.43RE5664-308 -*
  009F0140  404DC35D  40C3D6D7  E8D9C9C7  C8E340C9    * (C) COPYRIGHT I*
  009F0150  C2D440C3  D6D9D7D6  D9C1E3C9  D6D54060    *BM CORPORATION -*
  009F0160  40F1F9F8  F36BF1F9  F8F80000  00000000    * 1983,1988......*
  009F0170  5FC0CCA0  D727D058  D0581F77  D703D068    *¬...P.......P...*
  009F0180  D068D703  D064D064  D700D07B  D07BD203    *..P.....P..#.#K.*
  009F0190  D05CA03C  D203D060  A054D703  A0A0A0A0    *.*..K..-..P.....*
  009F01A0  4550C9D2  D202D06C  D0714130  000A4930    *.&IKK..%........*
  009F01B0  A02C47D0  C6484130  08014930  A02847D0    *....F...........*
  009F01C0  C6481FEE  43E0D071  89E00002  47FEC0C0    *F...............*
====>
```

Figure 44. Sample Output of a DISPLAY Subcommand without the Length Operand

Figure 45 illustrates the display generated using 31-bit indirect addressing with a length. The subcommand entered is:

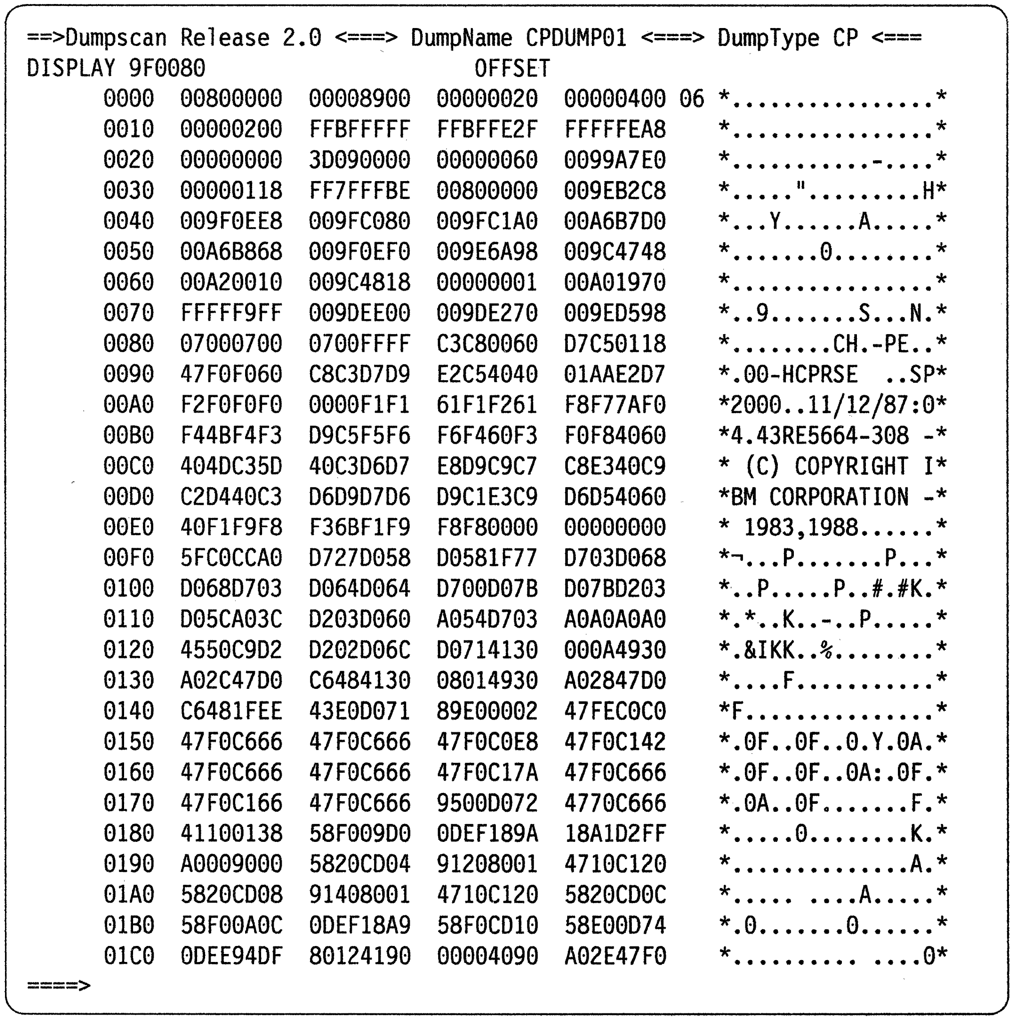
```
display 9f0080? 40
```

```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
DISPLAY 9F0080  ?  40   (00800000)
  00800000  80000020  80000020  80000020  80000020 06 *................*
  00800010  80000020  80000020  80000020  80000020    *................*
  00800020  80000020  80000020  80000020  80000020    *................*
  00800030  80000020  80000020  80000020  80000020    *................*
====>
```

Figure 45. Sample Output of a DISPLAY Subcommand with 31-Bit Indirect Addressing

Figure 46 illustrates the DISPLAY command using the OFFSET operand. Note that the left-hand column is the offset from the address entered. The subcommand entered is:

```
d 9f0080? offset
```

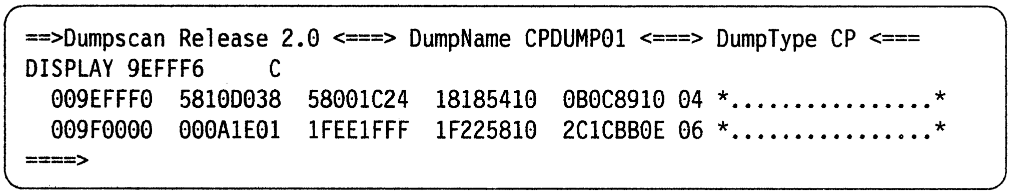


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
DISPLAY 9F0080                     OFFSET
      0000  00800000  00008900  00000020  00000400 06 *................*
      0010  00000200  FFBFFFFF  FFBFFE2F  FFFFFEA8    *................*
      0020  00000000  3D090000  00000060  0099A7E0    *...........-....*
      0030  00000118  FF7FFFBE  00800000  009EB2C8    *....."........H*
      0040  009F0EE8  009FC080  009FC1A0  00A6B7D0    *...Y......A.....*
      0050  00A6B868  009F0EF0  009E6A98  009C4748    *.......0........*
      0060  00A20010  009C4818  00000001  00A01970    *................*
      0070  FFFFF9FF  009DEE00  009DE270  009ED598    *..9.......S...N.*
      0080  07000700  0700FFFF  C3C80060  D7C50118    *........CH.-PE..*
      0090  47F0F060  C8C3D7D9  E2C54040  01AAE2D7    *.00-HCPRSE  ..SP*
      00A0  F2F0F0F0  0000F1F1  61F1F261  F8F77AF0    *2000..11/12/87:0*
      00B0  F44BF4F3  D9C5F5F6  F6F460F3  F0F84060    *4.43RE5664-308 -*
      00C0  404DC35D  40C3D6D7  E8D9C9C7  C8E340C9    * (C) COPYRIGHT I*
      00D0  C2D440C3  D6D9D7D6  D9C1E3C9  D6D54060    *BM CORPORATION -*
      00E0  40F1F9F8  F36BF1F9  F8F80000  00000000    * 1983,1988......*
      00F0  5FC0CCA0  D727D058  D0581F77  D703D068    *¬...P.......P...*
      0100  D068D703  D064D064  D700D07B  D07BD203    *..P.....P..#.#K.*
      0110  D05CA03C  D203D060  A054D703  A0A0A0A0    *.*..K..-..P.....*
      0120  4550C9D2  D202D06C  D0714130  000A4930    *.&IKK..%........*
      0130  A02C47D0  C6484130  08014930  A02847D0    *....F...........*
      0140  C6481FEE  43E0D071  89E00002  47FEC0C0    *F...............*
      0150  47F0C666  47F0C666  47F0C0E8  47F0C142    *.0F..0F..0.Y.0A.*
      0160  47F0C666  47F0C666  47F0C17A  47F0C666    *.0F..0F..0A:.0F.*
      0170  47F0C166  47F0C666  9500D072  4770C666    *.0A..0F.......F.*
      0180  41100138  58F009D0  0DEF189A  18A1D2FF    *.....0........K.*
      0190  A0009000  5820CD04  91208001  4710C120    *..............A.*
      01A0  5820CD08  91408001  4710C120  5820CD0C    *..... ....A.....*
      01B0  58F00A0C  0DEF18A9  58F0CD10  58E00D74    *.0.......0......*
      01C0  0DEE94DF  80124190  00004090  A02E47F0    *.......... ....0*
====>
```

Figure 46. Sample Output of a DISPLAY Subcommand with the OFFSET Operand

Figure 47 illustrates the DISPLAY subcommand specified with a length. Note that the data displayed begins and ends on a hex 10-byte boundary. The subcommand entered is:

```
display 9efff6 c
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
DISPLAY 9EFFF6     C
  009EFFF0  5810D038  58001C24  18185410  0B0C8910 04 *................*
  009F0000  000A1E01  1FEE1FFF  1F225810  2C1CBB0E 06 *................*
====>
```

Figure 47. Sample Output of a DISPLAY Subcommand with a Length Specified

DUMPID Subcommand

Use the DUMPID subcommand to display the dump identifier you assigned on a VMDUMP command. See *VM/XA SP: CP Command Reference* for more information on the VMDUMP command.

The format of the DUMPID subcommand is as follows:

DUMPID	

Usage Note:

The DUMPID subcommand is not supported for CP dumps.

END Subcommand

Use the END subcommand to end the session.

The format of the END subcommand is as follows:

END	

Usage Notes:

1. The END subcommand is equivalent to the XEDIT PQUIT subcommand.
2. If you enter the END subcommand and you have changed the session file, you must enter the XEDIT FILE subcommand to save the session file. If you do not wish to save the session file, enter the DUMPSCAN QUIT or DUMPSCAN HX subcommand.

FINDMOD Subcommand

Use this subcommand either to locate a specified module or entry point in the dump or to locate the module and the entry point that resides at a specified address.

The format of the FINDMOD subcommand is as follows:

FINDMod	{ *module name* *entry point name* *address* }

Where:

module name
is an alphanumeric string of 1-to-8 characters that specifies a module name. Place this variable within single quotes.

entry point name
is an alphanumeric string of 1-to-8 characters that specifies an entry point name. Place this variable within single quotes.

address
is a 31-bit (4-byte) hexadecimal address.

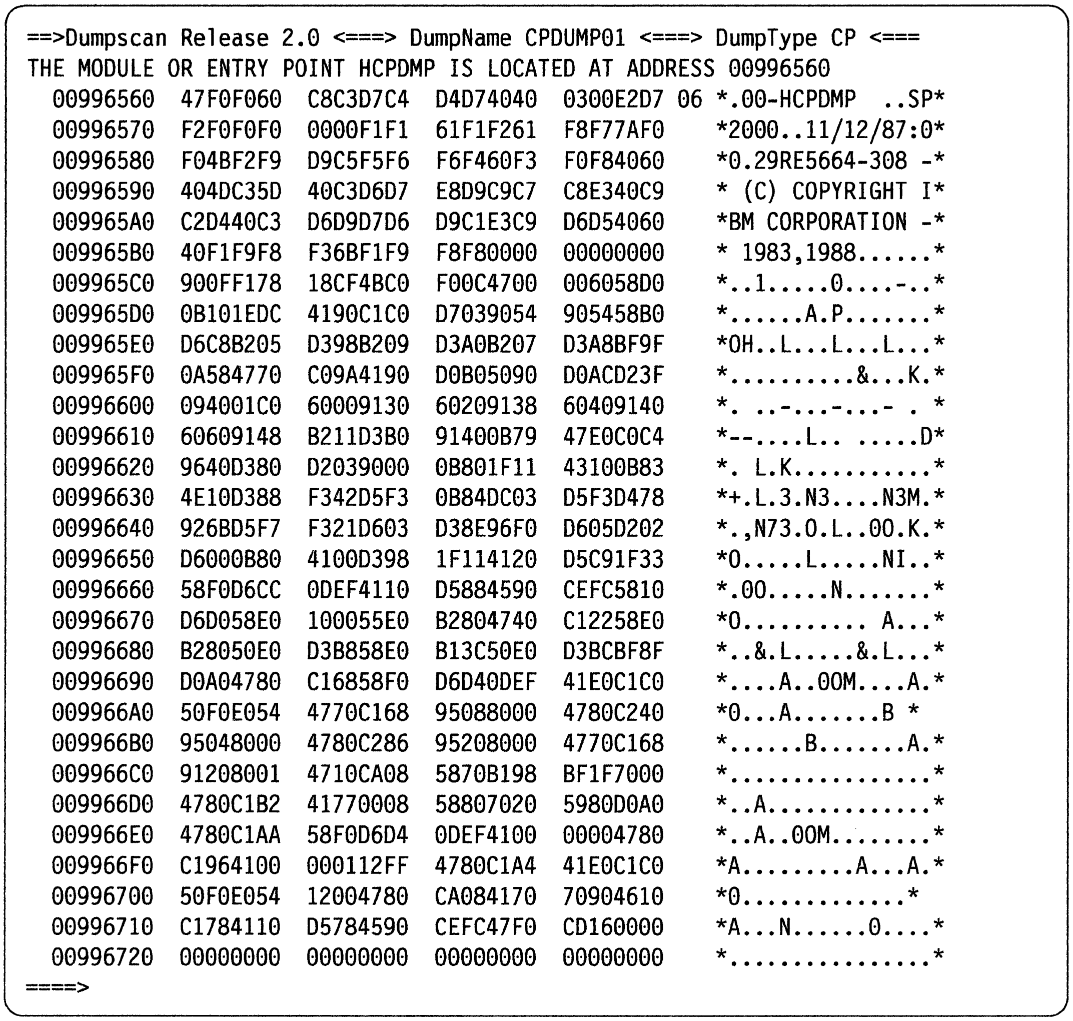
Usage Notes:

1. The FINDMOD subcommand requires that a module map of the dump be appended to the dump. For CP hard abend and stand-alone dumps only, if a load map is not currently appended to the dump, one is built dynamically and appended. However, to utilize the dynamic map building capabilities, the dump file must be on a CMS disk to which you have write access.
2. If the module name or entry point name operands are entered, the starting address of the module or entry point is displayed on the first line of the screen.
3. If the requested module is not in storage, an error message is issued.
4. A string in quotes is processed as a module or entry point name. A string not in quotes is handled first as an address. If the string is not a valid hexadecimal number, it is then processed as a module or entry point name. If the string is not a module and is not a valid hexadecimal address, an error message is issued indicating the string is not a valid hexadecimal address.
5. The output displays the name and hexadecimal location of the next lowest entry point and the displacement of the address from that entry point address, as well as the name of the module containing that entry point and the displacement from its start.
6. Scrolling subcommands (BACKWARD, FORWARD, + or increment, - or decrement) can be used once the module is found.
7. If the specified address is not in the dump, an error message is displayed.
8. If the operand entered is either a module name or an entry point name and it exits in the dump, the screen is cleared before the information is displayed.
9. If the operand entered is an address, the screen is not cleared prior to display of the information.
10. MAPA and MAPN are functionally equivalent to the FINDMOD subcommand. These are VM/SP Interactive Problem Control System (IPCS) subcommands.

Sample Output:

Figure 48 illustrates the data displayed when the FINDMOD subcommand is used to locate module HCPDMP. The subcommand entered is:

```
findmod  'hcpdmp'
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
THE MODULE OR ENTRY POINT HCPDMP IS LOCATED AT ADDRESS 00996560
  00996560  47F0F060  C8C3D7C4  D4D74040  0300E2D7 06 *.00-HCPDMP  ..SP*
  00996570  F2F0F0F0  0000F1F1  61F1F261  F8F77AF0    *2000..11/12/87:0*
  00996580  F04BF2F9  D9C5F5F6  F6F460F3  F0F84060    *0.29RE5664-308 -*
  00996590  404DC35D  40C3D6D7  E8D9C9C7  C8E340C9    * (C) COPYRIGHT I*
  009965A0  C2D440C3  D6D9D7D6  D9C1E3C9  D6D54060    *BM CORPORATION -*
  009965B0  40F1F9F8  F36BF1F9  F8F80000  00000000    * 1983,1988......*
  009965C0  900FF178  18CF4BC0  F00C4700  006058D0    *..1.....0....-..*
  009965D0  0B101EDC  4190C1C0  D7039054  905458B0    *......A.P.......*
  009965E0  D6C8B205  D398B209  D3A0B207  D3A8BF9F    *OH..L...L...L...*
  009965F0  0A584770  C09A4190  D0B05090  D0ACD23F    *..........&...K.*
  00996600  094001C0  60009130  60209138  60409140    *. ..-...-...- . *
  00996610  60609148  B211D3B0  91400B79  47E0C0C4    *--....L.. .....D*
  00996620  9640D380  D2039000  0B801F11  43100B83    *. L.K...........*
  00996630  4E10D388  F342D5F3  0B84DC03  D5F3D478    *+.L.3.N3....N3M.*
  00996640  926BD5F7  F321D603  D38E96F0  D605D202    *.,N73.O.L..0O.K.*
  00996650  D6000B80  4100D398  1F114120  D5C91F33    *O.....L.....NI..*
  00996660  58F0D6CC  0DEF4110  D5884590  CEFC5810    *.0O.....N.......*
  00996670  D6D058E0  100055E0  B2804740  C12258E0    *O.......... A...*
  00996680  B28050E0  D3B858E0  B13C50E0  D3BCBF8F    *..&.L.....&.L...*
  00996690  D0A04780  C16858F0  D6D40DEF  41E0C1C0    *....A..0OM....A.*
  009966A0  50F0E054  4770C168  95088000  4780C240    *0...A.......B *
  009966B0  95048000  4780C286  95208000  4770C168    *......B.......A.*
  009966C0  91208001  4710CA08  5870B198  BF1F7000    *................*
  009966D0  4780C1B2  41770008  58807020  5980D0A0    *..A.............*
  009966E0  4780C1AA  58F0D6D4  0DEF4100  00004780    *..A..0OM........*
  009966F0  C1964100  000112FF  4780C1A4  41E0C1C0    *A.........A...A.*
  00996700  50F0E054  12004780  CA084170  70904610    *0.............*
  00996710  C1784110  D5784590  CEFC47F0  CD160000    *A...N......0....*
  00996720  00000000  00000000  00000000  00000000    *................*
====>
```

Figure 48. FINDMOD Output When a Module Name Is Specified

Figure 49 illustrates the data displayed when an address operand (996770) is specified to determine the module that has the address. The subcommand entered is:

```
findmod 996770
```

```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
   00996770 IS 210 BYTES INTO MODULE HCPDMP AT 00996560
   00996770 IS 1B0 BYTES INTO ENTRY HCPDMPDK AT 009965C0
====>
```

Figure 49. FINDMOD Output When an Address Is Specified

FINDUSER Subcommand

Use the FINDUSER subcommand (macro) to determine the CP module flow for a hung virtual machine (if the save areas can be successfully located and chained). You can also use it to provide key status settings from the virtual machine's VMDBK and display each active save area. FINDUSER helps you find active save areas and VMDBK status settings for a specified userid. The output can also be put in a separate CMS file.

FINDUSER	*userid* [*FILE*]

Where:

userid
is the 1- to 8-character userid you want to locate.

FILE
is an optional parameter to specify that output will be placed in a separate CMS file called *userid dumpname* in addition to the output being appended to the session file.

Usage Notes:

1. This macro is fully supported for a CP hard abend or stand-alone type dump only. Partial output (VMDBK status and SAVBK) is displayed for soft abend dumps.
2. If only partial data is available in the dump (as in soft abend or virtual machine dumps), FINDUSER may return limited output.
3. If the FILE option is not specified on input, FINDUSER output will be appended to the session file.
4. The following error messages may be issued by FINDUSER. The action you should take appears below each message.
   a. COMMAND FORMAT IS: FINDUSER userid < FILE >

      Reissue the macro with the correct syntax.

   b. THE VMDBK FOR userid DOES NOT EXIST IN THE DUMP

      Reissue the macro with a valid userid. The DUMPSCAN subcommand VMDBK LIST provides all valid userids.

   c. UNABLE TO FIND START OF SAVE AREAS

      The macro cannot locate the save area queue (HCPRCCCM).

   d. NO ACTIVE SAVE AREAS FOUND FOR USERID - userid

      The macro cannot find any valid save area information for the specified userid. The output will contain VMDBK status settings.

   e. UNEXPECTED RC rc AFTER ISSUING: FINDSTRG fsstring fsstart fsstop fsincr

      FINDUSER initiated a FINDSTRG subcommand that was not successful. Based on the return code, try to determine if the macro was at fault (RC 16) or the subcommand failed.

f. UNEXPECTED RC rc AFTER ISSUING: READSTRG rsadr rslen

FINDUSER initiated a READSTRG subcommand that was not successful. Based on the return code, try to determine if the macro was at fault (RC 16) or the subcommand failed.

5. The following messages can appear in the output produced by FINDUSER:

a. NO ACTIVE SAVE AREAS FOUND FOR USERID - userid

Valid save area information for the specified userid cannot be found. The output will contain VMDBK status settings.

b. INTERSECTING/CONTAMINATED SAVE AREA CHAINING. SAVE AREAS BELOW ARE IN NO SPECIFIC ORDER.

The active save areas do not chain correctly. Each save area as well as the VMDBK status settings will appear in output.

c. BEGINNING OF SAVE AREA CAN NOT BE DETERMINED. SAVE AREAS BELOW ARE IN NO SPECIFIC ORDER.

The active save areas appear to loop. The output will display each save area as well as the VMDBK status settings.

d. MULTIPLE SAVE AREA CHAINS FOUND FOR USERID - userid

The active save areas form multiple chains. The output will contain module flow for each chain and for VMDBK status settings and will display each save area.

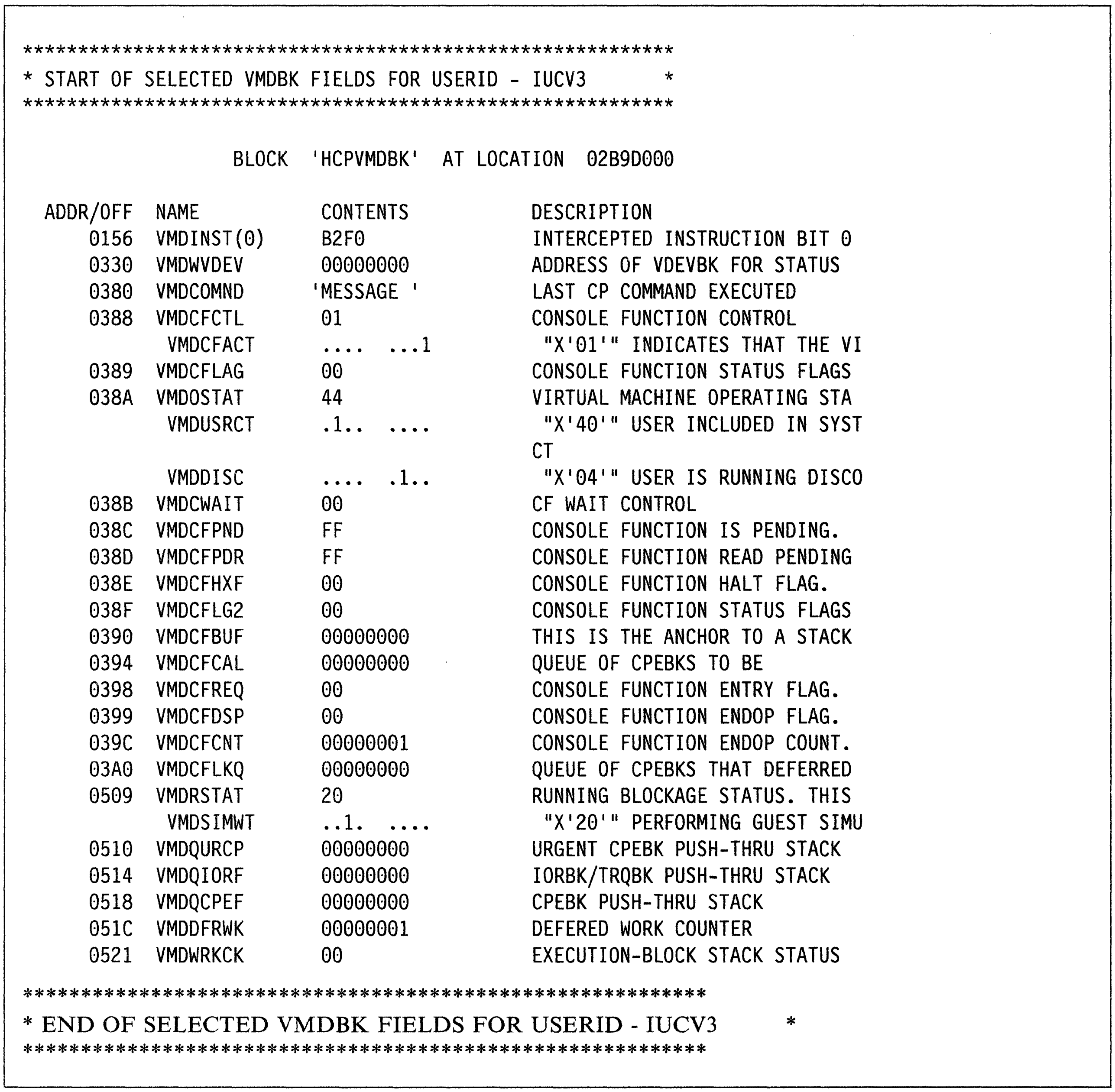
Sample Output:

Once in the DUMPSCAN environment, the FINDUSER macro can be invoked by typing FINDUSER userid (FILE) on the command line. An example of the DUMPSCAN macro FINDUSER follows. The following represents contiguous output. The subcommand entered is:

```
finduser iucv3 file
```

```
FINDUSER IUCV3 FILE (DUMPNAME CPDUMP01)

VMDBK AT 02B9D000 FOR USERID - IUCV3

**********************************************
* START OF MODULE CHAIN FOR USERID - IUCV3    *
**********************************************
   022B6F2A IS 2DA BYTES INTO MODULE HCPVIX AT 022B6C50
   022B6F2A IS 27A BYTES INTO ENTRY HCPVIXVI AT 022B6CB0
   02259E86 IS 6C6 BYTES INTO MODULE HCPIUF AT 022597C0
   02259E86 IS 26 BYTES INTO ENTRY HCPIUFRF AT 02259E60
   0225D232 IS 1D2 BYTES INTO MODULE HCPLCK AT 0225D060
   0225D232 IS 11A BYTES INTO ENTRY HCPLCKAX AT 0225D118

**********************************************
* END OF MODULE CHAIN FOR USERID - IUCV3      *
**********************************************
```

Figure 50. FINDUSER Output (1)



```
**********************************************************
* START OF SELECTED VMDBK FIELDS FOR USERID - IUCV3      *
**********************************************************

                   BLOCK  'HCPVMDBK'  AT LOCATION  02B9D000

  ADDR/OFF  NAME           CONTENTS           DESCRIPTION
      0156  VMDINST(0)     B2F0               INTERCEPTED INSTRUCTION BIT 0
      0330  VMDWVDEV       00000000           ADDRESS OF VDEVBK FOR STATUS
      0380  VMDCOMND      'MESSAGE '          LAST CP COMMAND EXECUTED
      0388  VMDCFCTL       01                 CONSOLE FUNCTION CONTROL
             VMDCFACT      ....  ...1          "X'01'" INDICATES THAT THE VI
      0389  VMDCFLAG       00                 CONSOLE FUNCTION STATUS FLAGS
      038A  VMDOSTAT       44                 VIRTUAL MACHINE OPERATING STA
             VMDUSRCT      .1..  ....          "X'40'" USER INCLUDED IN SYST
                                              CT
             VMDDISC       ....  .1..          "X'04'" USER IS RUNNING DISCO
      038B  VMDCWAIT       00                 CF WAIT CONTROL
      038C  VMDCFPND       FF                 CONSOLE FUNCTION IS PENDING.
      038D  VMDCFPDR       FF                 CONSOLE FUNCTION READ PENDING
      038E  VMDCFHXF       00                 CONSOLE FUNCTION HALT FLAG.
      038F  VMDCFLG2       00                 CONSOLE FUNCTION STATUS FLAGS
      0390  VMDCFBUF       00000000           THIS IS THE ANCHOR TO A STACK
      0394  VMDCFCAL       00000000           QUEUE OF CPEBKS TO BE
      0398  VMDCFREQ       00                 CONSOLE FUNCTION ENTRY FLAG.
      0399  VMDCFDSP       00                 CONSOLE FUNCTION ENDOP FLAG.
      039C  VMDCFCNT       00000001           CONSOLE FUNCTION ENDOP COUNT.
      03A0  VMDCFLKQ       00000000           QUEUE OF CPEBKS THAT DEFERRED
      0509  VMDRSTAT       20                 RUNNING BLOCKAGE STATUS. THIS
             VMDSIMWT      ..1.  ....          "X'20'" PERFORMING GUEST SIMU
      0510  VMDQURCP       00000000           URGENT CPEBK PUSH-THRU STACK
      0514  VMDQIORF       00000000           IORBK/TRQBK PUSH-THRU STACK
      0518  VMDQCPEF       00000000           CPEBK PUSH-THRU STACK
      051C  VMDDFRWK       00000001           DEFERED WORK COUNTER
      0521  VMDWRKCK       00                 EXECUTION-BLOCK STACK STATUS

**********************************************************
* END OF SELECTED VMDBK FIELDS FOR USERID - IUCV3        *
**********************************************************
```

Figure 51. FINDUSER Output (2)

Note: In the preceding figure, output produced by the BLOCK subcommand was modified by the EXEC. Specifically, only bits that were set on were displayed for the fields VMDOSTAT, VMDRSTAT, and VMDCFCTL.

```
**************************************************
* START OF SAVE AREA CHAIN FOR USERID - IUCV3    *
**************************************************
DISPLAY 02387D80   80              OFFSET
      0000  00000000  00000000  00000000  0388A080 06 *................*
      0010  00800000  0229500A  FF1E4000  0228F940    *......&... ...9 *
      0020  FF1E4000  0228F940  02259E60  010000E2    *.. ...9 ...-...S*
      0030  00000000  02259E60  02BE1148  00000000    *.......-........*
      0040  02B9D000  02B9D000  022B6C50  03EB0D00    *..........%&....*
      0050  822B6F2A  02259E60  02B9D828  00000000    *..?....-..Q.....*
      0060  00000000  00000000  00000000  00000000    *................*
      0070  00000000  00000000  00000000  00000000    *................*
DISPLAY 03EB0B00   80              OFFSET
      0000  00000000  00000000  00000000  0388A080 06 *................*
      0010  80400000  02294D70  FF1E4000  02B9D828    *. ....(... ...Q.*
      0020  02B9D828  0228F940  02259E60  010000E2    *..Q...9 ...-...S*
      0030  00000000  02259E60  02BE1148  00000000    *.......-........*
      0040  02B9D000  02B9D000  022597C0  02387D80    *..............'.*
      0050  82259E86  0225D232  00000000  02B9D000    *......K.........*
      0060  00000000  00000000  00000000  00000000    *................*
      0070  00000000  00000000  00000000  00000000    *................*
**************************************************
* END OF SAVE AREA CHAIN FOR USERID - IUCV3      *
**************************************************
```

Figure 52. FINDUSER Output (3)

FORWARD Subcommand

The FORWARD subcommand scrolls forward toward the highest address in the dump.

The format of the FORWARD subcommand is as follows:

Forward	

Usage Notes:

1. The FORWARD subcommand can be used after issuing the DISPLAY, LOCATE, FINDMOD, TRACE, or other scrolling subcommands.
2. The FORWARD subcommand may be reissued by pressing the ENTER key ("null line" subcommand).
3. The SCROLL subcommand is functionally equivalent to the FORWARD subcommand. SCROLL is a VM/SP Interactive Problem Control System (IPCS) subcommand.
4. If the OFFSET operand was specified on the previous DISPLAY subcommand, then issuing the FORWARD subcommand continues to display offsets.
5. You cannot display with offsets beyond hex FFF0.
6. The FORWARD subcommand does not wrap the screen.
7. Refer to the BACKWARD subcommand to scroll backward toward the lowest address in the dump.

Responses:

One full screen of the dump data is presented in both hexadecimal and EBCDIC.

When scrolling after the TRACE subcommand, the format of the next screen is identical to the format of the screen when TRACE was entered. For example, if the previous TRACE subcommand was for FORMAT output, scrolling continues with formatted output.

FRAMETBL Subcommand

Use the FRAMETBL subcommand to format and display the contents of a frame table entry for the frame containing the hexadecimal location specified. The frame table entry address may also be specified.

The format of the FRAMETBL subcommand is as follows:

FRAMEtbl **FRMtbl**	*storage address* **ENTRY** entry *address*

Where:

storage address
is a 31-bit (4-byte) hexadecimal number specifying the address whose entry in the frame table is to be described and formatted.

ENTRY
specifies that an actual frame table entry address will be given.

entry address
is a 31-bit (4-byte) hexadecimal number specifying the address of a frame table entry.

Usage Notes:

1. The FRAMETBL subcommand is not supported for virtual machine dumps or soft abend dumps.
2. The specified storage address is rounded down, if necessary, to the 4K page address. The formatted contents of the frame table entry for this page are displayed.
3. The frame entry address must be on a 16-byte boundary.
4. FRMtbl is a synonym for the FRAMEtbl subcommand.
5. If you specify an entry address outside the boundaries of the frame table but within the dump, the data specified is formatted only if it begins at an address on a 16-byte boundary.
6. Table 3 on page 109 describes the possible uses for the real storage frame.

Table 3. Real Storage Frame Use Table

Code	Frame Use
FRMOFFLN	Offline
FRMVR	V=R Guest Storage
FRMFRVR	V=R Free Storage
FRMFREE	Free Storage
FRMFRSY	System Extended Free Storage
FRMFRVM	VMDBK Extended Free Storage
FRMCP	CP Use
FRMTRACE	Trace Table
FRMPRFX	Prefix Page
FRMUSER	User xxxxxxxx
FRMSUSER	System Virtual Storage
FRMR370	Real I/O 370 Mode.

Responses:

The format of an entry in the frame table depends on the use of the page. Therefore, output from the FRAMETBL subcommand varies according to the use of the page being displayed. There are three possible responses from this subcommand. These are shown below as sample output. In each case, the system does not clear the screen before displaying the output from the FRAMETBL subcommand.

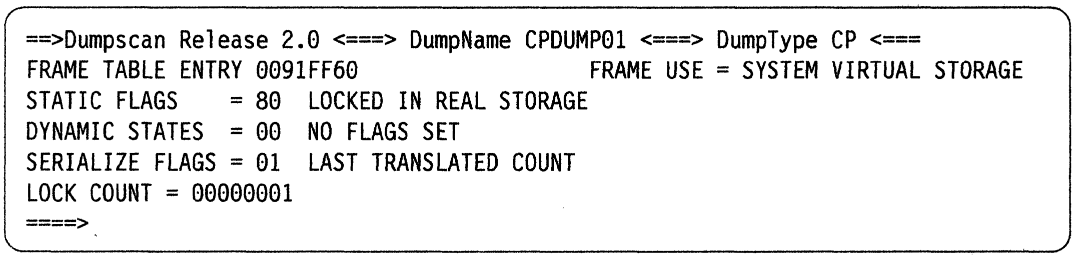
In all cases, the static, serialize flags, and dynamic states are defined, and the first line of output contains:

- The address of the frame table entry
- A description of the use of the real storage frame.

Sample Output:

Figure 53 through Figure 55 illustrate various output screens generated by the FRAMEtbl subcommand. For Figure 53, the subcommand entered is:

```
frm entry 91ff60
```

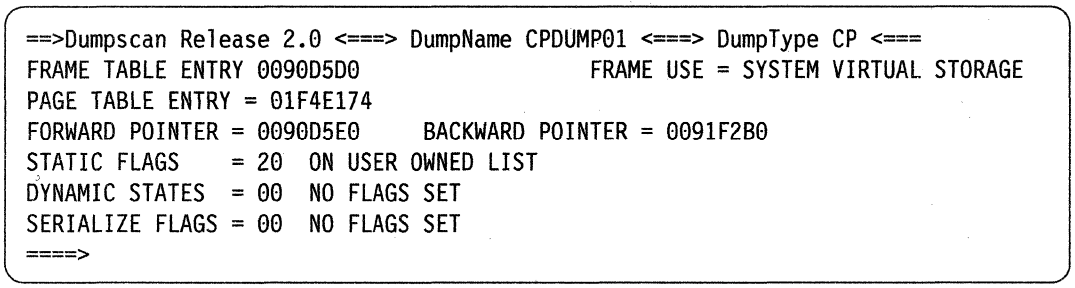


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
FRAME TABLE ENTRY 0091FF60                  FRAME USE = SYSTEM VIRTUAL STORAGE
STATIC FLAGS     = 80  LOCKED IN REAL STORAGE
DYNAMIC STATES   = 00  NO FLAGS SET
SERIALIZE FLAGS = 01  LAST TRANSLATED COUNT
LOCK COUNT = 00000001
====>
```

Figure 53. Sample Output of a FRAMETBL Subcommand for a Page Frame Locked in Real Storage

For Figure 54, the subcommand entered is:

```
frm entry 90d5d0
```

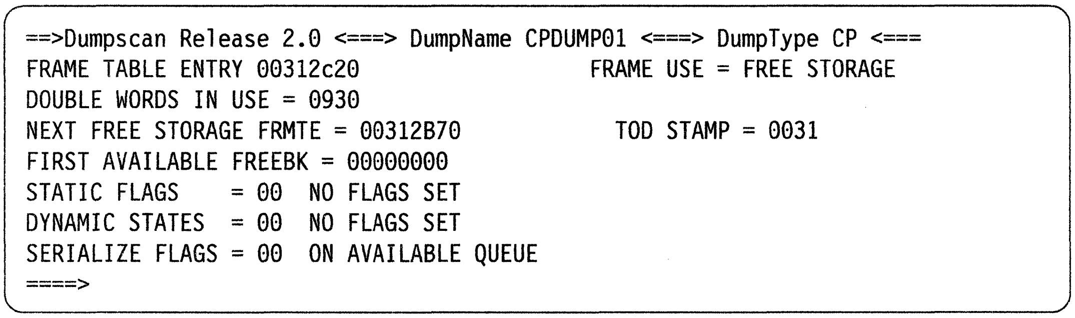


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
FRAME TABLE ENTRY 0090D5D0                  FRAME USE = SYSTEM VIRTUAL STORAGE
PAGE TABLE ENTRY = 01F4E174
FORWARD POINTER = 0090D5E0     BACKWARD POINTER = 0091F2B0
STATIC FLAGS     = 20  ON USER OWNED LIST
DYNAMIC STATES   = 00  NO FLAGS SET
SERIALIZE FLAGS = 00  NO FLAGS SET
====>
```

Figure 54. Sample Output of a FRAMETBL Subcommand for Pages That Are Chained

For Figure 55, the subcommand entered is:

```
frm entry 312c20
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
FRAME TABLE ENTRY 00312c20                  FRAME USE = FREE STORAGE
DOUBLE WORDS IN USE = 0930
NEXT FREE STORAGE FRMTE = 00312B70            TOD STAMP = 0031
FIRST AVAILABLE FREEBK = 00000000
STATIC FLAGS     = 00  NO FLAGS SET
DYNAMIC STATES   = 00  NO FLAGS SET
SERIALIZE FLAGS = 00  ON AVAILABLE QUEUE
====>
```

Figure 55. Sample Output of a FRAMETBL Subcommand for a Page Frame Used as System Free Storage

GREGS Subcommand

Use the GREGS subcommand to display general purpose registers for a specified processor.

The format of the GREGS subcommand is as follows:

Gregs	[*cpuaddr*]

Where:

cpuaddr
is a 1- to 4-digit hexadecimal number specifying the physical CPU address for which the general registers are to be displayed.

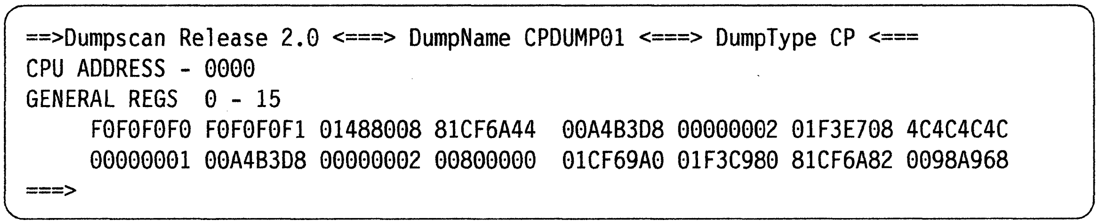
Usage Notes:

1. If the *cpuaddr* operand is not specified, it defaults to the failing processor for a CP abend dump, the IPL processor for a stand-alone dump, or the processor on which the CP VMDUMP command was issued for the virtual machine.
2. Use the CPU subcommand to obtain the CPU addresses in the dump.

Sample Output:

Figure 56 illustrates the output of the GREGS subcommand for CPU address 0000. The subcommand entered is:

```
gregs
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
CPU ADDRESS - 0000
GENERAL REGS  0 - 15
     F0F0F0F0 F0F0F0F1 01488008 81CF6A44  00A4B3D8 00000002 01F3E708 4C4C4C4C
     00000001 00A4B3D8 00000002 00800000  01CF69A0 01F3C980 81CF6A82 0098A968
===>
```

Figure 56. Sample Output of a GREGS Subcommand

HELP Subcommand

Use the HELP subcommand to display a summary of the DUMPSCAN subcommands. The HELP subcommand will describe the specified subcommand, including syntax, operands, and any relevant usage notes. If you do not enter a subcommand name, a list of all DUMPSCAN subcommands will be presented.

The format of the HELP subcommand is as follows:

```
HELP        [ MENU       ]
            [ subcommand ]
```

Where:

MENU
displays a list of all DUMPSCAN subcommands.

subcommand
can be any DUMPSCAN subcommand name and causes a description of the subcommand to be displayed. If the subcommand is not a DUMPSCAN subcommand, a list of all DUMPSCAN subcommands will be presented.

Usage Notes:

1. If you do not enter a subcommand name, or if you enter an invalid subcommand, a list of subcommands is displayed.
2. DUMPSCAN subcommand abbreviations can be used in the subcommand operand.

Responses:

The dump viewing facility HELP subcommand provides online information about command syntax, formats, and usage notes. The HELP text displayed is the same as displayed when HELP is invoked from CMS.

To display a menu of DUMPSCAN subcommands, enter:

```
help
```

To display the HELP text of a specific subcommand (for example, for the REGS subcommand) enter:

```
help regs
```

Error Messages

Error messages are issued from the CMS HELP facility.

Return Codes:

RC	Explanation
0	Normal.
6	Subcommand rejected in the profile due to a LOAD error, or a QUIT subcommand has been issued in a macro called from the last file in the ring.
11 and above	Standard CMS HELP command return codes.

HX Subcommand

Use the HX subcommand to end the session and return to CMS. The format of the HX subcommand is as follows:

HX	

Usage Note:

The HX subcommand is equivalent to the QUIT subcommand.

Response:

CMS ready message.

LOCATE(UP) Subcommand

Use the LOCATE subcommand to search the dump for a particular string of data.

The format of the LOCATE subcommand is as follows:

Locate **LocateUp**	*string* *X'string'*	*fromaddr*	*toaddr*	[*increment* <u>1</u>]

Where:

string
is 1- to 8-EBCDIC characters to be searched for.

X'string'
is a 1- to 16-digit hexadecimal string to be searched for. The string must be in quotes and preceded by the letter X.

fromaddr
is the 31-bit (4-byte) hexadecimal starting address for the search.

toaddr
is the 31-bit (4-byte) hexadecimal number that is the ending address for the search.

increment
is a 1- to 4-digit hexadecimal number to change the current address after each match attempt.

1
is the default increment if none is specified.

Usage Notes:

1. All EBCDIC strings are truncated on the right to 8 characters. All hexadecimal strings are truncated on the right to 16 hexadecimal digits.

2. The second quote of the *X'string'* operand is optional.

3. The LOCATE subcommand may be reissued by pressing the ENTER key ("null line" subcommand). The from address (*fromaddr*) is updated using the current address and the increment, and the subcommand is then reissued.

4. If the following conditions are true, use of the increment operand in the LOCATE subcommand can reduce search time by eliminating unwanted matches.

   - If the target string is at a fixed displacement in each entry of a data area, and each entry has a fixed length

   - If the target string is at a fixed boundary (fullword, doubleword, 16-byte, 32-byte, and so on).

   For example, to check the beginning of each hex 20-byte entry from address hex 4000 to hex 8000 for the character string ABCD, enter:

   ```
   locate abcd 4000 8000 20
   ```

   The data at the hex addresses 4000, 4020, 4040, ... 8000 is searched for the string ABCD until the first occurrence (if any) is reached. These addresses are the increment length (hex 20) apart.

5. The valid increment range is from hex 1 to 1000.
6. If the LOCATE subcommand is placed in the &name table, the maximum string of 8 characters includes the hexadecimal identifier X with the hexadecimal characters with quotes (for example, X'13AB4').
7. If LOCATE is specified, the starting address must be less than the ending address; otherwise, an error message is issued.
8. If LOCATEUP is specified, the starting address must be greater than the ending address. If it is not, an error message is issued.

Response:

Provided the end of dump has not been reached, one full screen of data is presented, in both hexadecimal and EBCDIC. The target string is positioned on the first line at the hexadecimal location where the string begins.

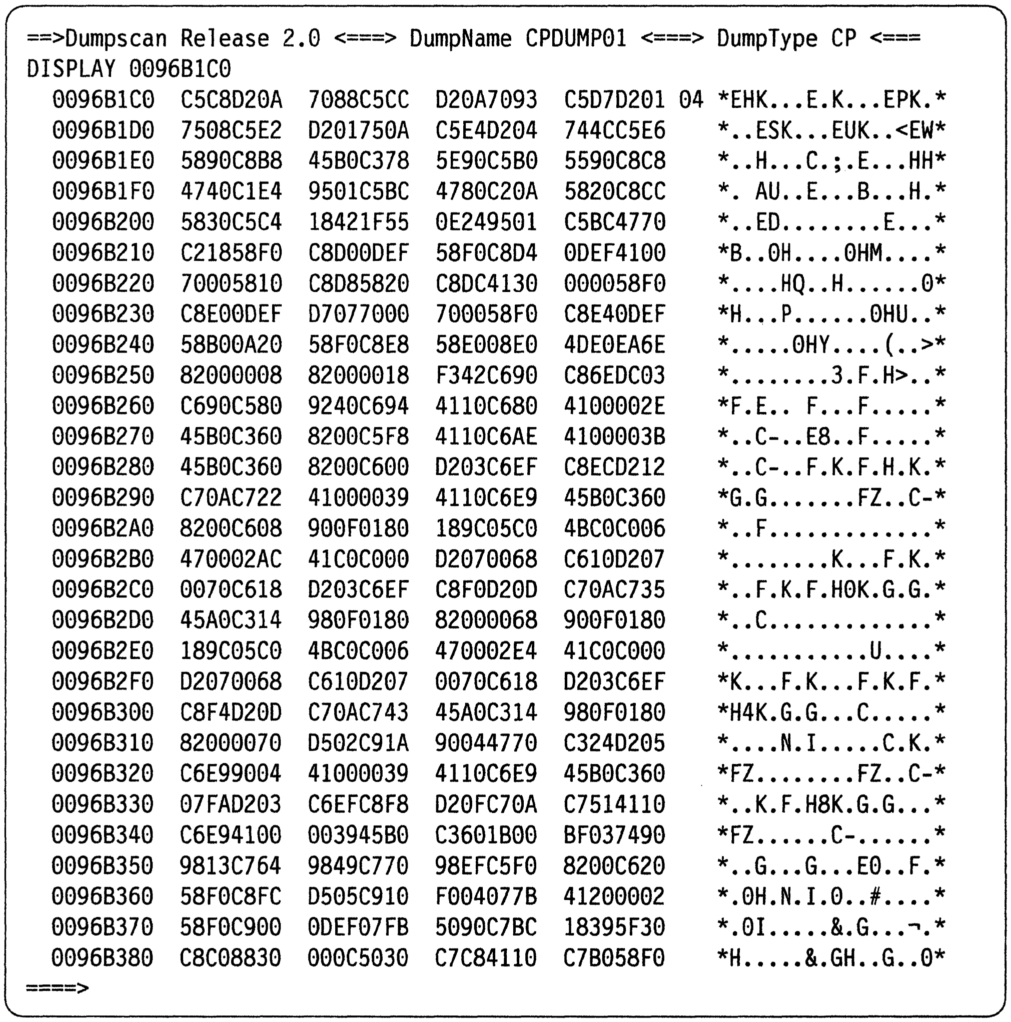
Sample Output:

Figure 57 illustrates the screen displayed when the subcommand LOCATE X'C5C8D20A' 900000 990000 8 is entered. This subcommand searches the dump from hex address 900000 up through address 9900000 searching for the hexadecimal string X'C5C8D20A'. The dump is stepped through by adding 8 bytes from the current address until either a match is found or the to address is reached.

Note that the first occurrence of the string is on the first line of data at address 0096B1C0.

The subcommand entered is:

```
lo x'c5c8d20a' 900000 990000 8
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
DISPLAY 0096B1C0
  0096B1C0  C5C8D20A  7088C5CC  D20A7093  C5D7D201 04 *EHK...E.K...EPK.*
  0096B1D0  7508C5E2  D201750A  C5E4D204  744CC5E6    *..ESK...EUK..<EW*
  0096B1E0  5890C8B8  45B0C378  5E90C5B0  5590C8C8    *..H...C.;.E...HH*
  0096B1F0  4740C1E4  9501C5BC  4780C20A  5820C8CC    *. AU..E...B...H.*
  0096B200  5830C5C4  18421F55  0E249501  C5BC4770    *..ED........E...*
  0096B210  C21858F0  C8D00DEF  58F0C8D4  0DEF4100    *B..0H....0HM....*
  0096B220  70005810  C8D85820  C8DC4130  000058F0    *....HQ..H......0*
  0096B230  C8E00DEF  D7077000  700058F0  C8E40DEF    *H...P......0HU..*
  0096B240  58B00A20  58F0C8E8  58E008E0  4DE0EA6E    *.....0HY....(..>*
  0096B250  82000008  82000018  F342C690  C86EDC03    *........3.F.H>..*
  0096B260  C690C580  9240C694  4110C680  4100002E    *F.E.. F...F.....*
  0096B270  45B0C360  8200C5F8  4110C6AE  4100003B    *..C-..E8..F.....*
  0096B280  45B0C360  8200C600  D203C6EF  C8ECD212    *..C-..F.K.F.H.K.*
  0096B290  C70AC722  41000039  4110C6E9  45B0C360    *G.G.......FZ..C-*
  0096B2A0  8200C608  900F0180  189C05C0  4BC0C006    *..F.............*
  0096B2B0  470002AC  41C0C000  D2070068  C610D207    *........K...F.K.*
  0096B2C0  0070C618  D203C6EF  C8F0D20D  C70AC735    *..F.K.F.H0K.G.G.*
  0096B2D0  45A0C314  980F0180  82000068  900F0180    *..C.............*
  0096B2E0  189C05C0  4BC0C006  470002E4  41C0C000    *...........U....*
  0096B2F0  D2070068  C610D207  0070C618  D203C6EF    *K...F.K...F.K.F.*
  0096B300  C8F4D20D  C70AC743  45A0C314  980F0180    *H4K.G.G...C.....*
  0096B310  82000070  D502C91A  90044770  C324D205    *....N.I.....C.K.*
  0096B320  C6E99004  41000039  4110C6E9  45B0C360    *FZ........FZ..C-*
  0096B330  07FAD203  C6EFC8F8  D20FC70A  C7514110    *..K.F.H8K.G.G...*
  0096B340  C6E94100  003945B0  C3601B00  BF037490    *FZ......C-......*
  0096B350  9813C764  9849C770  98EFC5F0  8200C620    *..G...G...E0..F.*
  0096B360  58F0C8FC  D505C910  F004077B  41200002    *.0H.N.I.0..#....*
  0096B370  58F0C900  0DEF07FB  5090C7BC  18395F30    *.0I.....&.G...¬.*
  0096B380  C8C08830  000C5030  C7C84110  C7B058F0    *H.....&.GH..G..0*
====>
```

Figure 57. Sample Output of a LOCATE Subcommand with the Increment Operand

PRINT Subcommand

Use the PRINT subcommand to print data displayed on your terminal by one of the DUMPSCAN subcommands. The format of the PRINT subcommand is as follows:

```
Print        ┌              ┐
PRT          │ subcommand   │
             │ ON           │
             │ OFF          │
             │ CLOSE        │
             │ ?            │
             └              ┘
```

Where:

subcommand
is a DUMPSCAN subcommand to be issued. Its results are printed and displayed.

ON
turns on the print switch to collect data for printing.

OFF
turns off the print switch, but does not close the virtual printer.

CLOSE
sends data from the virtual printer to the real printer, but does not turn the print switch off.

?
displays the print switch status (ON or OFF).

Usage Notes:

1. When the print switch is ON, all data displayed at the terminal is also written to the virtual printer.
2. PRINT with no operands reissues the subcommand previously entered and prints the data. The data is not redisplayed at the terminal.
3. The PRINT subcommand is not allowed in the &name table.
4. CLOSE is automatically issued at the end of the DUMPSCAN session.
5. If the print switch is OFF and the print subcommand is issued, with or without an operand, the print switch is turned on for that operation, then turned off. If the print switch was ON, it is left on.
6. PRT is a synonym for the PRINT subcommand.

QUIT Subcommand

Use the QUIT subcommand to end the session and return to CMS. The format of the QUIT subcommand is as follows:

QUIT	

Usage Note:

The QUIT subcommand is equivalent to the HX subcommand.

REGS Subcommand

Use the REGS subcommand to display registers, clocks, timer, and program status words for a specific processor. The format of the REGS subcommand is as follows:

Regs	[*cpuaddr*]

Where:

cpuaddr
is a 1- to 4-digit hexadecimal number specifying the CPU address for which the information is to be displayed.

Usage Notes:

1. For System/370 virtual machine dumps, the output of the REGS subcommand also includes:
   - Channel status word (CSW)
   - Channel address word (CAW)
   - Interval timer
   - Current program status word (PSW).
2. The REGS subcommand clears the screen prior to presenting data.
3. Use the CPU subcommand to obtain the CPU addresses in the dump.
4. If the *cpuaddr* operand is not specified, it defaults to the failing processor for CP dumps that are not stand-alone dumps, the IPL processor for stand-alone dumps, or the processor on which the CP VMDUMP command was issued for the virtual machine.

Sample Output:

Figure 58 illustrates the output of the REGS subcommand for the failing processor. The subcommand entered is:

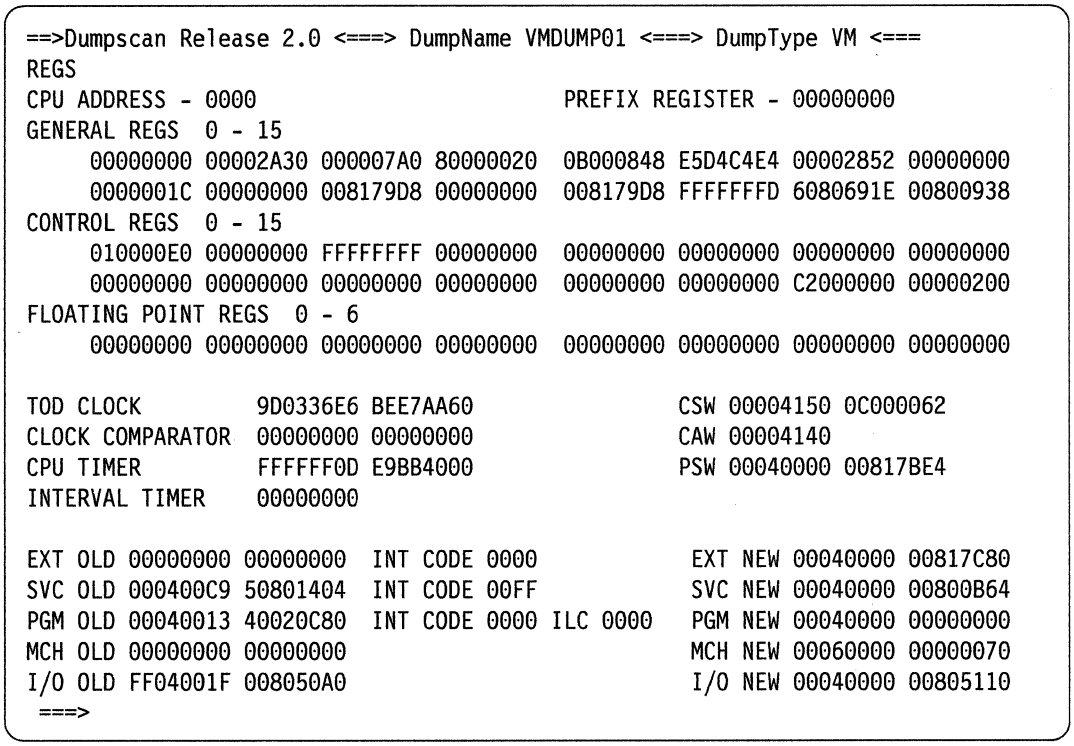
```
regs 01
```

```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
REGS    01
CPU ADDRESS - 0001                         PREFIX REGISTER - 01CB0000
GENERAL REGS  0 - 15
     00000000 40C5E7E3 00001201 033C0000  80A27F5E 80000002 0000005D 01F505C0
     00923000 809F6A80 00900000 00900000  009F6270 00000018 809F6EF0 00A27A40
CONTROL REGS  0 - 15
     80B0FE40 00800001 00000000 00000000  00000000 00000000 48000000 00000000
     00000000 00000000 00000000 00000000  01C51721 00000000 5F000000 00000000
FLOATING POINT REGS  0 - 6
     00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000

TOD CLOCK          NOT AVAILABLE
CLOCK COMPARATOR   FFFFFFFF FFFFF000
CPU TIMER          7FFFFFFE FFA66000


EXT OLD 033C0000 80A27F5E  INT CODE 1201            EXT NEW 000C0000 809A2290
SVC OLD 00000000 00000000  INT CODE 0000            SVC NEW 000C0000 809FB2B0
PGM OLD 000C1000 80981B76  INT CODE 0001 ILC 0004   PGM NEW 000A0000 00009005
MCH OLD 00000000 00000000                           MCH NEW 000A0000 00009005
I/O OLD 00000000 00000000                           I/O NEW 000C0000 809B4518
====>
```

Figure 58. Sample Output of a REGS Subcommand

Figure 59 illustrates the output of the REGS subcommand for a virtual machine dump. The subcommand entered is:

```
regs
```



```
==>Dumpscan Release 2.0 <===> DumpName VMDUMP01 <===> DumpType VM <===
REGS
CPU ADDRESS - 0000                         PREFIX REGISTER - 00000000
GENERAL REGS  0 - 15
     00000000 00002A30 000007A0 80000020  0B000848 E5D4C4E4 00002852 00000000
     0000001C 00000000 008179D8 00000000  008179D8 FFFFFFFD 6080691E 00800938
CONTROL REGS  0 - 15
     010000E0 00000000 FFFFFFFF 00000000  00000000 00000000 00000000 00000000
     00000000 00000000 00000000 00000000  00000000 00000000 C2000000 00000200
FLOATING POINT REGS  0 - 6
     00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000

TOD CLOCK         9D0336E6 BEE7AA60                CSW 00004150 0C000062
CLOCK COMPARATOR  00000000 00000000                CAW 00004140
CPU TIMER         FFFFFF0D E9BB4000                PSW 00040000 00817BE4
INTERVAL TIMER    00000000

EXT OLD 00000000 00000000  INT CODE 0000            EXT NEW 00040000 00817C80
SVC OLD 000400C9 50801404  INT CODE 00FF            SVC NEW 00040000 00800B64
PGM OLD 00040013 40020C80  INT CODE 0000 ILC 0000   PGM NEW 00040000 00000000
MCH OLD 00000000 00000000                           MCH NEW 00060000 00000070
I/O OLD FF04001F 008050A0                           I/O NEW 00040000 00805110
 ===>
```

Figure 59. Sample Output of a REGS Subcommand for a Virtual Machine Dump

RIOBLOK Subcommand

Use the RIOBLOK subcommand to display summary information about real I/O control blocks. You can request information about a real device by real device number, logical real device number, real device block (RDEV) address, or by logical real device block address.

The format of the RIOBLOK subcommand is as follows:

RIOblok	[FOR] *device* [FOR] L*device* AT *address*

Where:

FOR device
is the 1- to 4-digit hexadecimal device number of the real device for which information is to be displayed. The keyword FOR may be omitted.

FOR Ldevice
is the 1- to 4-digit hexadecimal device number of the logical real device for which information is to be displayed. The L immediately precedes the logical real device number. The keyword FOR may be omitted.

AT address
is the 31-bit (4-byte) hexadecimal address of the real device block (RDEV) or the logical real device block address for which information is to be displayed. The address must be preceded by the AT keyword to distinguish it from a device number.

Usage Notes:

1. The RIOBLOK subcommand is not supported for virtual machine dumps.

2. You may enter a real device number or a logical device number without leading zeros. For example, to display information on logical real device hex 0AFF, enter RIOBLOK FOR LAFF.

3. The display screen is cleared upon every successful invocation of the subcommand. For example, whenever real device information is located and displayed, the display screen is cleared prior to presentation of the requested data.

4. If information is to be displayed from a control block (other than the RDEV) that is not contained in the dump, NOT AVAILABLE appears in place of the control block data. For example: USER VMDBK USERID = NOT AVAILABLE or LOCK OWNER VMDBK USERID = NOT AVAILABLE.

5. If information is to be displayed about a device that does not have a currently active IORBK, NO ACTIVE IORBK is displayed in place of the IORBK fields.

Responses:

Three types of information may be displayed for the RIOBLOK subcommand:

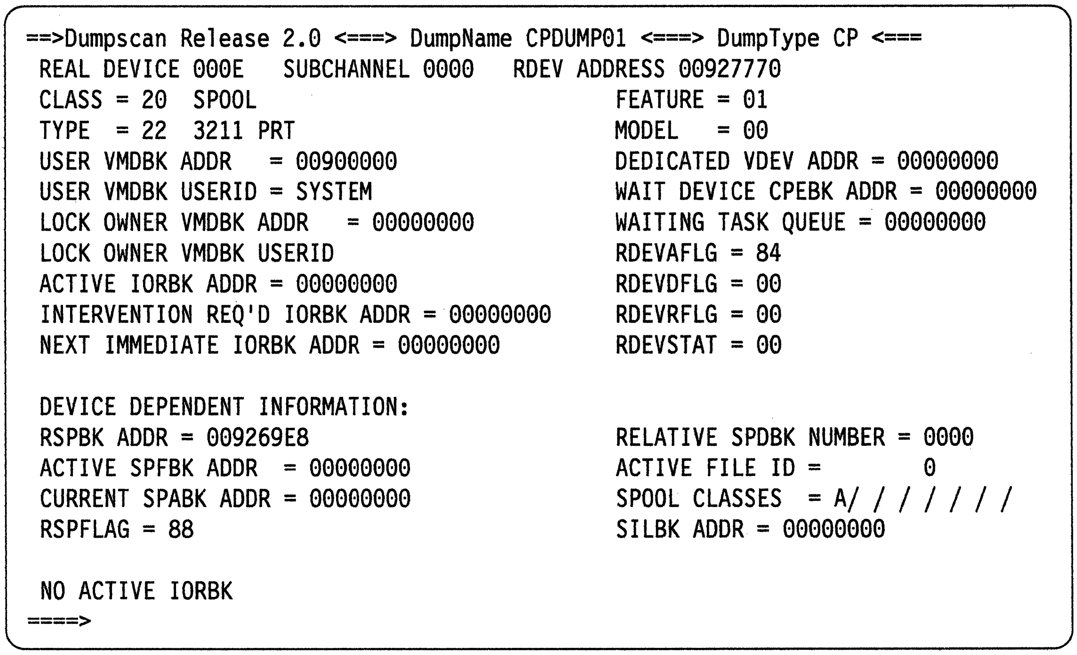
- General device information. Displayed for each device.
- Device-dependent information. Information about spooling, direct access storage devices, and display devices.
- Active IORBK information. If the real device block had I/O active at the time of the dump, information from the active I/O request and response block (IORBK) is provided.

For information on the corresponding dump data fields displayed by this command, refer to Appendix A.

Sample Output:

Figures 46 through 50 illustrate the output for the RIOBLOK subcommand with various device types specified. The subcommand entered for Figure 60 is:

```
rio for e
```

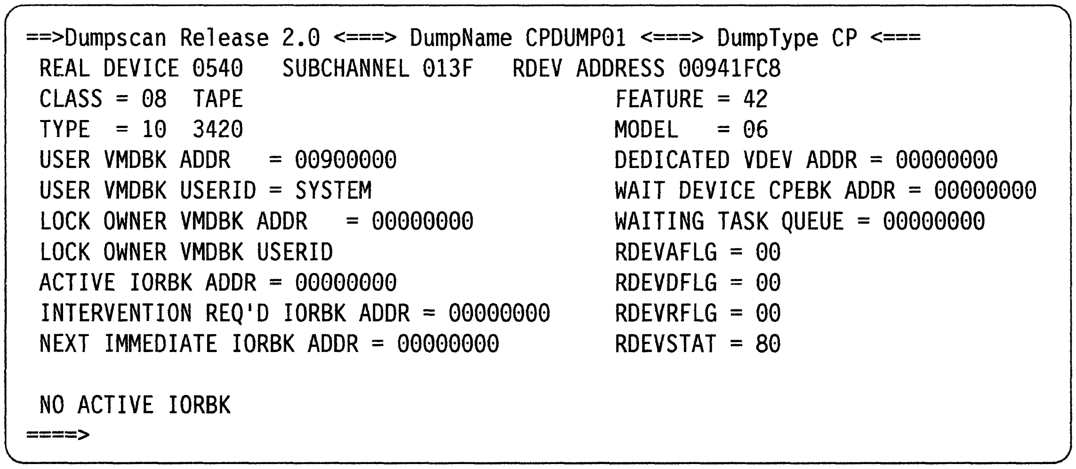


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 REAL DEVICE 000E   SUBCHANNEL 0000   RDEV ADDRESS 00927770
 CLASS = 20  SPOOL                            FEATURE = 01
 TYPE  = 22  3211 PRT                         MODEL   = 00
 USER VMDBK ADDR   = 00900000                 DEDICATED VDEV ADDR = 00000000
 USER VMDBK USERID = SYSTEM                   WAIT DEVICE CPEBK ADDR = 00000000
 LOCK OWNER VMDBK ADDR   = 00000000           WAITING TASK QUEUE = 00000000
 LOCK OWNER VMDBK USERID                      RDEVAFLG = 84
 ACTIVE IORBK ADDR = 00000000                 RDEVDFLG = 00
 INTERVENTION REQ'D IORBK ADDR = 00000000     RDEVRFLG = 00
 NEXT IMMEDIATE IORBK ADDR = 00000000         RDEVSTAT = 00

 DEVICE DEPENDENT INFORMATION:
 RSPBK ADDR = 009269E8                        RELATIVE SPDBK NUMBER = 0000
 ACTIVE SPFBK ADDR  = 00000000                ACTIVE FILE ID =         0
 CURRENT SPABK ADDR = 00000000                SPOOL CLASSES  = A/ / / / / / /
 RSPFLAG = 88                                 SILBK ADDR = 00000000

 NO ACTIVE IORBK
====>
```

Figure 60. Sample Output of an RIOBLOK Subcommand for a Spooling Device

The subcommand entered for Figure 61 is:

```
rio for 540
```

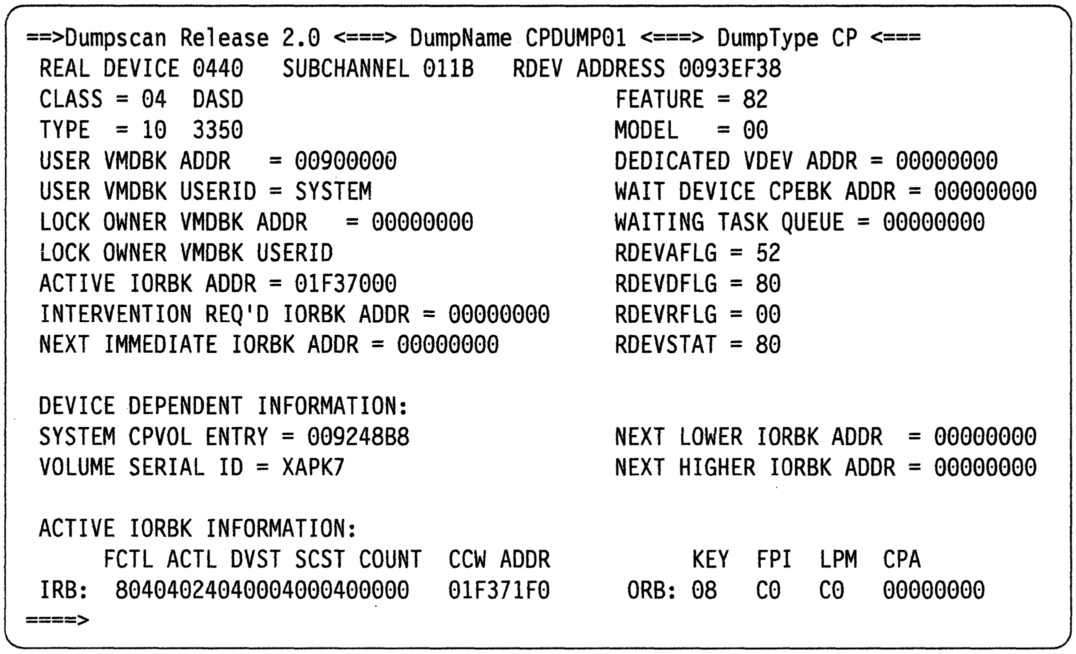


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 REAL DEVICE 0540   SUBCHANNEL 013F   RDEV ADDRESS 00941FC8
 CLASS = 08  TAPE                             FEATURE = 42
 TYPE  = 10  3420                             MODEL   = 06
 USER VMDBK ADDR   = 00900000                 DEDICATED VDEV ADDR = 00000000
 USER VMDBK USERID = SYSTEM                   WAIT DEVICE CPEBK ADDR = 00000000
 LOCK OWNER VMDBK ADDR   = 00000000           WAITING TASK QUEUE = 00000000
 LOCK OWNER VMDBK USERID                      RDEVAFLG = 00
 ACTIVE IORBK ADDR = 00000000                 RDEVDFLG = 00
 INTERVENTION REQ'D IORBK ADDR = 00000000     RDEVRFLG = 00
 NEXT IMMEDIATE IORBK ADDR = 00000000         RDEVSTAT = 80

 NO ACTIVE IORBK
====>
```

Figure 61. Sample Output of an RIOBLOK Subcommand for a Tape Device

The subcommand entered for Figure 62 is:

```
rio 440
```

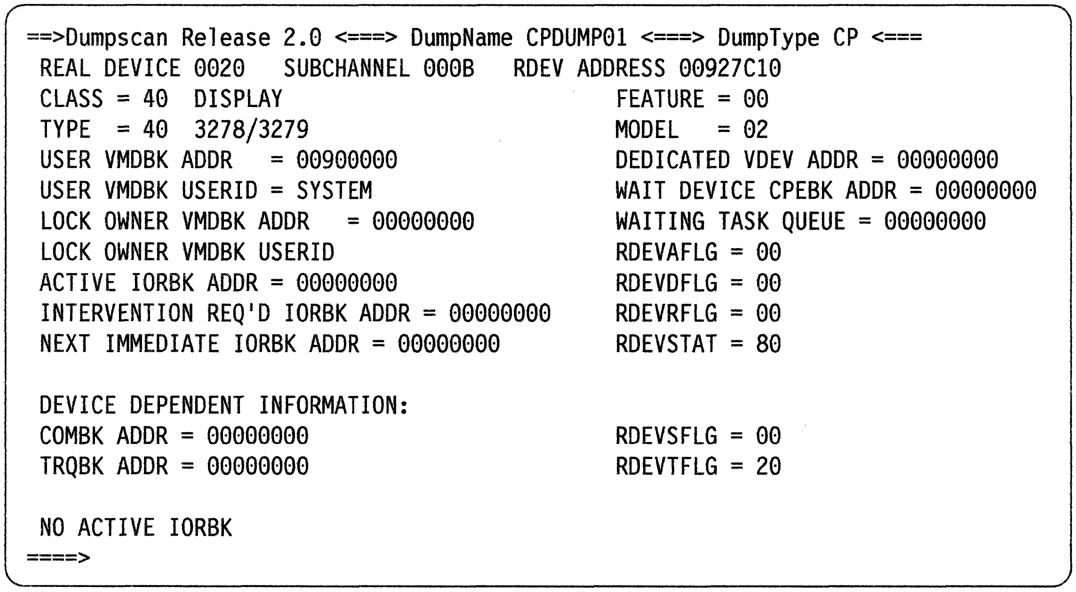


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 REAL DEVICE 0440   SUBCHANNEL 011B   RDEV ADDRESS 0093EF38
 CLASS = 04  DASD                             FEATURE = 82
 TYPE  = 10  3350                             MODEL   = 00
 USER VMDBK ADDR   = 00900000                 DEDICATED VDEV ADDR = 00000000
 USER VMDBK USERID = SYSTEM                   WAIT DEVICE CPEBK ADDR = 00000000
 LOCK OWNER VMDBK ADDR   = 00000000           WAITING TASK QUEUE = 00000000
 LOCK OWNER VMDBK USERID                      RDEVAFLG = 52
 ACTIVE IORBK ADDR = 01F37000                 RDEVDFLG = 80
 INTERVENTION REQ'D IORBK ADDR = 00000000     RDEVRFLG = 00
 NEXT IMMEDIATE IORBK ADDR = 00000000         RDEVSTAT = 80

 DEVICE DEPENDENT INFORMATION:
 SYSTEM CPVOL ENTRY = 009248B8                NEXT LOWER IORBK ADDR  = 00000000
 VOLUME SERIAL ID = XAPK7                     NEXT HIGHER IORBK ADDR = 00000000

 ACTIVE IORBK INFORMATION:
      FCTL ACTL DVST SCST COUNT  CCW ADDR            KEY  FPI  LPM  CPA
 IRB:  80404024040004000400000   01F371F0       ORB: 08   C0   C0   00000000
====>
```

Figure 62. Sample Output of an RIOBLOK Subcommand for a Device with an Active IORBK

The subcommand entered for Figure 63 is:

```
rio 20
```

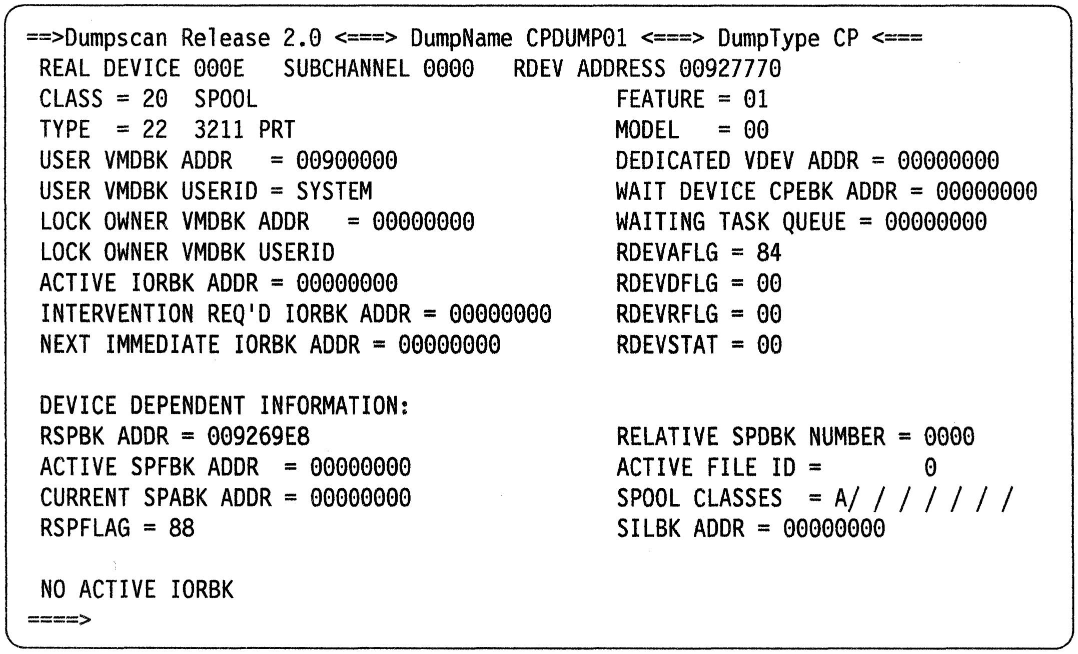


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 REAL DEVICE 0020   SUBCHANNEL 000B   RDEV ADDRESS 00927C10
 CLASS = 40  DISPLAY                          FEATURE = 00
 TYPE  = 40  3278/3279                        MODEL    = 02
 USER VMDBK ADDR   = 00900000                 DEDICATED VDEV ADDR = 00000000
 USER VMDBK USERID = SYSTEM                   WAIT DEVICE CPEBK ADDR = 00000000
 LOCK OWNER VMDBK ADDR   = 00000000           WAITING TASK QUEUE = 00000000
 LOCK OWNER VMDBK USERID                      RDEVAFLG = 00
 ACTIVE IORBK ADDR = 00000000                 RDEVDFLG = 00
 INTERVENTION REQ'D IORBK ADDR = 00000000     RDEVRFLG = 00
 NEXT IMMEDIATE IORBK ADDR = 00000000         RDEVSTAT = 80

 DEVICE DEPENDENT INFORMATION:
 COMBK ADDR = 00000000                        RDEVSFLG = 00
 TRQBK ADDR = 00000000                        RDEVTFLG = 20

 NO ACTIVE IORBK
====>
```

Figure 63. Sample Output of an RIOBLOK Subcommand for a Display Device

The subcommand entered for Figure 64 is:

```
rio at 927770
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 REAL DEVICE 000E   SUBCHANNEL 0000   RDEV ADDRESS 00927770
 CLASS = 20  SPOOL                            FEATURE = 01
 TYPE  = 22  3211 PRT                         MODEL    = 00
 USER VMDBK ADDR   = 00900000                 DEDICATED VDEV ADDR = 00000000
 USER VMDBK USERID = SYSTEM                   WAIT DEVICE CPEBK ADDR = 00000000
 LOCK OWNER VMDBK ADDR   = 00000000           WAITING TASK QUEUE = 00000000
 LOCK OWNER VMDBK USERID                      RDEVAFLG = 84
 ACTIVE IORBK ADDR = 00000000                 RDEVDFLG = 00
 INTERVENTION REQ'D IORBK ADDR = 00000000     RDEVRFLG = 00
 NEXT IMMEDIATE IORBK ADDR = 00000000         RDEVSTAT = 00

 DEVICE DEPENDENT INFORMATION:
 RSPBK ADDR = 009269E8                        RELATIVE SPDBK NUMBER = 0000
 ACTIVE SPFBK ADDR  = 00000000                ACTIVE FILE ID =        0
 CURRENT SPABK ADDR = 00000000                SPOOL CLASSES  = A/ / / / / / /
 RSPFLAG = 88                                 SILBK ADDR = 00000000

 NO ACTIVE IORBK
====>
```

Figure 64. Sample Output of an RIOBLOK Subcommand with an Address Specified

SELECT Subcommand

Use the SELECT subcommand to select the trace table entries you want to see. Reduction capability is enabled by a number of options.

There are three forms of the SELECT subcommand. The format of the SELECT subcommand to select all of the trace entries follows:

SELect	**ALL**

The format of the SELECT subcommand to enable or reset selection criteria follows:

SELect	**CODE [RESET]** { *value1*[*value2...*] } **CPU** { **ALL** } **RDEV** **VDEV** **VMDBK** **USER**

The format of the SELECT subcommand to display the current selection criteria in effect follows:

SELect	[<u>**QUERY**</u>]

Where:

CODE
is the keyword to specify trace entries with the CODE values requested to be either SET or RESET.

CPU
is the keyword to specify trace entries with the CPU values requested to be either SET or RESET.

RDEV
is the keyword to specify trace entries with the RDEV values requested to be either SET or RESET.

VDEV
is the keyword to specify trace entries with the VDEV values requested to be either SET or RESET.

VMDBK
is the keyword to specify trace entries with the VMDBK address values requested to be either SET or RESET.

USER
is the keyword to specify trace entries with the USERID values requested to be either SET or RESET.

ALL
is the keyword to specify that all trace entries for the trace tables will be displayed. There is no selectivity when ALL is specified.

QUERY
is the keyword to display current selection criteria in effect. If no operands are specified, QUERY is the default. No other operands may be specified with the QUERY keyword.

RESET
allows the deletion of one or more selection values in effect for the type specified. RESET should follow a keyword and precede the ALL or values parameter.

value(s)
is the keyword for the specific value requested. There up are to 16 values allowed.

Note: If a 17th value is requested, a message is issued to inform you that the maximum operand has been met.

ALL
is the keyword to specify that all values for a specific request should be displayed. ALL is specified instead of values. If ALL is specified with one or more values, it will result in error message 801E.

All is an implied RESET whenever you use it. The RESET operand need not be entered when ALL is the selected value.

CODE
is the 4-digit hexadecimal value of trace codes. Leading zeros are required. The last two digits of any of the codes can be specified as the character string * or ** to indicate that you want a range of trace codes. For example, for codes beginning with 15, you would enter either SELECT CODE 15** or SELECT CODE 15*.

Note: If you select 15* or 15** and a reset of 1501 is requested, you receive an error message. Conversely, if you select 1501, then request a reset for 15* or 15**, you receive an error message.

CPU
is a 4-digit hexadecimal value. Leading zeros are not required. Each CPU address entered must be in the range of 0 to 3F.

RDEV
is a value of up to four hexadecimal digits which represents the RDEV. Leading zeros are not required.

VDEV
is a value of up to four hexadecimal digits representing the VDEV. Leading zeros are not required. Each device number must fall in the range of 0 to FFFF.

VMDBK
is a value of up to eight hexadecimal digits representing the VMDBK address. Leadings zeros are not required. Each address must be in the of 0000 to 7FFFF000 and on a 4K boundary.

USER
is a keyword of up to 8 characters representing the USERID.

Usage Notes:

1. The dump viewing facility does not support trace table viewing in virtual machine dumps, so viewing is limited to CP and stand-alone dumps.

2. You cannot make multiple combinations of reduction selection type keywords on the same invocation. Subsequent invocations of SELECT can add or delete reduction criteria for trace entries.

3. A maximum of 16 values can be set for each selection type. This number applies to the total number of invocations of SELECT.

4. A maximum of 16 values can be set for each of the selection types. This number applies to the total number of invocations of SELECT.

5. SELECT ALL will delete all values that have been saved due to prior selection criteria.

6. Enter:

   ```
   SELECT keyword RESET ALL
   ```

   to reset just the values for a specific keyword.

7. Enter:

   ```
   SELECT keyword RESET value
   ```

   to reset a specific value.

8. If any error occurs on the command line, you receive a message that your request will not be processed.

9. An individual trace entry may or may not contain a field of the same type specified by the SELECT subcommand. Certain selection criteria are always present in a trace entry. The presence of other criteria are dependent on the trace entry's architecture.

SELECTION TYPE	ALWAYS IN A TRACE ENTRY
CODE	YES
CPU	YES
RDEV	NO
USER	NO
VDEV	NO
VMDBK	NO

   As a result, there are two main rules which apply to trace entry selection.

   - When only one selection type is in affect, only entries having that selection type and value are chosen.

     For instance:

     ```
     SELECT VMDBK 1000 2000
     TRACE
     ```

     Only entries which have VMDBK addresses of 1000 and 2000 are viewed.

   - Trace selection when multiple selection types were entered (combinations of CODE, VMDBK, RDEV, VDEV, CPU) follow these guidelines:

     a. Trace entries that contain fields for only some of the selection types you indicated, but whose value in those fields match the values you designated, are chosen for output.

     b. A trace entry does not have to have all the selection types you specified in order to be chosen for output.

For instance:

```
SELECT CODE 0C32 1000 0100
SELECT RDEV 399
SELECT VDEV 191
TRACE
```

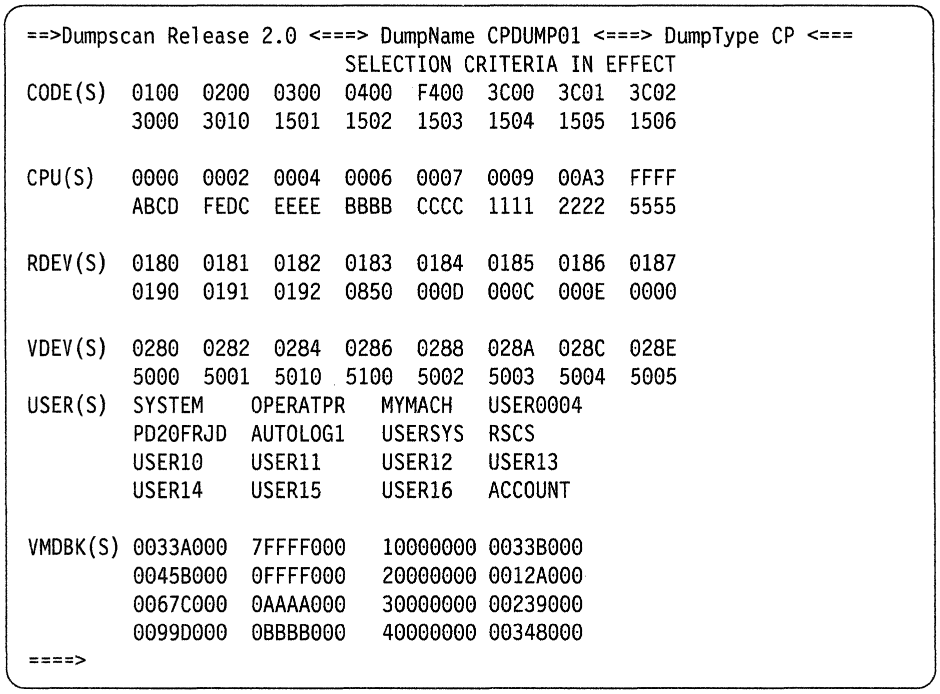
In this case, trace entries to be selected for output would depend on the following:

- Trace entries with code 0C32 contain both an RDEV and VDEV value. Therefore, a trace entry of 0C32 must have an RDEV field value of 399 and a VDEV field value of 191 in order to be selected for output.
- Trace entries with code 1000 contain only an RDEV value. The VDEV selection value does not apply in this situation. Therefore, all trace entries of code 1000 with an RDEV field value of 399 are chosen for output.
- Trace entries with code 0100 have neither an RDEV nor VDEV value. Therefore all trace entries of code 0100 are chosen for output.

Responses:

The following is the response to a request to determine the criteria that have been selected via the SELECT subcommand. Only those items which were actually selected are displayed. For example, if seven codes were selected, only seven codes are displayed. Each of the SELECT keywords can contain a maximum of 16 values, except the TIME keyword. The subcommand entered for Figure 65 is:

```
select query
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
                               SELECTION CRITERIA IN EFFECT
CODE(S)  0100  0200  0300  0400  F400  3C00  3C01  3C02
         3000  3010  1501  1502  1503  1504  1505  1506

CPU(S)   0000  0002  0004  0006  0007  0009  00A3  FFFF
         ABCD  FEDC  EEEE  BBBB  CCCC  1111  2222  5555

RDEV(S)  0180  0181  0182  0183  0184  0185  0186  0187
         0190  0191  0192  0850  000D  000C  000E  0000

VDEV(S)  0280  0282  0284  0286  0288  028A  028C  028E
         5000  5001  5010  5100  5002  5003  5004  5005
USER(S)  SYSTEM    OPERATPR    MYMACH   USER0004
         PD20FRJD  AUTOLOG1    USERSYS  RSCS
         USER10    USER11      USER12   USER13
         USER14    USER15      USER16   ACCOUNT

VMDBK(S) 0033A000  7FFFF000    10000000 0033B000
         0045B000  0FFFF000    20000000 0012A000
         0067C000  0AAAA000    30000000 00239000
         0099D000  0BBBB000    40000000 00348000
====>
```

Figure 65. Output from a SELECT QUERY request

SNAPLIST Subcommand

Use the SNAPLIST subcommand to view a summary snaplist in a soft abend dump. This provides you with a list of addresses for the snapped data, and the buffer area or areas which contain a copy of the data at specified addresses.

The format of the SNAPLIST subcommand is as follows:

SNAPlist	

Usage Notes:

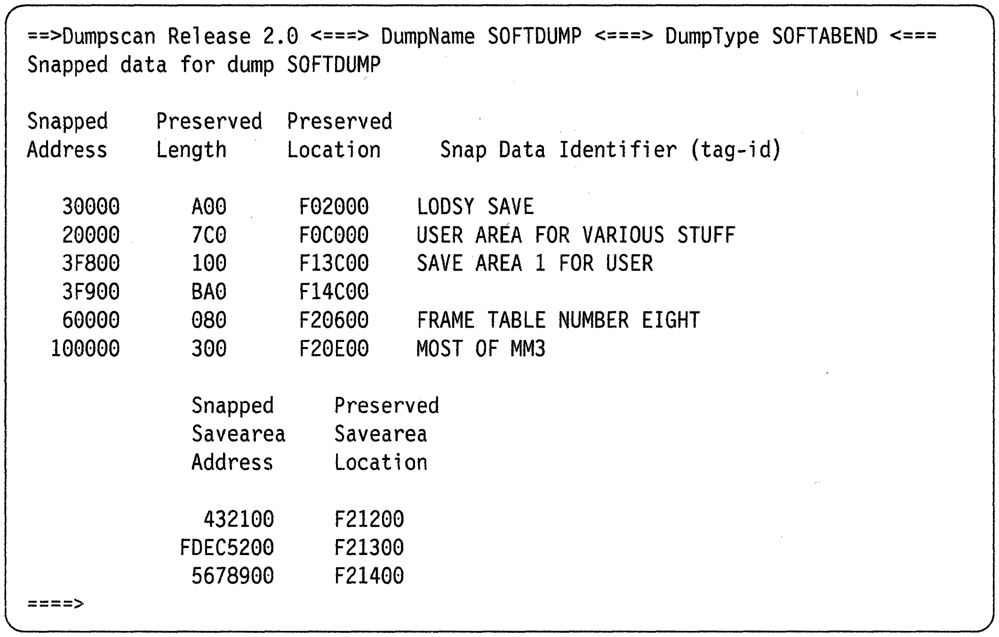
1. Only soft abend dump types have snaplists. When you use the SNAPLIST subcommand on any other type of dump, you receive an error message.
2. If you issue the SNAPLIST subcommand for a soft abend dump that does not have a snaplist, you receive an message informing you of this condition.
3. If the status flags indicate that a certain field to be displayed is not complete, you receive information regarding the incomplete information in the output of your request.
4. The information available to you from the SNAPLIST subcommand includes:
   - The address of snapped data and length, which enables you to locate the snapped area and the amount of information you need to display
   - The address of the dumped page from which snapped data comes, which may be used for comparative study.
5. The SNAPLIST subcommand can display only the preserved data. If you specify that the abending module should preserve save areas and not snap data, you can only display save areas.

Responses:

The following is the response to the DUMPSCAN subcommand SNAPLIST, showing both module areas and save areas. This figure shows the SNAPLIST subcommand issued against a soft abend dump.

The subcommand entered for Figure 66 on page 132 is:

```
snaplist
```



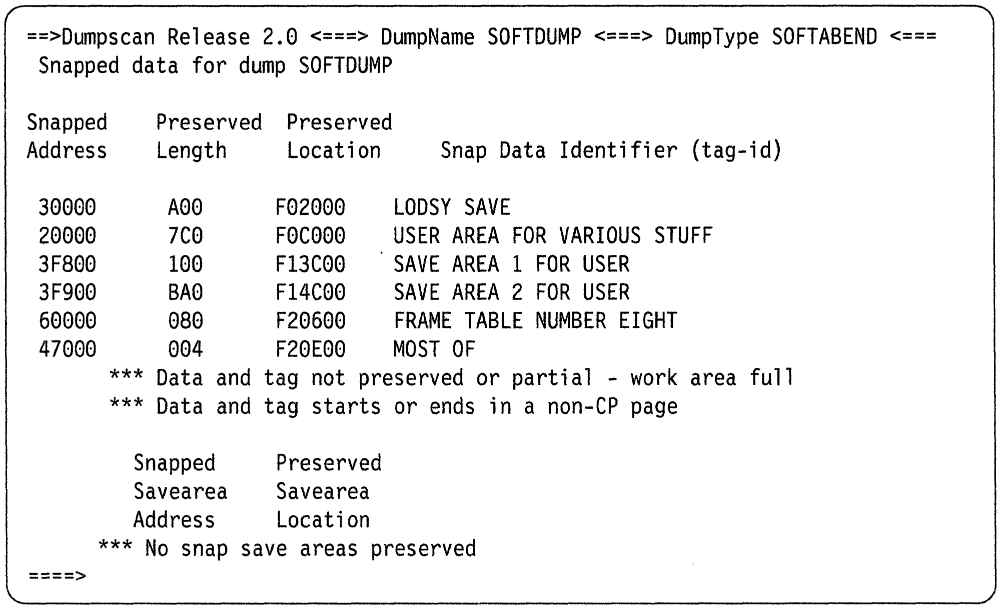
```
==>Dumpscan Release 2.0 <===> DumpName SOFTDUMP <===> DumpType SOFTABEND <===
Snapped data for dump SOFTDUMP

Snapped    Preserved  Preserved
Address    Length     Location     Snap Data Identifier (tag-id)

   30000      A00      F02000    LODSY SAVE
   20000      7C0      F0C000    USER AREA FOR VARIOUS STUFF
   3F800      100      F13C00    SAVE AREA 1 FOR USER
   3F900      BA0      F14C00
   60000      080      F20600    FRAME TABLE NUMBER EIGHT
  100000      300      F20E00    MOST OF MM3

               Snapped     Preserved
               Savearea    Savearea
               Address     Location

                432100     F21200
              FDEC5200     F21300
               5678900     F21400
====>
```

Figure 66. Snapped Module and Save Areas

The following is the response to the DUMPSCAN subcommand SNAPLIST, showing the work area is full and the snap data and the snap data identifier tag are located, at least partially, in non-CP pages. The save areas were not requested.

The subcommand entered for Figure 67 is:

```
snaplist
```



```
==>Dumpscan Release 2.0 <===> DumpName SOFTDUMP <===> DumpType SOFTABEND <===
 Snapped data for dump SOFTDUMP

Snapped    Preserved  Preserved
Address    Length     Location     Snap Data Identifier (tag-id)

 30000      A00      F02000    LODSY SAVE
 20000      7C0      F0C000    USER AREA FOR VARIOUS STUFF
 3F800      100      F13C00    SAVE AREA 1 FOR USER
 3F900      BA0      F14C00    SAVE AREA 2 FOR USER
 60000      080      F20600    FRAME TABLE NUMBER EIGHT
 47000      004      F20E00    MOST OF
       *** Data and tag not preserved or partial - work area full
       *** Data and tag starts or ends in a non-CP page

         Snapped     Preserved
         Savearea    Savearea
         Address     Location
      *** No snap save areas preserved
====>
```

Figure 67. SNAPLIST Truncated Due to Full Work Area

Additional SNAPLIST Responses:

Creation and content of the soft abend dump is through the use of HCPABEND. The SNAPLIST option of HCPABEND specifies the SDPL that describes the addresses and sizes of the data to be saved. The SNAPSAVE option of HCPABEND specifies whether the save areas are to be preserved. The following responses result from the various conditions that can be encountered depending on the use of HCPABEND.

```
*** Snapped data page list starts in a non-CP page
```

This is the response when the first snap data page is located in a non-CP page and no further processing of snap data is possible.

```
*** No snap data entries preserved
```

This is the response when the snap data areas were not requested.

```
*** Snaplist truncated - work area is full
```

This is the response when the amount of snap data requested exhausts the preallocated work area.

```
*** List has been truncated - either a non-CP page or more than 256
    entries were encountered.
```

This is the response when the last SPDL entry points to a non-CP page or the list of snap data addresses was greater than 256.

```
*** Save area page list starts in a non-CP page
```

This is the response when SAVER13 in the SAVEBK points to a non-CP page.

```
*** No snap save areas preserved
```

This is the response when HCPABEND is specified with SAVEAREA=NO, the default.

```
*** Save area list truncated - work area full
```

This is the response when the pre-allocated work area was exhausted by previous save area processing or by snap data area processing.

```
*** Snap data not preserved - work area full
```

This is the response when the the pre-allocated work area was exhausted by snap data area processing.

```
*** Data and tag starts or ends in a non-CP page
```

This is the response when an SPDL entry contains an SPDLID and an SPDLDA that point to data which is partially in non-CP pages.

```
*** Data starts or ends in a non-CP page
```

This is the response when an SPDL entry contains an SPDLDA that points to data which is partially in non-CP pages.

```
*** Tag starts or ends in a non-CP page
```

This is the response when an SPDL entry contains an SPDLID that points to a tag which is partially in a non-CP page.

```
*** Data not preserved or partial - work area full
```

This is the response when the work area is full and the last snap area is incomplete.

```
*** Tag not preserved or partial - work area full
```

This is the response when the work area is full and the last snap id is incomplete.

```
*** Data and tag not preserved or partial - work area full
```

This is the response when the work area is full and the last snap id and snap data area are incomplete.

SYMPTOM Subcommand

Use the SYMPTOM subcommand to display formatted symptom record information at your terminal. The format of the SYMPTOM subcommand is as follows:

SYMPtom	

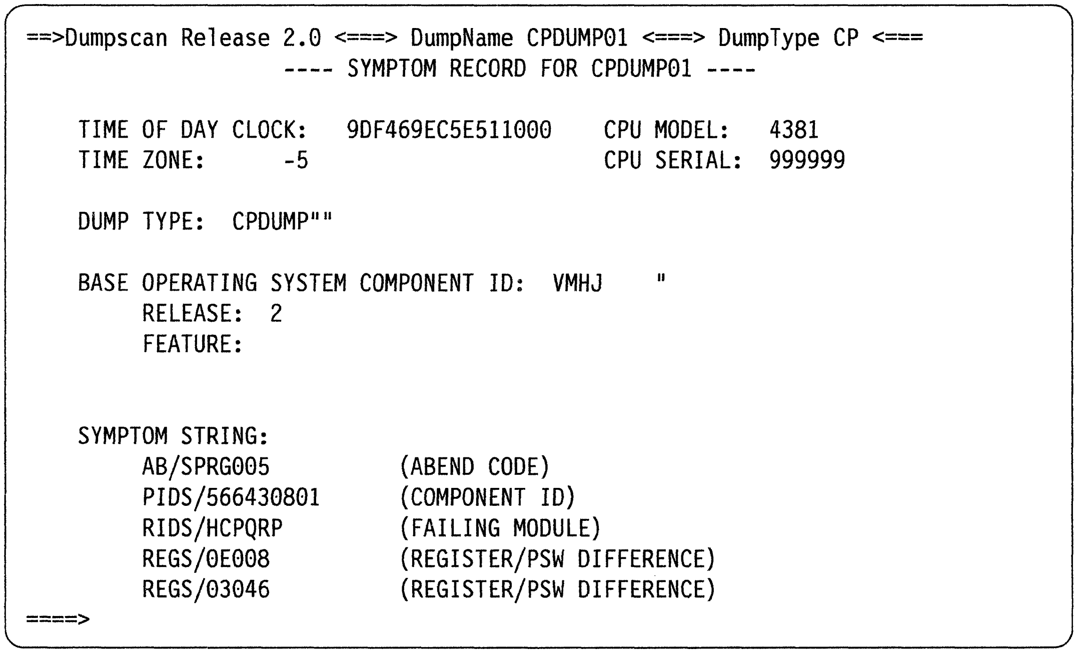
Usage Notes:

1. The screen is cleared before the symptom record information is displayed.
2. If a symptom string is not in the dump symptom record in the dump, it is not displayed.
3. If the dump symptom record is missing or not readable, an error message is issued.

Sample Output:

Figure 68 illustrates the format of the output screen for a CP dump when the SYMPTOM subcommand is entered. The subcommand entered is:

```
symptom
```

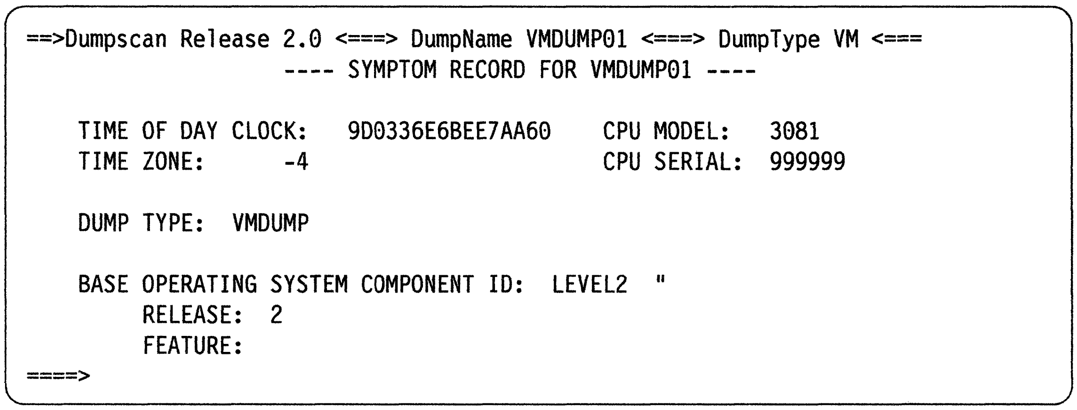


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
                    ---- SYMPTOM RECORD FOR CPDUMP01 ----

     TIME OF DAY CLOCK:   9DF469EC5E511000    CPU MODEL:   4381
     TIME ZONE:      -5                       CPU SERIAL:  999999

     DUMP TYPE:  CPDUMP""

     BASE OPERATING SYSTEM COMPONENT ID:  VMHJ    "
          RELEASE:  2
          FEATURE:


     SYMPTOM STRING:
          AB/SPRG005          (ABEND CODE)
          PIDS/566430801      (COMPONENT ID)
          RIDS/HCPQRP         (FAILING MODULE)
          REGS/0E008          (REGISTER/PSW DIFFERENCE)
          REGS/03046          (REGISTER/PSW DIFFERENCE)
====>
```

Figure 68. Output Format of a SYMPTOM Subcommand for a CP Dump

Figure 69 illustrates the format of the output screen for a VM dump when the SYMPTOM subcommand is entered. The subcommand entered is:

```
symptom
```



```
==>Dumpscan Release 2.0 <===> DumpName VMDUMP01 <===> DumpType VM <===
                    ---- SYMPTOM RECORD FOR VMDUMP01 ----

    TIME OF DAY CLOCK:   9D0336E6BEE7AA60    CPU MODEL:   3081
    TIME ZONE:      -4                       CPU SERIAL:  999999

    DUMP TYPE:  VMDUMP

    BASE OPERATING SYSTEM COMPONENT ID:  LEVEL2  "
         RELEASE:  2
         FEATURE:
====>
```

Figure 69. Output Format of a SYMPTOM Subcommand for a VM Dump

TRACE Subcommand

Use the TRACE subcommand to display trace table entries. You can merge the trace table entries for one or all of the processors in the dump. The output resulting from this command is formatted in either hexadecimal or EBCDIC characters.

Trace	[[FOR]*count*]	[**FROM** *address*]	[**FORMat** **HEX**]

Where:

FOR
Indicates that the next operand (the count variable) is the number of trace table entries to be displayed. The keyword FOR may be omitted.

count
is a decimal number from 1 to 999 specifying the number of entries in the trace table to be displayed.

FROM
indicates that the next operand (the address variable) is the storage address from which trace entries are to be displayed.

address
is the 31-bit (4-byte) hexadecimal address from which the trace entries are to be displayed. The trace table entry address must be on a 32-byte boundary.

FORMat
indicates that the TRACE subcommand output should be formatted before being displayed. If the FORMAT option is not specified, the output is not formatted before being displayed.

HEX
indicates that unformatted trace table entries should be displayed in hexadecimal. This is the default value.

Usage Notes:

1. The TRACE subcommand is not supported for virtual machine dumps.
2. The default number of trace table entries displayed is dependent on the size of output screen you are using. Because of this, the FOR *count* operand is optional.
3. If a *count* value entered is larger that the physical screen size, the XEDIT subcommands FORWARD and BACKWARD can be used to scroll through the output.
4. The operands HEX and FORMAT are mutually exclusive. If you enter these operands in any combination, you receive an error message.
5. The keywords and associated operands may be specified in any order.
6. Use the SELECT subcommand to control the trace table entries you want to view. Various trace entry reduction criteria can be enabled or reset with SELECT.

Responses:

Depending on the operands specified, a display is presented of the trace table entries, either in hexadecimal or formatted translation.

Note: If you need an explanation of the trace codes and trace table entry contents, see the manual *VM/XA SP CP Diagnosis Reference.*

Trace entries are displayed with the requested entry appearing last.

If the HEX keyword is entered, one line of output is displayed for each trace entry. Each trace entry contains the following fields: address, CPU ID, time-of-day clock, trace code, and trace entry contents. The CPU address and the address of the most recent entry are displayed in the first line on the screen. If the output contains trace tables merged from two or more processors, the CPU address is ALL.

If the FORMAT keyword is entered, the output for each trace entry consists of up to three lines of data. They are as follows:

Line #1 Trace code, CPU identifier, descriptive name, time-of-day clock, and address

Line #2 Headings describing the format of the trace entry contents

Line #3 Trace entry contents (up to 20 bytes of interpreted data)

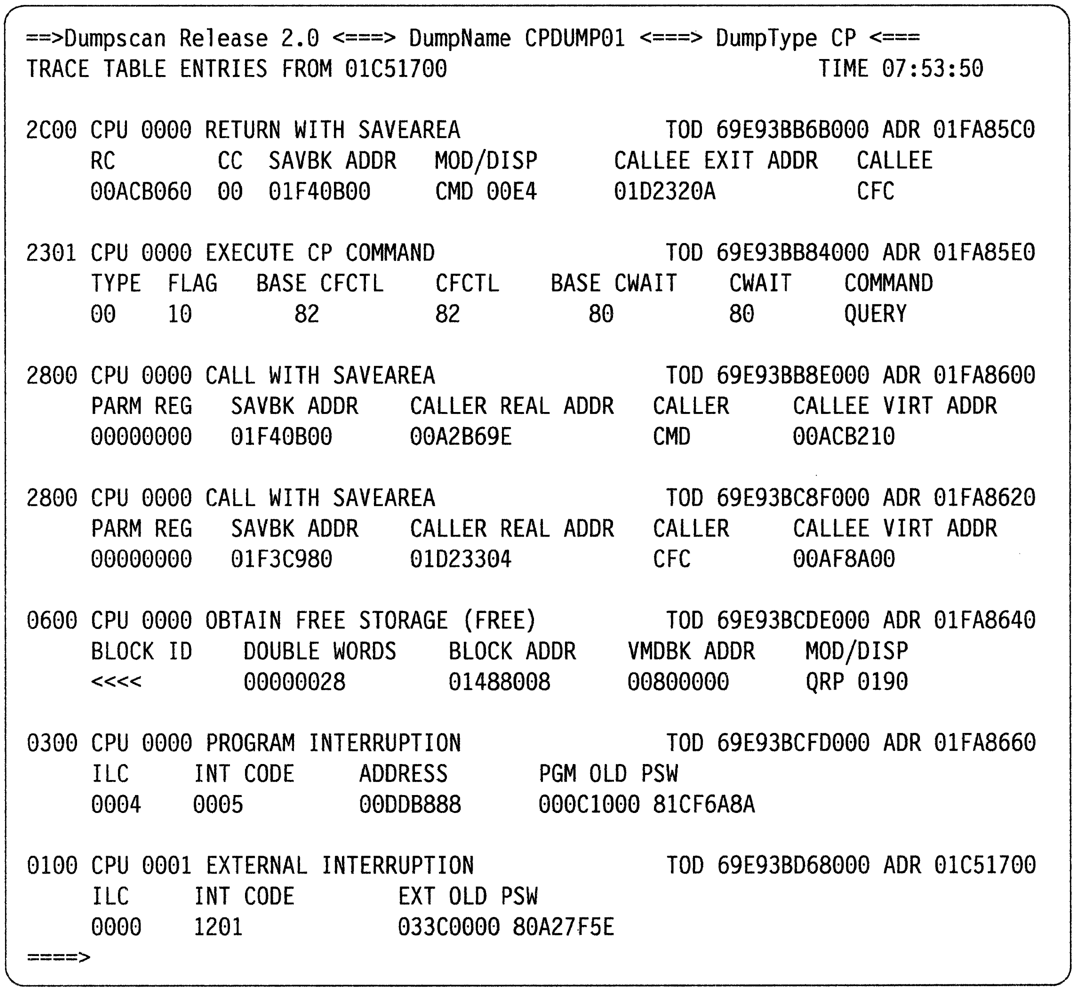
Line #4 Blank line.

Figure 70 shows how the formatted output looks.

Sample Output:

The subcommand entered for Figure 70 is:

```
trace form
```



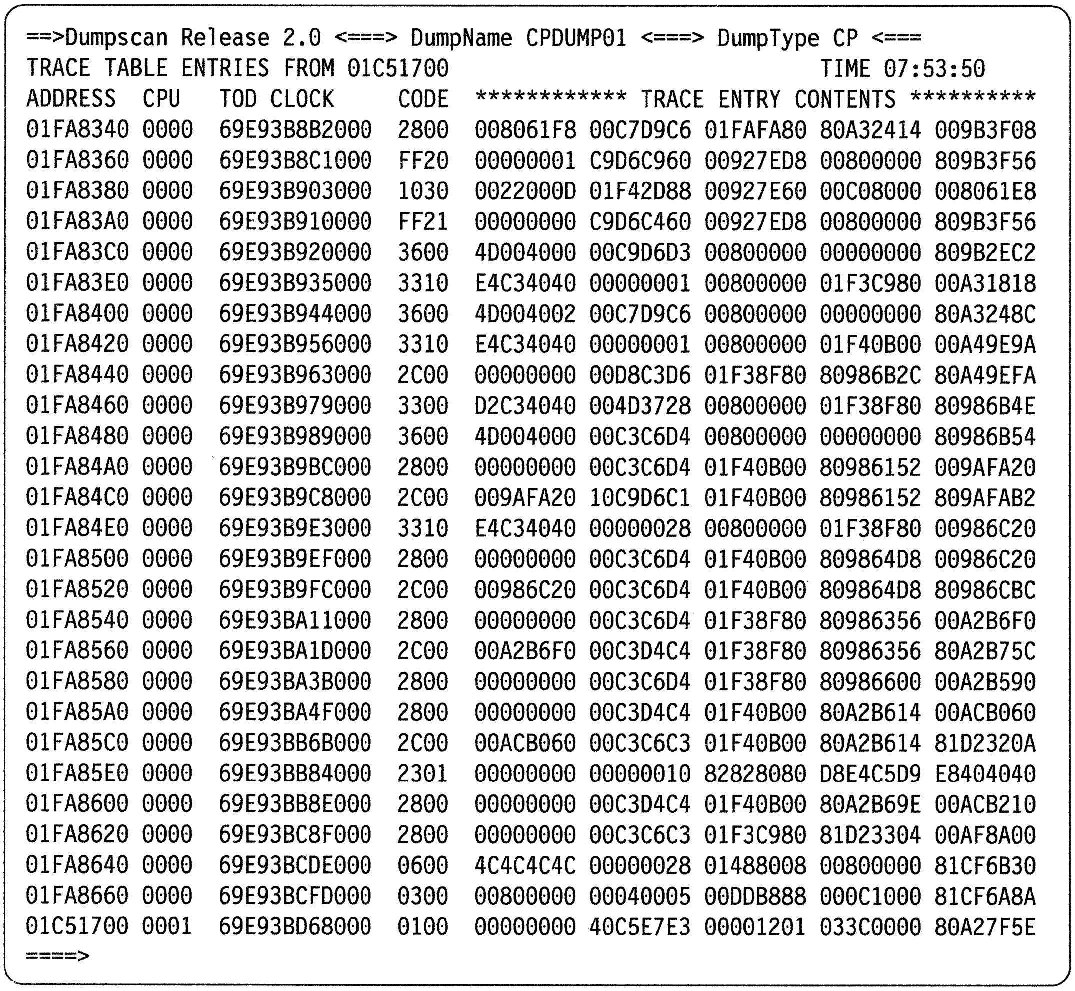
```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
TRACE TABLE ENTRIES FROM 01C51700                             TIME 07:53:50

2C00 CPU 0000 RETURN WITH SAVEAREA                TOD 69E93BB6B000 ADR 01FA85C0
     RC        CC  SAVBK ADDR   MOD/DISP      CALLEE EXIT ADDR   CALLEE
     00ACB060  00  01F40B00     CMD 00E4      01D2320A           CFC

2301 CPU 0000 EXECUTE CP COMMAND                  TOD 69E93BB84000 ADR 01FA85E0
     TYPE  FLAG   BASE CFCTL    CFCTL    BASE CWAIT    CWAIT    COMMAND
     00    10        82         82          80         80       QUERY

2800 CPU 0000 CALL WITH SAVEAREA                  TOD 69E93BB8E000 ADR 01FA8600
     PARM REG   SAVBK ADDR    CALLER REAL ADDR   CALLER     CALLEE VIRT ADDR
     00000000   01F40B00      00A2B69E           CMD        00ACB210

2800 CPU 0000 CALL WITH SAVEAREA                  TOD 69E93BC8F000 ADR 01FA8620
     PARM REG   SAVBK ADDR    CALLER REAL ADDR   CALLER     CALLEE VIRT ADDR
     00000000   01F3C980      01D23304           CFC        00AF8A00

0600 CPU 0000 OBTAIN FREE STORAGE (FREE)          TOD 69E93BCDE000 ADR 01FA8640
     BLOCK ID    DOUBLE WORDS    BLOCK ADDR    VMDBK ADDR    MOD/DISP
     <<<<        00000028        01488008      00800000      QRP 0190

0300 CPU 0000 PROGRAM INTERRUPTION                TOD 69E93BCFD000 ADR 01FA8660
     ILC     INT CODE     ADDRESS       PGM OLD PSW
     0004    0005         00DDB888      000C1000 81CF6A8A

0100 CPU 0001 EXTERNAL INTERRUPTION               TOD 69E93BD68000 ADR 01C51700
     ILC     INT CODE        EXT OLD PSW
     0000    1201            033C0000 80A27F5E
====>
```

Figure 70. Sample Output of a TRACE Subcommand with the FORMAT Operand Specified

Figure 71 illustrates the output of the TRACE subcommand with the default HEX operand.

The subcommand entered for Figure 71 is:

```
trace
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
TRACE TABLE ENTRIES FROM 01C51700                           TIME 07:53:50
ADDRESS  CPU   TOD CLOCK     CODE  ************ TRACE ENTRY CONTENTS **********
01FA8340 0000  69E93B8B2000  2800  008061F8 00C7D9C6 01FAFA80 80A32414 009B3F08
01FA8360 0000  69E93B8C1000  FF20  00000001 C9D6C960 00927ED8 00800000 809B3F56
01FA8380 0000  69E93B903000  1030  0022000D 01F42D88 00927E60 00C08000 008061E8
01FA83A0 0000  69E93B910000  FF21  00000000 C9D6C460 00927ED8 00800000 809B3F56
01FA83C0 0000  69E93B920000  3600  4D004000 00C9D6D3 00800000 00000000 809B2EC2
01FA83E0 0000  69E93B935000  3310  E4C34040 00000001 00800000 01F3C980 00A31818
01FA8400 0000  69E93B944000  3600  4D004002 00C7D9C6 00800000 00000000 80A3248C
01FA8420 0000  69E93B956000  3310  E4C34040 00000001 00800000 01F40B00 00A49E9A
01FA8440 0000  69E93B963000  2C00  00000000 00D8C3D6 01F38F80 80986B2C 80A49EFA
01FA8460 0000  69E93B979000  3300  D2C34040 004D3728 00800000 01F38F80 80986B4E
01FA8480 0000  69E93B989000  3600  4D004000 00C3C6D4 00800000 00000000 80986B54
01FA84A0 0000  69E93B9BC000  2800  00000000 00C3C6D4 01F40B00 80986152 009AFA20
01FA84C0 0000  69E93B9C8000  2C00  009AFA20 10C9D6C1 01F40B00 80986152 809AFAB2
01FA84E0 0000  69E93B9E3000  3310  E4C34040 00000028 00800000 01F38F80 00986C20
01FA8500 0000  69E93B9EF000  2800  00000000 00C3C6D4 01F40B00 809864D8 00986C20
01FA8520 0000  69E93B9FC000  2C00  00986C20 00C3C6D4 01F40B00 809864D8 80986CBC
01FA8540 0000  69E93BA11000  2800  00000000 00C3C6D4 01F38F80 80986356 00A2B6F0
01FA8560 0000  69E93BA1D000  2C00  00A2B6F0 00C3D4C4 01F38F80 80986356 80A2B75C
01FA8580 0000  69E93BA3B000  2800  00000000 00C3C6D4 01F38F80 80986600 00A2B590
01FA85A0 0000  69E93BA4F000  2800  00000000 00C3D4C4 01F40B00 80A2B614 00ACB060
01FA85C0 0000  69E93BB6B000  2C00  00ACB060 00C3C6C3 01F40B00 80A2B614 81D2320A
01FA85E0 0000  69E93BB84000  2301  00000000 00000010 82828080 D8E4C5D9 E8404040
01FA8600 0000  69E93BB8E000  2800  00000000 00C3D4C4 01F40B00 80A2B69E 00ACB210
01FA8620 0000  69E93BC8F000  2800  00000000 00C3C6C3 01F3C980 81D23304 00AF8A00
01FA8640 0000  69E93BCDE000  0600  4C4C4C4C 00000028 01488008 00800000 81CF6B30
01FA8660 0000  69E93BCFD000  0300  00800000 00040005 00DDB888 000C1000 81CF6A8A
01C51700 0001  69E93BD68000  0100  00000000 40C5E7E3 00001201 033C0000 80A27F5E
====>
```

Figure 71. Sample Output of a TRACE Subcommand in Hexadecimal

VIOBLOK Subcommand

Use the VIOBLOK subcommand to display summary information about virtual I/O control blocks. When you issue this subcommand, the system displays information about a virtual device or subchannel for the userid specified. You may also request information by the virtual device block (VDEV) address.

The format of the VIOBLOK subcommand is as follows:

```
VIOblok     ┌                      ┐     ┌        ┐
            │ FOR device           │     │ userid │
            │ FOR SCH subchannel   │     │ userid │
            └                      ┘     └        ┘
              AT    address
```

Where:

FOR device
is a 1- to 4-digit hexadecimal device number of the virtual device for which information is to be displayed. You may enter only one virtual device number. The keyword FOR may be omitted.

FOR SCH subchannel
is the 1- to 4-digit hexadecimal number of the virtual subchannel for which information is to be displayed. You may enter only one subchannel number. You must precede the subchannel number by the SCH keyword to distinguish it from a device number. The keyword FOR may be omitted.

userid
is the 1- to 8-character userid of the virtual machine for which virtual device information is to be displayed. You may enter only one userid.

AT address
is the 31-bit (4-byte) hexadecimal address of the virtual device block (VDEV) for which information is to be displayed. You may enter only one address. You must precede the address by the AT keyword to distinguish it from a device number.

Usage Notes:

1. The VIOBLOK subcommand is not supported for virtual machine dumps.

2. You may enter the device number, subchannel number, and address operands without leading zeros.

3. If either the VIOBLOK or VMDBK subcommand has been invoked with a userid during the current DUMPSCAN session, the userid defaults to the virtual machine whose VMDBK was last requested by either of these subcommands.

   If neither the VIOBLOK nor VMDBK subcommand has been invoked during the current DUMPSCAN session, the userid defaults to the system operator. If the system operator cannot be determined, then you must not allow the userid to default.

4. If information is to be displayed from a control block (other than VDEV) that is not contained in the dump, NOT AVAILABLE appears in place of the control block data.

For example, if the VMDBK for the user cannot be found, USER MODE = NOT AVAILABLE is displayed and the top line includes USER NOT AVAILABLE.

If the RDEV cannot be found, then REAL DEVICE NUMBER = NOT AVAILABLE is displayed.

5. If information is to be displayed about a device that does not have currently active I/O, NO ACTIVE IORBK is displayed in place of the IORBK fields for existing VIOBLOKs.
6. This subcommand clears the screen for existing VIOBLOKs prior to displaying the requested information.
7. If you specify a userid for which more than one VMDBK is found, the VIOBLOK subcommand terminates with message 624I (USERID userid IS A DUPLICATE USERID). You may still display virtual I/O information about the desired userid by using the following procedure:
   a. Issue the VMDBK LIST subcommand to determine the address of the desired VMDBK.
   b. Once you know the address of the VMDBK for the required userid, issue the VMDBK AT *address* subcommand to set the default VMDBK.
   c. Issue the VIOBLOK subcommand without a userid for the devices for which you want to view information.

Responses:

There are three types of information that the display for the VIOBLOK may include:

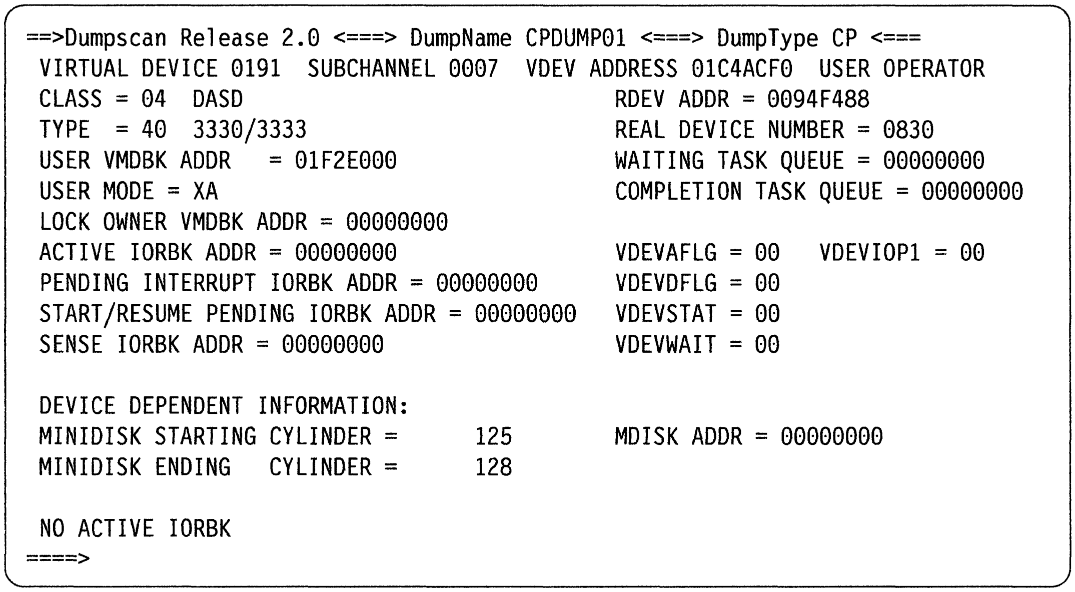
- General device information. Displayed for each device.
- Device-dependent information. Information about spooling, direct access storage devices (DASD), and channel-to-channel adapter (CTCA) devices.
- Active IORBK information. If the virtual device block had I/O active at the time of the dump, information from the active I/O request and response block (IORBK) is provided.

For information on the corresponding dump data fields displayed by this subcommand, refer to Appendix B.

Sample Output:

Figure 72 on page 143 illustrates the screen you see when using the VIOBLOK subcommand. The subcommand entered is:

```
vioblok at 1c4acf0
```

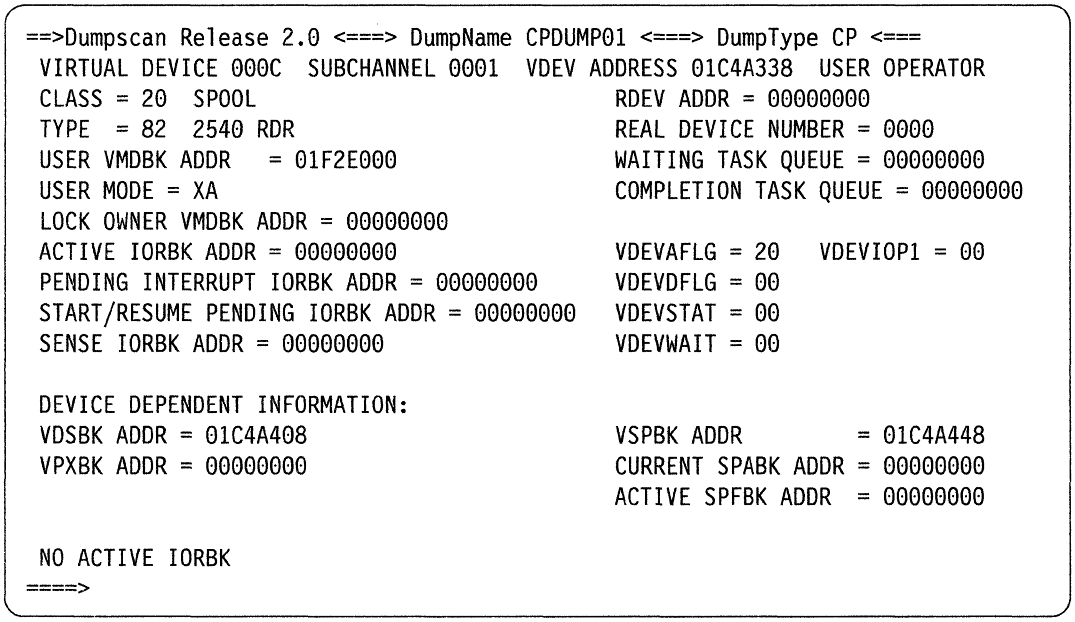


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 VIRTUAL DEVICE 0191  SUBCHANNEL 0007  VDEV ADDRESS 01C4ACF0  USER OPERATOR
 CLASS = 04  DASD                             RDEV ADDR = 0094F488
 TYPE  = 40  3330/3333                        REAL DEVICE NUMBER = 0830
 USER VMDBK ADDR   = 01F2E000                 WAITING TASK QUEUE = 00000000
 USER MODE = XA                               COMPLETION TASK QUEUE = 00000000
 LOCK OWNER VMDBK ADDR = 00000000
 ACTIVE IORBK ADDR = 00000000                 VDEVAFLG = 00   VDEVIOP1 = 00
 PENDING INTERRUPT IORBK ADDR = 00000000      VDEVDFLG = 00
 START/RESUME PENDING IORBK ADDR = 00000000   VDEVSTAT = 00
 SENSE IORBK ADDR = 00000000                  VDEVWAIT = 00

 DEVICE DEPENDENT INFORMATION:
 MINIDISK STARTING CYLINDER =      125        MDISK ADDR = 00000000
 MINIDISK ENDING   CYLINDER =      128

 NO ACTIVE IORBK
====>
```

Figure 72. Sample Output of a VIOBLOK Subcommand

Figures 59 through 62 illustrate the screens you see when information for various devices are requested using the VIOBLOK subcommand. The subcommand entered for Figure 73 is:

```
vio for c operator
```

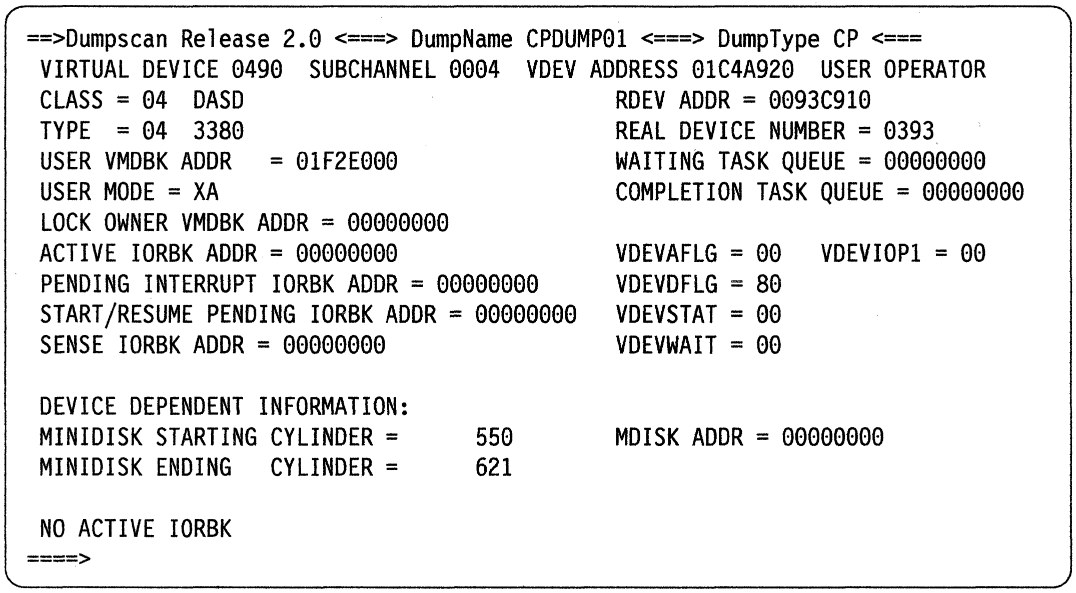


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 VIRTUAL DEVICE 000C  SUBCHANNEL 0001  VDEV ADDRESS 01C4A338  USER OPERATOR
 CLASS = 20  SPOOL                            RDEV ADDR = 00000000
 TYPE  = 82  2540 RDR                         REAL DEVICE NUMBER = 0000
 USER VMDBK ADDR   = 01F2E000                 WAITING TASK QUEUE = 00000000
 USER MODE = XA                               COMPLETION TASK QUEUE = 00000000
 LOCK OWNER VMDBK ADDR = 00000000
 ACTIVE IORBK ADDR = 00000000                 VDEVAFLG = 20   VDEVIOP1 = 00
 PENDING INTERRUPT IORBK ADDR = 00000000      VDEVDFLG = 00
 START/RESUME PENDING IORBK ADDR = 00000000   VDEVSTAT = 00
 SENSE IORBK ADDR = 00000000                  VDEVWAIT = 00

 DEVICE DEPENDENT INFORMATION:
 VDSBK ADDR = 01C4A408                        VSPBK ADDR         = 01C4A448
 VPXBK ADDR = 00000000                        CURRENT SPABK ADDR = 00000000
                                              ACTIVE SPFBK ADDR  = 00000000

 NO ACTIVE IORBK
====>
```

Figure 73. Sample Output of a VIOBLOK Subcommand for a Spooling Device

For Figure 74, the subcommand entered is:

```
vio sch 04 operator
```

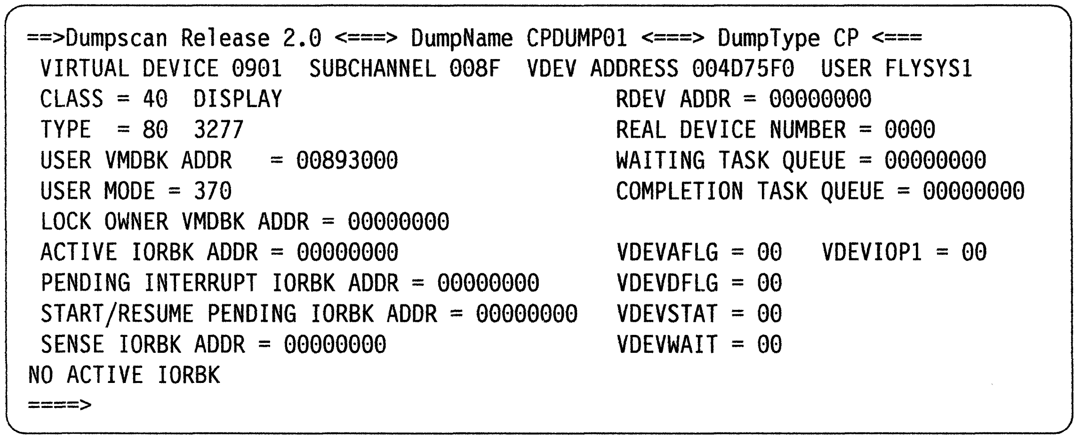


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 VIRTUAL DEVICE 0490  SUBCHANNEL 0004  VDEV ADDRESS 01C4A920  USER OPERATOR
 CLASS = 04  DASD                           RDEV ADDR = 0093C910
 TYPE  = 04  3380                           REAL DEVICE NUMBER = 0393
 USER VMDBK ADDR   = 01F2E000               WAITING TASK QUEUE = 00000000
 USER MODE = XA                             COMPLETION TASK QUEUE = 00000000
 LOCK OWNER VMDBK ADDR = 00000000
 ACTIVE IORBK ADDR = 00000000               VDEVAFLG = 00   VDEVIOP1 = 00
 PENDING INTERRUPT IORBK ADDR = 00000000    VDEVDFLG = 80
 START/RESUME PENDING IORBK ADDR = 00000000  VDEVSTAT = 00
 SENSE IORBK ADDR = 00000000                VDEVWAIT = 00

 DEVICE DEPENDENT INFORMATION:
 MINIDISK STARTING CYLINDER =      550      MDISK ADDR = 00000000
 MINIDISK ENDING   CYLINDER =      621

 NO ACTIVE IORBK
====>
```

Figure 74. Sample Output of a VIOBLOK Subcommand for a DASD Device

For Figure 75, the subcommand entered is:

```
vio 901 flysys1
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 VIRTUAL DEVICE 0901  SUBCHANNEL 008F  VDEV ADDRESS 004D75F0  USER FLYSYS1
 CLASS = 40  DISPLAY                        RDEV ADDR = 00000000
 TYPE  = 80  3277                           REAL DEVICE NUMBER = 0000
 USER VMDBK ADDR   = 00893000               WAITING TASK QUEUE = 00000000
 USER MODE = 370                            COMPLETION TASK QUEUE = 00000000
 LOCK OWNER VMDBK ADDR = 00000000
 ACTIVE IORBK ADDR = 00000000               VDEVAFLG = 00   VDEVIOP1 = 00
 PENDING INTERRUPT IORBK ADDR = 00000000    VDEVDFLG = 00
 START/RESUME PENDING IORBK ADDR = 00000000  VDEVSTAT = 00
 SENSE IORBK ADDR = 00000000                VDEVWAIT = 00
NO ACTIVE IORBK
====>
```

Figure 75. Sample Output of a VIOBLOK Subcommand for a Display Device

For Figure 76, the subcommand entered is:

```
vio 4F8 rscs
```

```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
 VIRTUAL DEVICE 04F8  SUBCHANNEL 0009  VDEV ADDRESS 01329860  USER RSCS
 CLASS = 02  SPECIAL                          RDEV ADDR = 00D94248
 TYPE  = 80  CTCA                             REAL DEVICE NUMBER = 088C
 USER VMDBK ADDR   = 01774000                 WAITING TASK QUEUE = 00000000
 USER MODE = 370                              COMPLETION TASK QUEUE = 00000000
 LOCK OWNER VMDBK ADDR = 00000000
 ACTIVE IORBK ADDR = 01F4E6C8                 VDEVAFLG = 08   VDEVIOP1 = 00
 PENDING INTERRUPT IORBK ADDR = 00000000      VDEVDFLG = 00
 START/RESUME PENDING IORBK ADDR = 00000000   VDEVSTAT = C0
 SENSE IORBK ADDR = 00000000                  VDEVWAIT = 00

 DEVICE DEPENDENT INFORMATION:
 X-SIDE CACBK ADDR = 00000000

 ACTIVE IORBK INFORMATION:
      FCTL ACTL DVST SCST COUNT  CCW ADDR           KEY  FPI  LPM  CPA
 IRB: 80404044040800000000001   01F400B0      ORB: 00   E0   80   01F400A8
                                              IORU: 00   28   FF
====>
```

Figure 76. Sample Output of a VIOBLOK Subcommand for a Device with an Active IORBK

VMDBK Subcommand

Use the VMDBK subcommand to scan the VMDBK chain and display summary information including the userid and VMDBK address for each user logged on to the system at the time of the dump. You can also use this subcommand to display selected fields from a specific user's VMDBK, or display a summary of a single user or VMDBK. The three forms of the VMDBK subcommand follow.

To provide a list of users in the dump, use the following command:

Vmdbk	[LIST]

To provide a summary of information for a specific user, use the following command:

Vmdbk	[**FOR**] *userid* **AT** *address*

To provide the values and descriptions of specific fields in the VMDBK for a specified user, use the following command:

Vmdbk	[[**FOR**] *userid* **AT** *address*] [**FIELD**] *field1* [*field2...*]

Where:

LIST
is an optional keyword operand. If you enter it, you see the userid, the VMDBK address, and the key fields of the VMDBK for each user on the global cyclic list. If the LIST operand is used, no other operands are allowed. This is the default if you do not enter any operands.

FOR userid
1 to 8 characters specifying the single userid for which the VMDBK data is desired. FOR is an optional keyword indicating that the following operand is the userid.

AT address
is the 31-bit (4-byte) hexadecimal address of the VMDBK that is to be viewed. The address must be on a 4K-byte boundary. It must also be preceded by the AT keyword operand.

FIELD field1
is a defined field name in the VMDBK that is to be displayed with a description. A list of fields separated by blanks may be entered (*field2 field3* ....). At least one field name must be entered if the FIELD operand is entered. FIELD is an optional keyword operand.

Usage Notes:

1. The VMDBK subcommand is not supported for virtual machine dumps.
2. You may omit leading zeros on the input VMDBK address.
3. If the userid is LIST, AT, FOR, FIELD, or if it begins with the characters VMD, the keyword FOR must be specified preceding the userid to distinguish it from a VMDBK field name.
4. When using the FIELD operand, there is no limit to the number of fields that may be specified other than the maximum length of the input buffer.
5. You need not specify a userid or address. The default userid is one of the following:
   - The virtual machine whose VMDBK was last requested by either the VMDBK or VIOBLOK subcommands during the current DUMPSCAN session.
   - The system VMDBK, if this is the first time the VMDBK subcommand has been issued and the VIOBLOK subcommand has not been issued.
6. Once you enter the VMDBK subcommand with the userid or address specified, you may continue to display fields from that VMDBK by entering the desired field names. The subcommand name, VMDBK, and the userid or address fields may be omitted after they are initially entered.

   For example, if you enter:

   ```
   vmd for user1 field vmdmode
   ```

   and then wish to view the VMDBK type, enter:

   ```
   vmdtype
   ```

   The output shows the VMDBK type for USER1.
7. When you specify no operands, the LIST operand is assumed and the output consists of the userid, the VMDBK address, and some key fields for each VMDBK in the dump.
8. The following fields are displayed if the default LIST operand is used.

Field	**Length**	**Description**
VMDUSER	8	User logon identification
VMDMODE	1	Guest machine mode controls
VMDSTYPE	2	Supplement of the VMDMODE field
VMDSLIST	1	Scheduling list definition
VMDSTATE	1	Scheduler/dispatcher state
VMDICODE	1	Interception event code
VMDCOMND	8	The last CP command executed
VMDGSTAT	1	The guest virtual running status
VMDWSTAT	1	Pseudo-wait conditions
VMDOSTAT	1	Virtual machine operating status
VMDRSTAT	1	Running blockage status
VMDCFCTL	1	Console function controls
VMDCFLAG	1	Console function status flags.

   The VMDBK address appears in the leftmost column of the output.

9. The following fields are displayed if a userid or a VMDBK address is used.

Field	Length	Description
VMDUSER	8	User logon identification
VMDMODE	1	Guest machine mode controls
VMDSTYPE	2	A supplement of the VMDMODE field
VMDPSW	8	The guest PSW
VMDTYPE	1	The VMDBK type
VMDCOMND	8	The last CP command executed
VMDCFLAG	1	The console function status flag
VMDCFCTL	1	Console functions controls
VMDPCL	4	Authorized privilege classes
VMDINST	6	An intercepted instruction
VMDICODE	1	The interception event code
VMDCPUDS	2	The host CPU address on which the user was last dispatched
VMDSLIST	1	A scheduling list definition
VMDSTATE	1	A scheduler/dispatcher state
VMDIOACT	4	The number of I/O operations outstanding
VMDWVDEV	4	The address of VDEV for the status response
VMDWRKLC	4	Local work bits
VMDWRKCS	4	The compare-and-swap work bits field
VMDGSTAT	1	The guest virtual running status
VMDRSTAT	1	The running blockage status
VMDGPRS	64	Guest general purpose registers
VMDCRS	64	Guest control registers 0-15.

10. If you specify a userid for which more than one VMDBK is found, the VMDBK subcommand terminates with message 624I (USERID userid IS A DUPLICATE USERID). You may still display VMDBK information about the desired userid by using the following procedure:

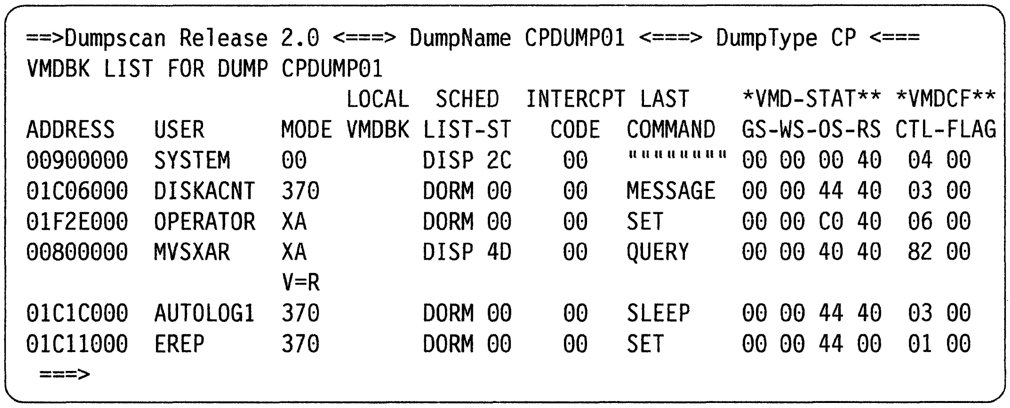
    a. Issue the VMDBK LIST subcommand to determine the address of the desired VMDBK.

    b. Once you know the address of the VMDBK for the required userid, issue the VMDBK AT address subcommand.

Sample Output:

Figure 77 on page 149 illustrates the output you receive when you enter the VMDBK subcommand using the default LIST operand. In the dump, you receive a list of all of the users along with their VMDBK address and some key fields. The subcommand entered is:

```
vmdbk or vmdbk list
```

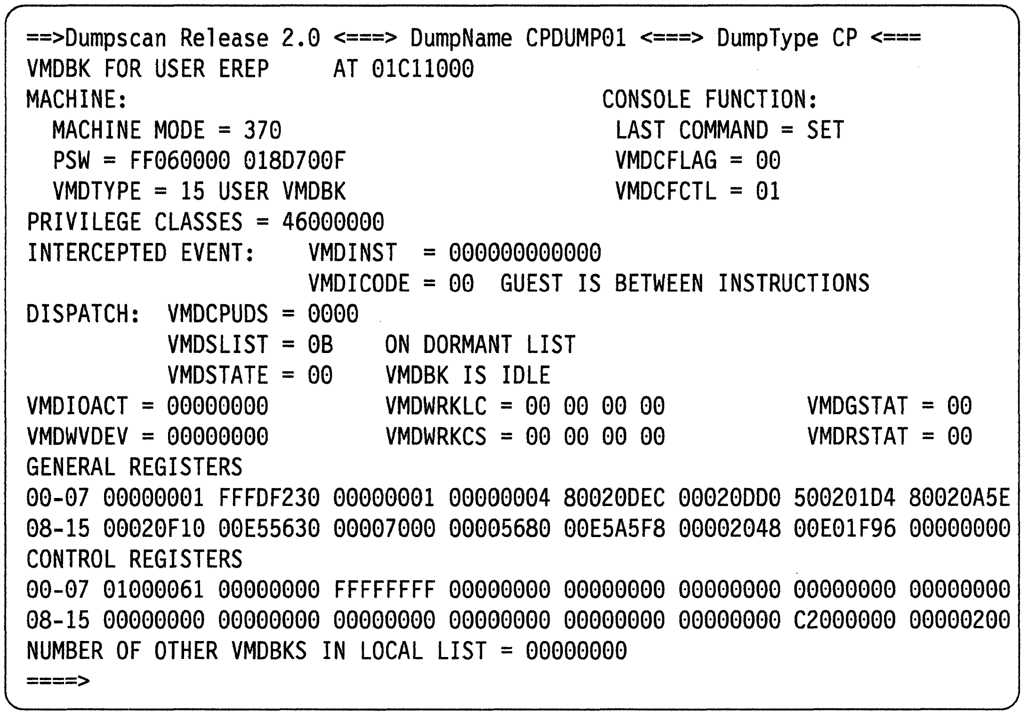


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
VMDBK LIST FOR DUMP CPDUMP01
                         LOCAL  SCHED  INTERCPT LAST     *VMD-STAT** *VMDCF**
ADDRESS   USER      MODE VMDBK LIST-ST   CODE  COMMAND  GS-WS-OS-RS CTL-FLAG
00900000  SYSTEM    00         DISP 2C    00   """""""" 00 00 00 40  04 00
01C06000  DISKACNT  370        DORM 00    00   MESSAGE  00 00 44 40  03 00
01F2E000  OPERATOR  XA         DORM 00    00   SET      00 00 C0 40  06 00
00800000  MVSXAR    XA         DISP 4D    00   QUERY    00 00 40 40  82 00
                    V=R
01C1C000  AUTOLOG1  370        DORM 00    00   SLEEP    00 00 44 40  03 00
01C11000  EREP      370        DORM 00    00   SET      00 00 44 00  01 00
 ===>
```

Figure 77. Sample Output of a VMDBK Subcommand with the LIST Operand

Figure 78 illustrates the output you receive using the VMDBK subcommand with a userid or an address specified. The subcommand entered is:

```
vmdbk for erep or vmdbk at 01c11000
```

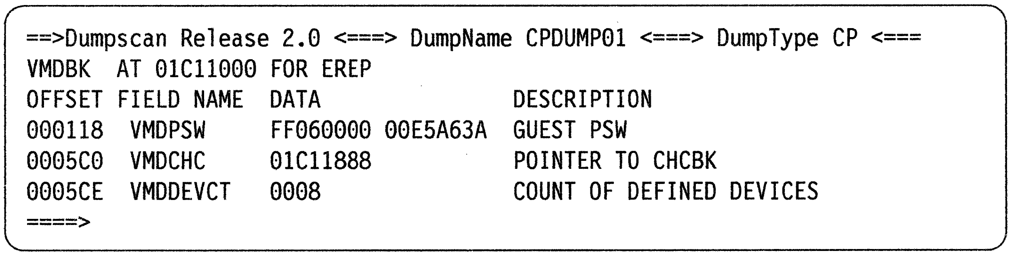


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
VMDBK FOR USER EREP     AT 01C11000
MACHINE:                                     CONSOLE FUNCTION:
  MACHINE MODE = 370                          LAST COMMAND = SET
  PSW = FF060000 018D700F                     VMDCFLAG = 00
  VMDTYPE = 15 USER VMDBK                     VMDCFCTL = 01
PRIVILEGE CLASSES = 46000000
INTERCEPTED EVENT:    VMDINST  = 000000000000
                      VMDICODE = 00  GUEST IS BETWEEN INSTRUCTIONS
DISPATCH:  VMDCPUDS = 0000
           VMDSLIST = 0B    ON DORMANT LIST
           VMDSTATE = 00    VMDBK IS IDLE
VMDIOACT = 00000000         VMDWRKLC = 00 00 00 00          VMDGSTAT = 00
VMDWVDEV = 00000000         VMDWRKCS = 00 00 00 00          VMDRSTAT = 00
GENERAL REGISTERS
00-07 00000001 FFFDF230 00000001 00000004 80020DEC 00020DD0 500201D4 80020A5E
08-15 00020F10 00E55630 00007000 00005680 00E5A5F8 00002048 00E01F96 00000000
CONTROL REGISTERS
00-07 01000061 00000000 FFFFFFFF 00000000 00000000 00000000 00000000 00000000
08-15 00000000 00000000 00000000 00000000 00000000 00000000 C2000000 00000200
NUMBER OF OTHER VMDBKS IN LOCAL LIST = 00000000
====>
```

Figure 78. Sample Output of a VMDBK Subcommand with a Userid or an Address Specified

Figure 79 is an example of the output you receive when you specify a userid plus specific VMDBK fields. The subcommand entered is:

```
vmd for erep field vmdpsw vmdchc vmddevct
```

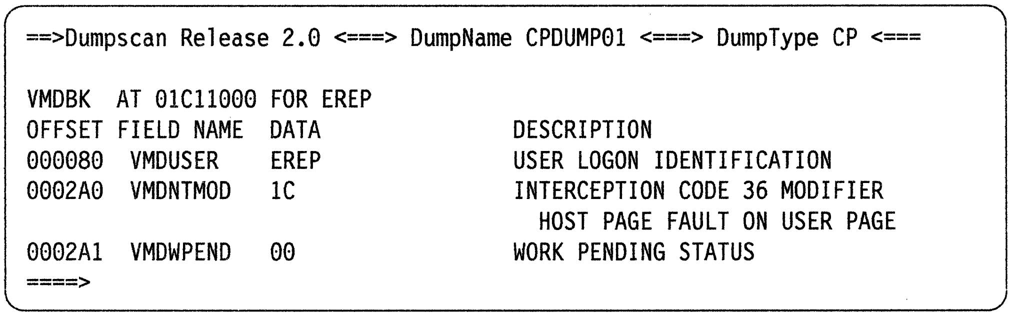


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
VMDBK  AT 01C11000 FOR EREP
OFFSET FIELD NAME  DATA               DESCRIPTION
000118  VMDPSW     FF060000 00E5A63A  GUEST PSW
0005C0  VMDCHC     01C11888           POINTER TO CHCBK
0005CE  VMDDEVCT   0008               COUNT OF DEFINED DEVICES
====>
```

Figure 79. Sample Output of a VMDBK Subcommand with a Userid and VMDBK Fields Specified

Figure 80 is an example of the output you receive when you specify an address plus specific VMDBK fields. The subcommand entered is:

```
vmd at 1c11000 field vmduser vmdntmod vmdwpend
```

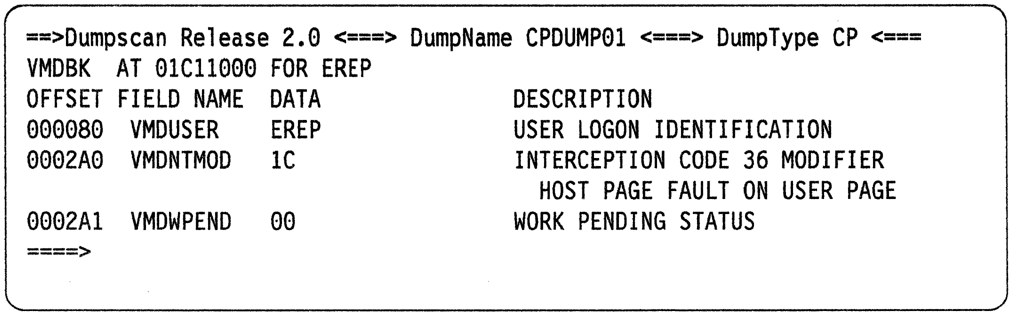


```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===

VMDBK  AT 01C11000 FOR EREP
OFFSET FIELD NAME  DATA               DESCRIPTION
000080  VMDUSER    EREP               USER LOGON IDENTIFICATION
0002A0  VMDNTMOD   1C                 INTERCEPTION CODE 36 MODIFIER
                                        HOST PAGE FAULT ON USER PAGE
0002A1  VMDWPEND   00                 WORK PENDING STATUS
====>
```

Figure 80. Sample Output of a VMDBK Subcommand with an Address and VMDBK Fields Specified

Figure 81 is an example of the VMDBK subcommand with only VMDBK fields specified. The VMDBK information displayed is for the same VMDBK specified in the previous issuance of the VMDBK subcommand. The subcommand entered is:

```
vmdbk vmduser vmdntmod vmdwpend
```



```
==>Dumpscan Release 2.0 <===> DumpName CPDUMP01 <===> DumpType CP <===
VMDBK  AT 01C11000 FOR EREP
OFFSET FIELD NAME  DATA               DESCRIPTION
000080  VMDUSER    EREP               USER LOGON IDENTIFICATION
0002A0  VMDNTMOD   1C                 INTERCEPTION CODE 36 MODIFIER
                                        HOST PAGE FAULT ON USER PAGE
0002A1  VMDWPEND   00                 WORK PENDING STATUS
====>
```

Figure 81. Sample Output of a VMDBK Subcommand with Only VMDBK Fields Specified

VPAIR Subcommand

Use the VPAIR subcommand to view the contents of a specific even-odd pair of vector registers or a designated number of vector elements if that information is available in a VM dump. The format of the contents displayed is compatible with that from CP's DISPLAY command.

The format of the VPair subcommand is as follows:

```
VPair   [ reg1 [ -  [ reg2   ] ] ]  [ ,elt1 [ -  [ elt2   ] ] ]  [ CPU xx ]
          0    [ :  [ END    ] ]            [ :  [ END    ] ]
               [                ]           [                ]
               [ {.}[ regcnt ]  ]           [ {.}[ eltcnt ]  ]
               [    [ END    ]  ]           [    [ END    ]  ]
```

Where:

reg1
specifies the vector register pair whose contents are to be displayed. *reg1* may be either an even decimal number from 0 to 14 or an even hexadecimal number from 0 to E. The number specified represents the first or only vector register pair whose contents are to be displayed. When specifying the register pair, use the number of the even register in the even-odd pair. Register 0 is the default.

- *or* **:**
indicates that a range of vector register pairs is to be displayed.

reg2
represents the last vector register pair whose contents are to be displayed. *reg2* may be either an even decimal number or an even hexadecimal number from 0 to E. *reg2* must be equal to or greater than *reg1*. *reg2* must also be the even register of an even-odd pair.

END
is the default value for *reg2*. If you select END, or if it serves as the default, all register pairs from the pair specified by *reg1* to register pair 14-15 are displayed.

indicates that a count of vector register pairs is to be displayed.

regcnt
specifies the number of vector register pairs whose contents are to be displayed. *regcnt* is a number from 1 to 8. The range indicated by *reg1* and *regcnt* cannot exceed the last valid register pair (that is, registers 14-15).

END
is the default value for *regcnt*. If you select END, or if it serves as the default, the contents of all register pairs from the pair specified by *reg1* to register pair 14-15 are displayed.

:

indicates that a register element specification follows. If you omit both the comma and element specification, the contents of element 0 are displayed.

elt1

represents the first or only vector element of a register pair to be displayed. *elt1* is a hexadecimal number from 0 to one less than the section size.

- *or* **:**

indicates that a range of vector elements is to be displayed.

elt2

represents the last element of a register pair whose contents are to be displayed. *elt2* is a hexadecimal number from 0 to one less than the section size. *elt2* must be equal to or greater than *elt1*.

END

is the default value for *elt2*. If you select END, or if it serves as the default, all vector elements from the element specified by *elt1* to the last element of the register pair designated are displayed.

indicates that a count of vector elements is to be displayed.

eltcnt

indicates the number of elements whose contents are to be displayed. *eltcnt* is a hexadecimal number from 1 to the section size. The range indicated by *elt1* and *eltcnt* cannot exceed the maximum number of elements.

END

is the default value for *eltcnt*. It calls for the display of the contents of all vector elements of the register pair, starting with the element specified by *elt1* and including the last element of the register pair.

CPU xx

is the address of the processor for which the vector register pair information is to be displayed. *xx* is a hexadecimal number between 0 and 3F. If you do not specify an address, the requested information is displayed for the processor on which the dump was initiated.

Usage Notes:

1. When you specify ranges for vector register pairs or elements of a vector register pair, the command cannot contain embedded blanks. For example, the following range specifications are invalid:

   ```
   VP 0 - 8
   ```

   or

   ```
   VP 0-8, 1 - 200
   ```

   The following range specifications are valid:

   ```
   VP 0-8
   ```

   or

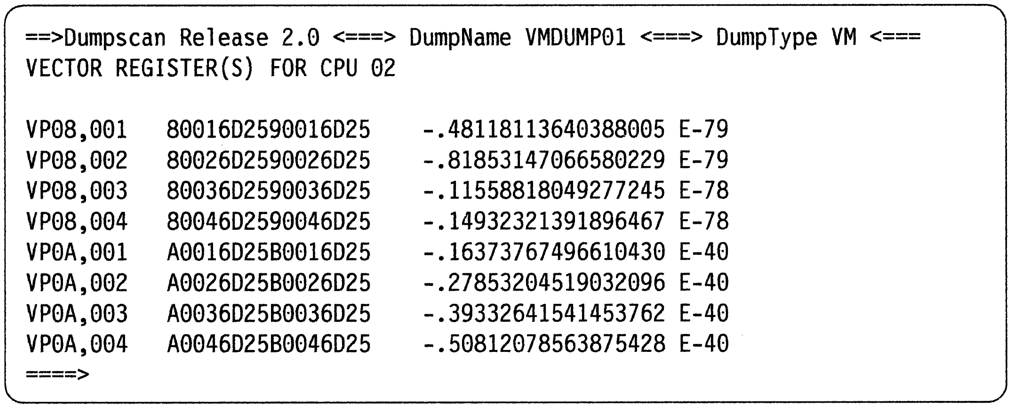
   ```
   VP 0-8,1-200
   ```

2. You can use the dump viewing facility's CPU subcommand to find the address of each processor contained in the dump.

Example:

Figure 82 shows the screen output you would receive if you entered the VPAIR command. The subcommand entered is:

```
vpair 8-a,1-4
```



```
==>Dumpscan Release 2.0 <===> DumpName VMDUMP01 <===> DumpType VM <===
VECTOR REGISTER(S) FOR CPU 02

VP08,001   80016D2590016D25    -.48118113640388005 E-79
VP08,002   80026D2590026D25    -.81853147066580229 E-79
VP08,003   80036D2590036D25    -.11558818049277245 E-78
VP08,004   80046D2590046D25    -.14932321391896467 E-78
VP0A,001   A0016D25B0016D25    -.16373767496610430 E-40
VP0A,002   A0026D25B0026D25    -.27853204519032096 E-40
VP0A,003   A0036D25B0036D25    -.39332641541453762 E-40
VP0A,004   A0046D25B0046D25    -.50812078563875428 E-40
====>
```

Figure 82. Output from a VPAIR Subcommand

VREG Subcommand

Use the VREG subcommand to view the contents of a specific vector register or a designated number of vector elements if that information is available in a VM dump. The format of the contents displayed is compatible with that from CP's DISPLAY command.

The format of the VReg subcommand is as follows:

```
VReg   ┌ ┌                      ┐ ┐ ┌ ┌                      ┐ ┐  ┌        ┐
       │ │reg1  -  ┌ reg2 ┐     │ │ │ │,elt1  -  ┌ elt2 ┐  │ │  │ CPU xx │
       │ │ 0    :  └ END  ┘     │ │ │ │       :  └ END  ┘  │ │  └        ┘
       │ │                      │ │ │ │                    │ │
       │ │     {.} ┌ regcnt ┐   │ │ │ │     {.} ┌ eltcnt ┐ │ │
       │ │         └ END    ┘   │ │ │ │         └ END    ┘ │ │
       │ └                      ┘ │ │ └                    ┘ │
       └                          ┘ └                        ┘
```

Where:

reg1
specifies the vector register whose contents are to be displayed. *reg1* may be either a decimal number from 0 to 15 or a hexadecimal number from 0 to F. Register 0 is the default.

- *or* **:**
indicates that a range of vector registers is to be displayed.

reg2
represents the last vector register whose contents are to be displayed. *reg2* may be either a decimal number from 0 to 15 or a hexadecimal number from 0 to E. *reg2* must be equal to or greater than *reg1.*

END
is the default value for *reg2.* If you select END, or if it serves as the default, the contents of all registers from the register specified by *reg1* to register 15 are displayed.

indicates you want a count of registers to be displayed.

regcnt
specifies the number of vector registers whose contents are to be displayed. *regcnt* may be either a decimal number from 1 to 16 or a hexadecimal number from 1 to F. The range indicated by *reg1* and *regcnt* cannot exceed register 15.

END
is the default value for *regcnt.* If you select END, or if it serves as the default, the contents of all registers from the register specified by *reg1* to register 15 are displayed.

:

indicates that a register element specification follows. If you omit both the comma and element specification, the contents of elements 0-3 are displayed.

elt1

represents the first or only element to be displayed. *elt1* is a hexadecimal number from 0 to one less than the section size.

- *or* **:**

indicates that a range of vector elements is to be displayed.

elt2

represents the last element of a register whose contents are to be displayed. *elt2* is a hexadecimal number from 0 to one less than the section size. *elt2* must be equal to or greater that *elt1*.

END

is the default value for *elt2*. If you select END, or if it serves as the default, all vector elements from the element specified by *elt1* to the last element of the register designated are displayed.

indicates you want a count of elements to be displayed.

eltcnt

indicates the number of elements whose contents are to be displayed. *eltcnt* is a hexadecimal number from 1 to the section size. The range indicated by *elt1* and *eltcnt* cannot exceed the maximum number of elements.

END

is the default value for *eltcnt*. It calls for the display of the contents of all register elements from the element specified by *elt1* to the last element of the register.

CPU xx

is the address of the processor for which the vector register information is to be displayed. xx is a hexadecimal number between 0 and 3F. If you do not specify an address, the requested information is displayed for the processor on which the dump was initiated.

Usage Notes:

1. When you specify ranges for vector registers or elements of a vector register, the command cannot contain embedded blanks. For example, the following register specifications are invalid:

   ```
   VR 1 - 13
   ```

   or

   ```
   VR 1-13, 3 - 100
   ```

   The following register specifications are valid:

   ```
   VR 1-13
   ```

   or

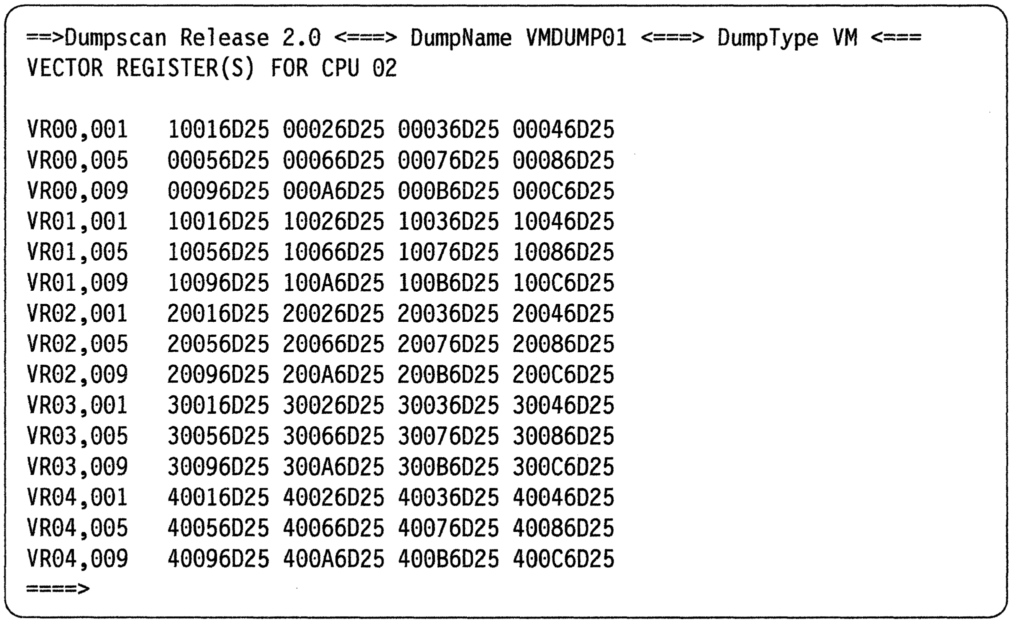
   ```
   VR 1-13,3-100
   ```

2. You can use the dump viewing facility's CPU subcommand to find the address of each processor contained in the dump.

Example:

Figure 83 shows the screen output you would receive if you entered the VREG command. The subcommand entered is:

```
vreg 0-4,1-c
```



```
==>Dumpscan Release 2.0 <===> DumpName VMDUMP01 <===> DumpType VM <===
VECTOR REGISTER(S) FOR CPU 02

VR00,001   10016D25 00026D25 00036D25 00046D25
VR00,005   00056D25 00066D25 00076D25 00086D25
VR00,009   00096D25 000A6D25 000B6D25 000C6D25
VR01,001   10016D25 10026D25 10036D25 10046D25
VR01,005   10056D25 10066D25 10076D25 10086D25
VR01,009   10096D25 100A6D25 100B6D25 100C6D25
VR02,001   20016D25 20026D25 20036D25 20046D25
VR02,005   20056D25 20066D25 20076D25 20086D25
VR02,009   20096D25 200A6D25 200B6D25 200C6D25
VR03,001   30016D25 30026D25 30036D25 30046D25
VR03,005   30056D25 30066D25 30076D25 30086D25
VR03,009   30096D25 300A6D25 300B6D25 300C6D25
VR04,001   40016D25 40026D25 40036D25 40046D25
VR04,005   40056D25 40066D25 40076D25 40086D25
VR04,009   40096D25 400A6D25 400B6D25 400C6D25
====>
```

Figure 83. Sample Output from a VREG Subcommand

VSTAT Subcommand

Use the VSTAT subcommand of DUMPSCAN to learn status information about the Vector Facility for any processor in a VM dump. Should no Vector Facility be available for the virtual machine or the processor requested, you receive an informational message.

VStat	[**CPU** *xx*]

Where:

CPU xx
specifies the processor address for which you want to have vector status information displayed. The processor address (xx) may be a hexadecimal number between 0 and 3F. If you do not specify the processor number, the vector status is displayed for the processor on which the dump was initiated.

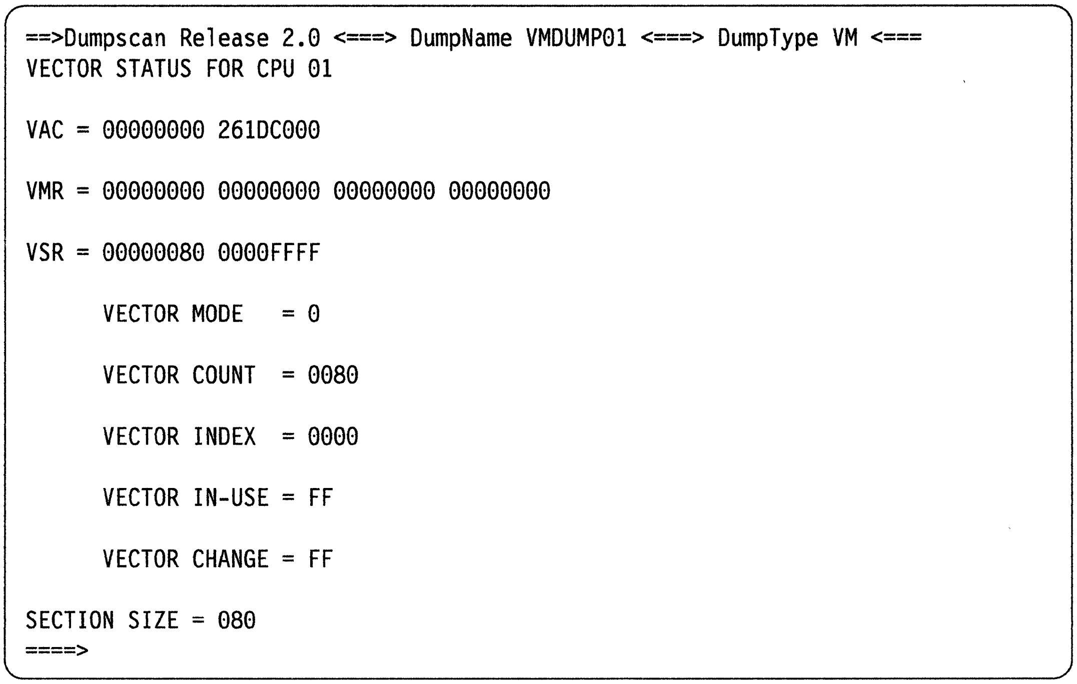
Usage Note:

You can use the CPU subcommand of DUMPSCAN to obtain the address of each processor in the dump.

Example:

Figure 84 shows the screen output you would receive if you entered the VSTAT command. The subcommand entered is:

```
vstat cpu 1
```



```
==>Dumpscan Release 2.0 <===> DumpName VMDUMP01 <===> DumpType VM <===
VECTOR STATUS FOR CPU 01

VAC = 00000000 261DC000

VMR = 00000000 00000000 00000000 00000000

VSR = 00000080 0000FFFF

      VECTOR MODE   = 0

      VECTOR COUNT  = 0080

      VECTOR INDEX  = 0000

      VECTOR IN-USE = FF

      VECTOR CHANGE = FF

SECTION SIZE = 080
====>
```

Figure 84. Sample Output from a VSTAT Subcommand

XEDIT Subcommand

Use the XEDIT subcommand to force the dump viewing facility to pass the commandline to XEDIT for execution.

The format of the CMS subcommand is as follows:

Xedit	[*commandline*]

Where:

commandline
is any valid XEDIT subcommand or macro and its operands.

Usage Note:

Any XEDIT subcommand should be prefaced with XEDIT to prevent the dump viewing facility from processing the subcommand.

Chapter 5. Messages and Message Summaries

This chapter contains all of the messages that the dump viewing facility will issue. This chapter is organized as follows:

- Message format
- Message explanations
- Message summary by message number.

Dump Viewing Facility Message Format

A dump viewing facility message contains a message identifier and message text. The message identifier uniquely identifies each message. The message text states an error or condition that has occurred, or it may request a user response. The format for dump viewing facility messages appears below.

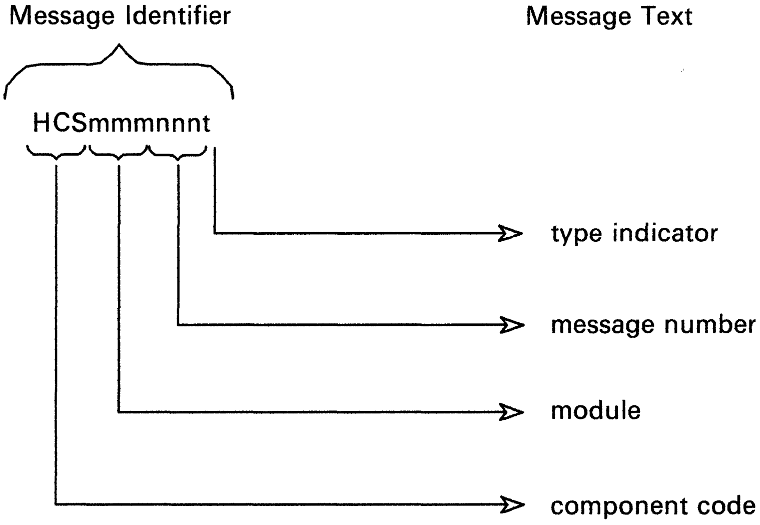


Figure 85. Message Format for Dump Viewing Facility Messages

The characters within the message identifier indicate the following:

HCS The component code for all messages issued in the dump viewing facility.
mmm The name of the module that issued the message.
nnn The three-digit message number.
t The letter code designating the type of message.
It may be one of three letters — A, E, or I.

- *A*—the user should take immediate action, which may be simply responding to a prompt.
- *E*—an error has occurred.
- *I*—the message is informational only.

How Messages Are Displayed

Messages can be displayed in three ways, depending on how you issue the CP command SET EMSG. The choices and their results are:

Command	**Display**
SET EMSG TEXT	Message text
SET EMSG CODE	HCSmmmnnnt
SET EMSG ON	HCSmmmnnnt message text

Note: For a more detailed description of the SET EMSG command, refer to the *VM/XA SP: CP Command Reference* manual.

The dump viewing facility messages are listed numerically. The message description includes:

Message number — A unique three-digit number and type indicator for each message.

Message text — A short sentence that describes a condition that has occurred or requests a response from the user.

System action — The action of the system after the condition was detected.

User response — The action the user takes after receiving a message.

Commands that detect condition — A list of commands and subcommands that detect an error condition.

The dump viewing facility messages are described with the following conventions:

Lowercase letters — These letters represent variables the system will replace with specific values when the message is displayed.

{...|...} — Within the braces, a bar separates the choices the system must make to complete the information in the message text.

Dump Viewing Facility Messages

010A **ENTER TRACE ENTRY SELECTION CRITERIA, NULL LINE TO END SELECTION, OR QUIT TO END TRACERED COMMAND**

Explanation: You are being prompted for trace selection criteria to be used in TRACERED command processing. If you enter a null line, *no* selection criteria are applied. As a result, everything will be traced.

If you enter selection criteria, you will continue to receive this prompt until you enter a null line, thus indicating that entry of selection criteria is complete.

For TRSOURCE-created data, the only applicable selection criteria is time. Prompting continues until you enter a valid time value, a null line, or QUIT.

System Action: The system waits for a response to the prompt.

User Response: Respond to the prompt with one of the following replies:

1. Any combination of the following selection type keywords and values:

   CODE xxxx xxxx . . .
   You may enter up to 16 four-digit hexadecimal values representing the trace codes to be selected during data reduction of the input tape or system trace file. See the *VM/XA SP: CP Command Reference* for a list of valid trace codes.

   Note: Leading zeros (0) are required.

   The last two digits of any code may be specified as the character string ** or *. This indicates that you want a range of trace codes. For instance, 15** would select all trace entries beginning with the hexadecimal digits 15. CODE is valid only for CPTRACE data.

   VMDBK xxxxxxxx xxxxxxxx . . .
   You may enter up to 16 hexadecimal values that represent the VMDBK addresses to be used during data reduction of the input tape or system trace file. Leading zeros are not required. Each address specified must be in the range of hex 0 to 7FFFF000 and on a page (4K-byte) boundary.

   RDEV xxxx xxxx . . .
   You may specify up to 16 hexadecimal values that represent the RDEV numbers to be used during data reduction of the input tape or system trace file. Leading zeros are not required. Each device number specified must be in the range of hex 0 to FFFF. RDEV is valid only for CPTRACE data.

   VDEV xxxx xxxx . . .
   You may specify up to 16 hexadecimal values that represent the VDEV numbers to be used during data reduction of the input tape or system trace file. Leading zeros are not required. Each device number specified must be in the range of hex 0 to FFFF. VDEV is valid only for CPTRACE data.

CPU xxxx xxxx . . .
You may specify up to 16 hexadecimal values that represent the processor addresses to be used during data reduction of the input tape or system trace file. Leading zeros are not required. Each processor address specified must be in the range of hex 0 to 3F. CPU is valid only for CPTRACE data.

TIME mm/dd/yy hh:mm:ss mm/dd/yy hh:mm:ss
You may designate a start and stop date/time for the data reduction of the trace entries. If this option is selected, only trace entries created within the designated time range will be processed. You may select only one time range. All TIME parameters can be entered on the same line. TIME is valid for both guest trace and CP entries.

2. CMS - To enter the CMS subset mode.
3. HELP - To invoke the help panel for the TRACERED command.
4. QUIT - To terminate the TRACERED command.
5. A null line - To indicate that you are done entering selection criteria and want to end prompting.

   If your immediate reply to the prompt is a null line, then all trace entries are selected.

See page 48 for usage notes for entering selection criteria.

Commands That Detect Condition:

```
TRACERED
```

015A

WILL YOU BE MOUNTING ANOTHER TRSAVE TAPE? ENTER YES OR NO

Explanation: TRACERED encountered the end-of-volume on the current trace input tape. If you enter “yes,” TRACERED will rewind and unload the current tape and wait for the next tape mount. If you enter “no,” TRACERED will complete processing.

System Action: The system waits for a reply to the prompt.

User Response: Reply “yes” if you want additional trace data to be processed. Reply “no” if you do not.

Commands That Detect Condition:

```
TRACERED
```

020E

‘operand1’ AND ‘operand2’ ARE CONFLICTING KEYWORD OPERANDS

Explanation: You entered conflicting operands (designated by “operand1” and “operand2” in the error message) in the TRACERED command.

System Action: The TRACERED command terminates.

User Response: Specify operands that do not conflict and reenter the TRACERED command.

Commands That Detect Condition:

```
TRACERED
```

021E

TAPE DEVICE ADDRESS INVALID

Explanation: The tape device address you entered in the TRACERED command is invalid. Only virtual tape addresses 181 through 184 are supported.

System Action: The TRACERED command terminates.

User Response: Reenter the TRACERED command specifying a tape device within the 181-184 range.

Commands That Detect Condition:

TRACERED

030I

PROCESSING TRSAVE TAPE - VOL SEQ # nn - CREATED mm/dd/yy hh:mm:ss

Explanation: This informational message indicates the TRACERED command has successfully read the volume label of the current TRSAVE input trace tape. The volume sequence number ("nn" in the message) and date the tape was created ("mm/dd/yy") are identified.

System Action: TRACERED processes trace entries from the specified tape until encountering the end-of-volume.

User Response: A message, 015A (see page 162), appears when end-of-volume is reached. You have the options of either ending processing or continuing with additional tape volumes.

Commands That Detect Condition:

TRACERED

031I

END OF TAPE ENCOUNTERED

Explanation: TRACERED has encountered the end-of-volume condition on the current input tape.

System Action: The input tape is rewound.

User Response: None

Commands That Detect Condition:

TRACERED

032I

WAITING FOR TAPE MOUNT

Explanation: You replied "yes" to prompt message 015A which asked you if you wanted a new TRSAVE tape to be mounted. As a result, TRACERED is waiting for the tape to be mounted.

System Action: Once the input tape is mounted, TRACERED will begin processing it.

User Response: Instruct the operator to mount the desired TRSAVE tape.

Commands That Detect Condition:

```
TRACERED
```

040E

TIME SELECTION RANGE MAY BE SPECIFIED ONLY ONCE

Explanation: You entered more than one time selection range in response to the prompt for trace selection criteria. Although you may specify selection criteria on many different lines in response to the prompt, you may specify *only one* time selection range for all the selection replies combined for one TRACERED command.

System Action: All of the selection criteria you entered in response to previous prompts remain in effect. The selection criteria you specified in the most recent response are ignored. You will be reprompted for additional trace selection criteria.

User Response: If you wish to enter additional values for trace selection criteria, do so when prompted. Or, if you wish to end the selection process, enter a null line. You may stop TRACERED processing altogether by entering QUIT.

Commands That Detect Condition:

```
TRACERED
```

041E

A MAXIMUM OF SIXTEEN CPU ADDRESSES MAY BE SPECIFIED

Explanation:

1. You entered more than 16 CPU address values in response to the prompt(s) for trace selection criteria. Although you may specify selection criteria on many different lines in response to the prompt(s), the total number of CPU addresses in all of your responses must not exceed 16 for one TRACERED command.
2. You entered more than 16 CPU address values under the SELECT subcommand of DUMPSCAN. Although you may specify selection criteria on many different invocations of SELECT, the total number of CPU address values in all of your requests must not exceed 16 for one invocation of the TRACE subcommand of DUMPSCAN.

System Action:

1. All of the selection criteria you entered in response to previous prompts remain in effect. The selection criteria you specified in the most recent response are ignored. You will be reprompted for additional trace selection criteria on the TRACERED command.
2. All of the selection criteria you entered previously remain in effect. The selection criteria you specified in the most recent response are ignored.

User Response:

1. When reprompted for additional trace selection criteria, enter the additional values. Or, if you want to end the selection process, enter a null line.

2. You may reenter other selection criteria or another subcommand or quit DUMPSCAN processing.

Commands That Detect Condition:

1. `TRACERED`

2. `SELECT subcommand of DUMPSCAN`

042E **A MAXIMUM OF SIXTEEN TRACE CODES MAY BE SPECIFIED**

Explanation:

1. You entered more than 16 trace code values in response to the prompt(s) for trace selection criteria. Although you may specify selection criteria on many different lines in response to the prompt(s), the total number of trace code values in all of your responses must not exceed 16 for one TRACERED command.

2. You entered more than 16 trace code values under the SELECT subcommand of DUMPSCAN. Although you may specify selection criteria on many different invocations of SELECT, the total number of trace code values in all your requests must not exceed 16 for one invocation of the TRACE subcommand of DUMPSCAN.

System Action:

1. All of the selection criteria you entered in response to previous prompts remain in effect. The selection criteria you specified in the most recent response is ignored. You will be reprompted for additional trace selection criteria.

2. All the selection criteria you entered previously remain in effect. The selection criteria you specified in the most recent request is ignored.

User Response:

1. When reprompted for additional trace selection criteria, enter the additional values. Or, if you want to end the selection process, enter a null line.

2. You may reenter other selection criteria on the SELECT subcommand of DUMPSCAN, or another subcommand, or you may quit DUMPSCAN processing.

Commands That Detect Condition:

1. `TRACERED`

2. `SELECT subcommand of DUMPSCAN`

043E

A MAXIMUM OF SIXTEEN VMDBK ADDRESSES MAY BE SPECIFIED

Explanation:

1. You entered more than 16 VMDBK address values in response to the prompts for trace selection criteria on the TRACERED command. Although you may specify selection criteria on many different lines in response to the prompts, the total number of VMDBK address values in all of your responses must not exceed 16 for one TRACERED command.
2. You entered more than 16 VMDBK address values under the SELECT subcommand of DUMPSCAN. Although you may specify selection criteria on many different invocations of SELECT, the total number of VMDBK address values in all of your requests must not exceed 16 for one invocation of the TRACE subcommand of DUMPSCAN.

System Action:

1. All of the selection criteria you entered in response to previous prompts remain in effect. The selection criteria you specified in the most recent response is ignored. You will be reprompted on the TRACERED command for additional trace selection criteria.
2. All the selection criteria you entered previously remain in effect. The selection criteria you specified in the most recent request is ignored.

User Response:

1. When reprompted for additional trace selection criteria on the TRACERED command, enter the additional values. Or, if you wish to end the selection process, enter a null line.
2. You may reenter selection criteria or another subcommand, or you can quit DUMPSCAN processing.

Commands That Detect Condition:

1. `TRACERED`
2. `SELECT subcommand of DUMPSCAN`

044E

A MAXIMUM OF SIXTEEN RDEV NUMBERS MAY BE SPECIFIED

Explanation:

1. You entered more than 16 RDEV values in response to the prompts for trace selection criteria. Although you may specify selection criteria on many different lines in response to the prompts, the total number of RDEV values in all of your responses must not exceed 16 for one TRACERED command
2. You entered more than 16 RDEV values under the SELECT subcommand of DUMPSCAN. Although you may specify selection criteria on many different invocations of SELECT, the total number of RDEV values in all of your requests must not exceed 16 for one invocation of the TRACE subcommand of DUMPSCAN.

System Action:

1. All of the selection criteria you entered in response to previous prompts remain in effect. The selection criteria you specified in the most recent response are ignored. You will be reprompted for additional trace selection critera.
2. All of the selection criteria you entered previously remain in effect. The selection criteria you specified in the most recent request is ignored.

User Response:

1. When reprompted for additional trace selection criteria, enter the additional values. Or, if you want to end the selection process, enter a null line.
2. You can reenter other selection criteria or another subcommand, or you can quit DUMPSCAN processing.

Commands That Detect Condition:

```
1. TRACERED
2. SELECT subcommand of DUMPSCAN
```

045E

A MAXIMUM OF SIXTEEN VDEV NUMBERS MAY BE SPECIFIED

Explanation:

1. You entered more than 16 VDEV values in response to the prompts for trace selection criteria. Although you may specify selection criteria on many different lines in response to the prompt, the total number of VDEV values in all of your responses must not exceed 16 for one TRACERED command.
2. You entered more than 16 VDEV values under the SELECT subcommand of DUMPSCAN. Although you may specify selection criteria on many different invocations of SELECT, the total number of VDEV values in all your requests must not exceed 16 for one invocation of the TRACE subcommand of DUMPSCAN.

System Action:

1. All of the selection criteria you entered in response to previous prompts remain in effect. The selection criteria you specified in the most recent response are ignored. You will be reprompted for additional trace selection criteria.
2. All of the selection criteria you entered previously remain in effect. The selection criteria you specified in the most recent request is ignored.

User Response:

1. When reprompted for additional trace selection criteria, enter the additional values. Or, if you want to end the selection process, enter a null line.
2. You can reenter other selection criteria or another subcommand or you can quit DUMPSCAN processing.

Commands That Detect Condition:

1. TRACERED
2. SELECT subcommand of DUMPSCAN

046E

A MAXIMUM OF SIXTEEN USERIDS MAY BE SPECIFIED

Explanation: You entered more than 16 USERID values under the SELECT subcommand of DUMPSCAN. Although you may specify selection criteria on many different uses of the SELECT subcommand, the total number of USERID values in all your requests must not exceed 16 for one invocation of the TRACE subcommand of DUMPSCAN.

System Action: All of the selection criteria you entered in response to previous prompts remain in effect. The selection criteria you specified in the most recent response are ignored.

User Response: You can enter other selection criteria or another subcommand, or you can quit DUMPSCAN processing.

Commands That Detect Condition:

```
SELECT subcommand of DUMPSCAN
```

049E

LEADING ZEROS REQUIRED FOR CODE SELECTION VALUE 'value'

Explanation:

1. You entered a CODE value without leading zeros in response to the prompt for trace selection criteria on the TRACERED command. A 4-digit trace selection code with leading zeros is expected as a CODE value when you are prompted for selection criteria.
2. You entered a CODE value without leading zeros as an operand on the SELECT subcommand of DUMPSCAN. A 4-digit trace selection code with leading zeros is expected as a CODE value on the SELECT subcommand of DUMPSCAN.

System Action:

1. All of the selection criteria you entered in response to previous prompts, remain in effect. The selection criteria you specified in the most recent response is ignored. You will be reprompted for additional trace selection criteria.
2. All of the selection criteria you entered previously, remain in effect. The selection criteria you specified in this request is ignored.

User Response:

1. When reprompted for additional trace selection criteria, reenter the trace CODE value with leading zeros or, if you wish to end the selection process, enter a null line.

   See message 010A for an explanation of trace CODE selection values that are valid.

2. Reenter the SELECT subcommand of DUMPSCAN specifying the trace CODE value with leading zeros. See the SELECT subcommand section of this manual for an explanation of the trace CODE selection values that are valid.

Commands That Detect Condition:

```
1. TRACERED
2. SELECT subcommand of DUMPSCAN
```

050E **'value' IS NOT A VALID SELECTION VALUE**

Explanation:

1. You entered an invalid value on the prompt for trace selection criteria and the invalid value is specified in the message.

2. You entered an invalid value on the SELECT subcommand of DUMPSCAN. That value is specified in the message.

System Action:

1. All of the selection criteria you entered in response to previous prompts remain in effect. The selection criteria you specified in the most recent response are ignored. You will be reprompted for additional trace selection criteria on the TRACERED command.

2. All of the selection criteria you entered previously remain in effect. The selection criteria you specified in the most recent request is ignored.

User Response:

1. When reprompted for additional trace selection criteria on the TRACERED command, enter the additional values. Or, if you want to end the selection process, enter a null line.

   See message 010A (page 161) or the TRACERED command section (page 48) of this manual for an explanation of the trace selection values that are valid.

2. Reenter the SELECT subcommand of DUMPSCAN with new criteria. See the SELECT subcommand section of this manual for an explanation of trace selection values that are valid.

Commands That Detect Condition:

```
1. TRACERED
2. SELECT subcommand of DUMPSCAN
```

051E

THE keyword OPERAND 'value' WAS NOT PREVIOUSLY SELECTED

Explanation: The value specified in the message is an invalid RESET value. You did not previously select that value and, therefore, you cannot RESET it.

System Action: All of the selection criteria you entered previously remain in effect. The SELECT RESET criteria you specified in the most recent request are ignored.

User Response: Reenter the SELECT subcommand of DUMPSCAN with the QUERY keyword. Verify the selection criteria you want to reset to a new value. Enter the SELECT RESET request with the correct data or enter another DUMPSCAN subcommand or quit.

Commands That Detect Condition:

```
SELECT subcommand of DUMPSCAN
```

052E

CP SELECTION KEYWORD(S) SPECIFIED FOR GUEST TRACE TABLES - KEYWORD(S) IGNORED

Explanation: The selection keyword you specified is not valid for guest trace tables.

System Action: Any CP selection keywords are ignored. You will be reprompted for a valid keyword.

User Response: Enter a valid selection keyword for guest trace table entries. TIME is the only valid selection. Or enter a null line to end the prompt.

Commands That Detect Condition:

```
TRACERED
```

060E

TAPE WAS NOT CREATED BY THE TRSAVE COMMAND

Explanation: The input tape does not contain a valid volume label created by the TRSAVE command.

System Action: The input tape is rewound. The system then allows you to mount another tape.

User Response: If you want additional trace data to be processed, make sure that the input tape you mounted was created by the TRSAVE command.

Commands That Detect Condition:

```
TRACERED
```

061E

TAPE vdev IS NOT ATTACHED

Explanation: The virtual tape device specified by the variable "vdev" in this message is not attached to your virtual machine.

System Action: The TRACERED command terminates.

User Response: Attach a real tape device to your virtual machine as indicated by the virtual device address variable. Then make sure the desired TRSAVE input tape is mounted. Reenter the TRACERED command.

Commands That Detect Condition:

```
TRACERED
```

062E

PERMANENT TAPE I/O ERROR

Explanation: A permanent I/O error occurred while the TRSAVE input tape was being read.

System Action: The TRACERED command terminates.

User Response: Ask the operator to take corrective action to avoid the I/O error. Reenter the TRACERED command. If the problem persists, contact IBM service personnel.

Commands That Detect Condition:

```
TRACERED
```

070I

NO TRACE ENTRIES FOUND MEETING THE REQUESTED SELECTION CRITERIA

Explanation: Although the TRACERED command or the TRACE subcommand of DUMPSCAN successfully processed the current TRSAVE input tape or trace table(s) in the dump file, no trace entries met the selection criteria you specified.

System Action: For TRACERED, the system allows you to have another TRSAVE input tape mounted or to end TRACERED command processing. Subcommand processing continues for the TRACE subcommand of DUMPSCAN. You can enter another subcommand or quit DUMPSCAN processing.

User Response: If you are using the TRACERED command, check the selection criteria you specified in response to the 010A prompt. For both TRACERED or the TRACE subcommand of DUMPSCAN, if you expected trace entries meeting the selection criteria to be found, make sure you correctly typed in all the data reduction values.

If you did correctly type in the values for TRACERED, reenter that command. Specify a different set of selection criteria when you respond to the 010A prompt on TRACERED. Or, mount a tape created in another time interval and issue the TRACERED command with the same selection criteria.

If you are using the TRACE subcommand of DUMPSCAN, check the selection criteria you specified with the SELECT subcommand. If you expected trace entries that meet the selection criteria you find, be sure you correctly typed in all the data reduction values. If you correctly entered those values, reenter the SELECT subcommand and specify a different set of selection criteria. Then enter the TRACE subcommand again.

Commands That Detect Condition:

```
TRACERED
TRACE subcommand of DUMPSCAN
```

071I

ONE OR MORE TRACE ENTRIES MISSING FROM TRSAVE TAPE(S)--VOL SEQ # nn

Explanation: When one or more TRSAVE input tapes were processed, certain records were encountered that indicated trace entries are missing. Missing trace entries can result when the TRSAVE input tapes are being created. The trace table wraps faster than CP can write the entries to the output device. This situation is recorded in a formatted form in the TRACERED print or CMS output files. The CPU address from which trace data was lost is recorded. Normal processing continues with the next available trace entry from that processor.

System Action: The TRACERED command has successfully completed and the system waits for a new command.

User Response: Examine the TRACERED output by checking the TRACERED print or CMS output files. All records identifying lost data are preceded and followed by a blank line.

Commands That Detect Condition:

```
TRACERED
```

074E **THE SYSTEM TRACE FILE NAME 'filename' WAS NOT FOUND**

Explanation: You specified a system trace filename as input to the TRACERED command and that file was not found.

System Action: Command processing is terminated.

User Response: You can verify the name of the referenced system trace file by using the QUERY TRFILES command.

Commands That Detect Condition:

```
TRACERED
```

076E **THE OPERAND GTRACE IS NOT VALID FOR CP TRACE DATA**

Explanation: You requested that TRACERED produce an output file in OS QSAM format containing GTRACE records, but you specified an input file containing CP trace table data rather than guest trace data recorded by GCS.

System Action: TRACERED command processing is rejected.

User Response: Use the TRACERED command again specifying the correct input file. Use the QUERY TRFILES command to determine the spoolid of the system trace file containing the GCS GTRACE records you want to process.

Commands That Detect Condition:

```
TRACERED
```

080E **DUPLICATE INPUT SYSTEM TRACE FILES WERE SPECIFIED**

Explanation: You entered two identical filenames or spoolids, or you entered a spoolid and then a filename, but the spoolid was within a set designated by the filename.

System Action: Command processing is terminated.

User Response: Reenter the TRACERED command with correct input.

Commands That Detect Condition:

```
TRACERED
```

081E

SYSTEM TRACE FILE spoolid IS INVALID

Explanation: You entered an invalid system trace file spoolid on the TRACERED command line. The parameter must be a decimal number from 1 to 9999 for the spoolid.

System Action: Command processing is terminated.

User Response: Use the QUERY TRFILES command to find the associated filename and then you can ascertain valid spoolids.

Commands That Detect Condition:

```
TRACERED
```

082E

ERROR { OPENING|READING|CLOSING} SYSTEM TRACE FILE spoolid, CODE nn

Explanation: TRACERED received an error during an open, read, or close of a system trace file.

System Action: Command processing is terminated.

User Response: See VM/XA SP CP Command Reference for an explanation of the code for DIAGNOSE X'E0'.

Commands That Detect Condition:

```
TRACERED
```

083I

PROCESSING SYSTEM TRACE FILES spoolid CREATED mm/dd hh:mm:ss

Explanation: The file designated by spoolid is being processed.

System Action: TRACERED will process trace entries from the specified system file.

User Response: None

Commands That Detect Condition:

```
TRACERED
```

084I

PROCESSING COMPLETE - nnnnnnnn TRACE ENTRIES FORMATTED

Explanation: The TRACERED command has finished processing. The total number of trace entries processed by the TRACERED command is specified by "nnnnnnnn".

System Action: Command processing has completed normally.

User Response: None

Commands That Detect Condition:

```
TRACERED
```

085E

SYSTEM TRACE FILENAME 'filename' HAS FILES FROM MORE THAN ONE ORIGINATOR. RE-ENTER COMMAND AND SPECIFY FILENAME WITH SPOOLID

Explanation: The filename entered was created by two different originators.

System Action: Command processing is terminated.

User Response: Issue the QUERY TRFILES command to find the starting spoolid for the filename belonging to the originator you want. Reenter the TRACERED command in a form with the originator's name, the filename, and id where id is the beginning spoolid for the originator's filename.

Commands That Detect Condition:

```
TRACERED
```

086E

SYSTEM TRACE FILE spoolid IS CURRENTLY IN USE

Explanation: TRACERED attempted a DIAGNOSE X'E0' OPEN and failed on the specific system trace file because it was in use.

System Action: Command processing is terminated.

User Response: Enter QUERY TRSAVE to determine if tracing is still active and if the specific system trace file you requested is still in use. If it is and you want to process immediately, you must turn TRSAVE OFF for the user. After processing, you can turn it on for the user with a different filename, then reenter the TRACERED command for that user.

Commands That Detect Condition:

```
TRACERED
```

088E

THE SYSTEM TRACE FILE SPOOLID 'spoolid' WAS NOT FOUND

Explanation: You issued TRACERED with a specific system trace file spoolid as input and that file was not found.

System Action: Command processing is terminated.

User Response: Use the QUERY TRFILES command to verify the spoolid of the referenced system trace file.

Commands That Detect Condition:

```
TRACERED
```

089E

ONLY ONE CP TAPE OR CP SYSTEM TRACE FILE IS ALLOWED

Explanation: A total of five system trace files can be merged. However, only one CP system trace file or tape can be included. You need to specify one of the following:

- One CP system trace file with up to four TRSOURCE system trace files
- One CP tape with up to four TRSOURCE system trace files
- Up to five TRSOURCE system trace files.

System Action: Command processing is terminated.

User Response: Reenter the TRACERED command and specify only one CP tape or file.

Commands That Detect Condition:

TRACERED

090E **DUPLICATE KEYWORDS WERE SPECIFIED**

Explanation: You entered a duplicate keyword in the TRACERED command, or the TIME selection keyword was entered twice.

System Action: Processing ends if a duplicate keyword was specified on the TRACERED command line, but processing continues for selection criteria. Current time parameters are ignored.

User Response: Reenter the TRACERED command and omit the duplicate keyword, or continue entering selection criteria.

Commands That Detect Condition:

TRACERED

091I **ONE OR MORE TRACE ENTRIES MISSING FROM SYSTEM TRACE FILE(S) FOR spoolid**

Explanation: During the processing of system trace files, records were found to indicate missing trace entries. Missing entries result if trace table entries wrap faster than CP can write the entries to an output device or file. The wrapping situation is recorded in TRACERED print or CMS output files with the CPU address from which the data was lost. Normal processing continues with the next available trace entry from that processor.

System Action: Command processing continues.

User Response: Examine the TRACERED output by checking the TRACERED print or CMS output files. All records identifying lost data will be preceded and followed by a blank line.

Commands That Detect Condition:

TRACERED

092I **TIME ZONES ARE DIFFERENT FOR SPOOLID spoolid. FILE IS NOT PROCESSED**

Explanation: The files you want to merge were created in different time zones.

System Action: Command processing is terminated.

User Response: Reenter the TRACERED command with files from the same time zone.

Commands That Detect Condition:

TRACERED

093I **TIME ZONES ARE DIFFERENT FOR TAPE - VOL SEQ# nn. FILE IS NOT PROCESSED.**

Explanation: You cannot merge a tape and file that were created in different time zones.

System Action: Command processing is terminated.

User Response: Reenter the TRACERED command with files from the same time zone.

Commands That Detect Condition:

```
TRACERED
```

094E **THE STARTING DATE AND TIME MUST BE LESS THAN THE ENDING DATE AND TIME**

Explanation: You specified a starting date or time that is greater than the ending date or time as a selection criteria.

System Action: TRACERED will prompt you for new selection criteria.

User Response: Reenter the TIME keyword and a starting date that is earlier than the ending date, or a null line to end the prompt.

Commands That Detect Condition:

```
TRACERED
```

095E **THE SYSTEM TRACE FILE SPOOLID 'spoolid' WAS NOT FOUND WITHIN FILENAME 'filename'**

Explanation: You issued the TRACERED command in the form "NAME fn, id". The *id* which represents a spoolid is not associated with the filename.

System Action: Command processing is terminated.

User Response: Use the QUERY TRFILES command to verify the spoolid of the referenced system trace file.

Commands That Detect Condition:

```
TRACERED
```

096E **RE-IPL CMS. TRACERED HAD A SEVERE ERROR**

Explanation: There was an I/O error writing to a file.

System Action: Command processing is terminated.

User Response: Re-IPL CMS and reenter the TRACERED command.

Commands That Detect Condition:

```
TRACERED
```

110E

THE FILE NAME 'filename' IS NOT VALID

Explanation: If in ADDMAP, DUMPSCAN, or PRTDUMP mode, the dump file name entered is not valid. If in MAP mode, the file name entered is not valid. Acceptable file names are the file names for a CMS file: a 1- to 8-character string with the characters 0-9, A-Z, #, @, +, -, _, :, and $.

System Action: If in ADDMAP, DUMPSCAN, or PRTDUMP mode, the message 120A will be issued to prompt you for the correct name of the CMS file containing the dump. If in MAP mode, either message 121A or 122A will be issued to prompt you for the correct name of the CMS file containing either the load map or the module map.

User Response: If in ADDMAP, DUMPSCAN, or PRTDUMP mode, enter the correct dump file name. If in MAP mode, enter either the correct load map or module map name.

Commands That Detect Condition:

```
ADDMAP
DUMPSCAN
PRTDUMP
MAP
```

120A

ENTER DUMP FILE NAME IN THE FORM XXXXXXXX, OR IN THE FORM XXXXXXXX FILEMODE

Explanation: A command was issued to process a specific dump file, and you are being asked to specify which file you wish to process.

System Action: No further processing occurs until the user responds.

User Response: You may specify a valid dump fileid to continue command processing, or enter HX to terminate the command.

Commands That Detect Condition:

```
ADDMAP
DUMPSCAN
PRTDUMP
```

121A

ENTER THE FILENAME FILETYPE FILEMODE OF THE INPUT type LOAD MAP, A NULL LINE, SUBSET OR HX

Explanation: You are being prompted for the filename, filetype, and filemode of a load map to be processed. If the file cannot be found, this prompt will be repeated.

Where "type" is:

CP NUCLEUS
CMS NUCLEUS
GCS NUCLEUS
PVM (Primary CMS/PVM load map)
RSCSNET
RSCSV2

System Action: The dump viewing facility waits for a response.

User Response: Enter one of the following:

1. "filename filetype filemode"

If the filetype and filemode are not included, then MAP and * will be employed, respectively.

2. A null line

   Requests use of the predefined filename:

Type	*Input Map Names*
CP	CPLOAD MAP *
CMS	CMSNUC MAP *
GCS	GCSNUC MAP *
PVM	CMSNUC MAP * (primary) PVM MAP * (secondary)
RSCSNET	RSCSNET MAP *
RSCSV2	RSCSV2 MAP *

3. SUBSET

   Requests CMS subset mode. You should verify the name of the file you wish to process. You should also verify that you are currently accessing the CMS disk on which the file resides.

4. HX

   Terminates the MAP command.

Commands That Detect Condition:

```
MAP
```

122A **ENTER THE FILENAME FILETYPE FILEMODE OF THE OUTPUT type MODULE MAP, A NULL LINE, SUBSET, OR HX**

Explanation: You are being prompted for the filename, filetype, and filemode that will contain the compressed input load maps.

Where type is:

CP
CMS
GCS
PVM
RSCSNET
RSCSV2

System Action: The dump viewing facility waits for a response.

User Response: Enter one of the following:

1. "filename filetype filemode"

   If the filetype and filemode are not included, then MAP and A will be employed, respectively.

2. A null line

   Requests use of the predefined filename:

Type	*Output Map Names*
CP	HCPXADVF MAP A
CMS	CMSDVF MAP A
GCS	GCSDVF MAP A
PVM	PVMDVF MAP A

RSCSNET	RSCSDVF MAP A
RSCSV2	RSV2DVF MAP A

3. SUBSET

   Requests CMS subset mode. You should verify the name of the file you wish to create. You should also verify that you are currently accessing the CMS disk on which the file will reside.

4. HX

   Terminates the MAP command.

Commands That Detect Condition:

```
MAP
```

123A **ENTER THE MAP TYPE, A NULL LINE, SUBSET, OR HEX**

Explanation: You entered the MAP PROmpt or MAP command. The dump viewing facility is requesting information on the type of map you want to process.

System Action: The dump viewing facility waits for a response.

User Response: Enter one of the following:

1. CP

   Specifies that a CP load map is to be processed.

2. CMS

   Specifies that a CMS load map is to be processed.

3. PVM

   Specifies that VM/Pass-Through load maps are to be processed.

4. RSCSnet

   Specifies that an RSCSNET load map is to be processed.

5. GSC

   Specifies that a GCS load map is to be processed.

6. RSCSV2

   Specifies that an RSCSV2 load map is to be processed.

7. Null line

   Specifies that a CP load map is to be processed.

8. SUBSET

   Requests CMS subset mode.

9. HX

   Terminates the MAP command.

Commands That Detect Condition:

```
MAP
```

130E

THE type FILE filename filetype filemode WAS NOT FOUND

Explanation: A command was issued to process a specific CMS file, and that file was not found.

System Action: Command processing is terminated, or you are prompted for the correct file name.

User Response: You should verify the name of the file you wish to process. You should also verify that you are currently accessing the CMS disk on which the file resides. If prompted, enter the correct file name, or else reissue the command with the correct file name.

Commands That Detect Condition:

Command	*Type*
ADDMAP	MODULE MAP
DUMPSCAN	DUMP FILE
MAP	type maptype LOAD MAP
PRTDUMP	DUMP FILE

Where:

type	**maptype**	
CP	NUCLEUS	
CMS	NUCLEUS	
GCS	NUCLEUS	
PVM	-	(Secondary PVM load map)
RSCSNET	-	
RSCSV2	-	

135E

THE FILE filename filetype filemode ALREADY EXISTS

Explanation: The output file specified on the command exists already.

System Action: If you included the specified file identifier in the TRACERED or MAP command, the command terminates. If you entered the specified fileid as a response to a MAP prompt, you will be reprompted for another fileid.

User Response: Check to see that you typed the fileid correctly. If you did not enter the fileid correctly, do so on either the TRACERED or MAP command line or in response to the MAP prompt. If you did enter the fileid correctly, try renaming the file and reentering the command or response.

Commands That Detect Condition:

```
MAP
TRACERED
```

140E

THE FILE filename filetype filemode IS NOT A VALID type FILE

Explanation: The specified file does not contain valid data for the specified type of file. For a load map, the file containing the map was not in the expected format.

System Action: Command processing is terminated.

User Response: Check to see if the fileid was typed incorrectly. Correct the input and reenter or enter a new command.

Commands That Detect Condition:

Command	*Type*
MAP	type

Where:

The type is one of the following:

CP NUCLEUS LOAD MAP
CMS NUCLEUS LOAD MAP
GCS NUCLEUS LOAD MAP
PVM LOAD MAP
RSCSNET LOAD MAP
RSCSV2 LOAD MAP

145E THE DUMP FILE REQUESTED IS NOT A VALID DUMP

Explanation: The specified dump file was not a valid CP or virtual machine dump.

System Action: Command processing is terminated.

User Response: Reenter the command with the correct file name.

Commands That Detect Condition:

DUMPSCAN

150I THE type FILE filename filetype filemode IS INCOMPLETE

Explanation: The requested file was incomplete.

System Action: For MAP and ADDMAP, processing is terminated. For DUMPSCAN and PRTDUMP, processing continues.

User Response: Generate a new load map.

Commands That Detect Condition:

Command	*Type*
ADDMAP	DUMP
DUMPSCAN	DUMP
MAP	CP NUCLEUS LOAD MAP
	CMS NUCLEUS LOAD MAP
	GCS NUCLEUS LOAD MAP
	PVM LOAD MAP (PVM secondary load map)
	RSCSNET LOAD MAP
	RSCSV2 LOAD MAP
PRTDUMP	DUMP

200I PROCESSING FILE filename DUMP filemode

Explanation: The command has begun processing the CMS filename DUMP.

System Action: Command processing continues.

User Response: None.

Commands That Detect Condition:

DUMPSCAN
PRTDUMP

210I **THE SUMMARY REPORTS HAVE BEEN PRINTED FOR FILE filename DUMP filemode**

Explanation: The PRTDUMP command has finished processing the specified dump file.

User Response: None.

Commands That Detect Condition:

PRTDUMP

220I **THE type SUMMARY REPORT IS NOT COMPLETE; SEE REPORT FOR DETAILS**

Explanation: An error was encountered in the PRTDUMP processing of the specified summary report, and it was not possible to print the entire summary report.

System Action: The printing of other dump data will continue.

User Response: Refer to the printed summary report for the detailed reasons why the summary report was not complete.

Commands That Detect Condition:

Command	*Type*
PRTDUMP	FRMTB
	FRAMETBL
	MAP
	PROCESSOR
	RIOBLOK
	SYMPTOM
	TRACE
	USER
	VMDBK
	DUMPID

230A **DO YOU WANT PRINTING TO CONTINUE? ENTER YES OR NO**

Explanation: A condition has been detected that might cause you not to want the dump summary reports printed. This message is preceded by another message, which explains the condition. You have the choice of whether or not to continue printing the dump summary reports.

System Action: If you enter an invalid response, the prompt is reissued. If you respond YES, command processing continues. If you respond NO, command processing is terminated.

User Response: You must say whether the printing of the dump summary reports should continue. Valid responses are YES or NO.

Commands That Detect Condition:

PRTDUMP

260I **THE FORMATTING ROUTINE 'routine name' FOR 'type' COULD NOT BE FOUND**

Explanation: A specific formatting routine was not found on any disk accessed by the user.

System Action: For DUMPSCAN and PRTDUMP, no user exit is taken and program product subcommands will not work.

User Response: You must verify the name of the file you want to use and that it is located on a disk you have accessed.

Commands That Detect Condition:

Command	*Routine*	*Type*
DUMPSCAN	DMMDCM	CMS
	CSIIDS	GCS
	DVMZDS	PVM
	DMTZDS	RSCSNET
	CSIIDS	RSCSV2
PRTDUMP	CSIIPR	GCS
	CSIIPR	RSCSV2

263E **SVC 199 IS CURRENTLY IN USE**

Explanation: The HNDSVC macro was issued for SVC 199 and that SVC number was already in use.

System Action: DUMPSCAN or PRTDUMP processing continues, but program product subcommands will not be recognized.

User Response: Determine which program is using SVC 199. That program cannot run concurrently with the dump viewing facility.

Commands That Detect Condition:

DUMPSCAN
PRTDUMP

265I **PRINTING FORMATTED STORAGE IS NOT SUPPORTED BY THE DUMP VIEWING FACILITY**

Explanation: You have tried to print a formatted dump from 0 to the end with a program product formatting routine. This is not supported by the dump viewing facility.

System Action: The load maps, registers, and PSW will be printed.

User Response: Use the DUMPLOAD command to print the formatted dump from 0 to the end.

Commands That Detect Condition:

PRTDUMP

266I THE ROUTINE routine name FOR spoolid CANNOT BE FOUND

Explanation: The user exit necessary for a formatting routine was not found.

System Action: Processing continues. The trace entry record will be processed in hex mode.

User Response: Run the TRACERED command again and be sure you are linked to the disks that contain the user exits.

Commands That Detect Condition:

```
TRACERED
```

300I THE DUMP HAS NO MODULE MAP

Explanation: A module map was not found at the end of the dump file currently being processed.

System Action: If the FINDMOD subcommand was issued, subcommand processing is terminated and the dump viewing facility waits for a new subcommand to be entered.

If the PRTDUMP command was issued, printing of other dump data will continue. A note will be printed in the output indicating that the module map was not available.

User Response: The ADDMAP command may be used to append a module map to the dump file.

Commands That Detect Condition:

```
FINDMOD subcommand of DUMPSCAN
PRTDUMP
```

301I DUMP filename filetype filemode ALREADY HAS A MODULE MAP

Explanation: The specified dump file already has an appended module map.

System Action: Message 302A is issued.

User Response: None.

Commands That Detect Condition:

```
ADDMAP
```

302A DO YOU WANT TO REPLACE THE PREVIOUS MAP? ENTER YES OR NO

Explanation: You are attempting to add a map to a dump that already has a map appended. You have the option of replacing the pointer to the old map with a pointer to the new map.

System Action: The dump viewing facility waits for a response.

User Response: Enter one of the following:

YES	The new map will be appended to the dump and the pointer updated.
NO	Terminate the ADDMAP command.
Any other response	Reprompt.

Commands That Detect Condition:

```
ADDMAP
```

304E

THE MODULE MAP CANNOT BE APPENDED TO A type DUMP

Explanation: You tried to append a module map to a dump of the type specified. The type is a soft abend.

System Action: Command processing continues.

User Response: None.

Commands That Detect Condition:

```
ADDMAP
```

310E

THE FILES filename filetype filemode AND filename filetype filemode ARE INCOMPATIBLE

Explanation: The dump file and the module map file represent two different levels of the system.

System Action: Command processing is terminated.

User Response: Verify that the correct fileids were specified.

Commands That Detect Condition:

```
ADDMAP
```

311E

THE type1 MODULE MAP FILE filename filetype filemode DOES NOT MATCH THE type2 DUMP filename filetype filemode

Explanation: The type of the module map to be appended does not match the type of the dump. *type1* is a CP, CMS, PVM, or RSCS module map and *type2* is the dump file type.

System Action: Command processing is terminated and the module map is not appended to the dump.

User Response: Verify that the correct fileids were specified on the command. Reenter the command with the correct files specified or enter another command.

Commands That Detect Condition:

```
ADDMAP
```

315E

THE PVM LOAD MAP filename filetype filemode OVERLAPS A PREVIOUS MAP

Explanation: The address range of the PVM input load map overlaps either partially or completely the address range of the previously included CMS nucleus load map in the output dump viewing facility module map.

System Action: Command processing is terminated and the module map is not created.

User Response: Reenter the command with the proper input load maps.

Commands That Detect Condition:

```
MAP
```

320I **THE MODULE MAP filename filetype filemode HAS BEEN CREATED FROM LOAD MAP filename filetype filemode**

Explanation: The module map has been successfully created from the input load map.

System Action: Command processing is finished.

User Response: None.

Commands That Detect Condition:

```
MAP
```

321I **THE MAP filename filetype filemode HAS BEEN APPENDED TO DUMP filename**

Explanation: The module map has been successfully appended to the dump indicated.

System Action: Command processing is finished.

User Response: None.

Commands That Detect Condition:

```
ADDMAP
```

330I **THE INPUT type LOAD MAP NAME IS filename filetype filemode**

Explanation: You entered a null line in response to a prompt for a load map name. Consequently, the MAP command substituted a predefined filename, filetype, and filemode.

Where the type is:

CP NUCLEUS
CMS NUCLEUS
PVM (secondary PVM load map)
RSCSNET
GCS NUCLEUS
RSCSV2.

System Action: Command processing continues.

User Response: Continue or terminate command processing when possible.

Commands That Detect Condition:

```
MAP
```

331I

MAP TYPE IS CP

Explanation: You entered a null line in response to prompting message 123A or the map type was not specified on the MAP command line. Consequently, CP map type processing will be used.

System Action: Command processing continues.

User Response: Continue or terminate command processing when possible.

Commands That Detect Condition:

```
MAP
```

332I

THE LOAD MAP filename filetype filemode HAS BEEN APPENDED TO MODULE MAP filename filetype filemode

Explanation: The load map has been successfully appended to the module map.

System Action: Command processing continues.

User Response: None.

Commands That Detect Condition:

```
MAP
```

340I

DYNAMICALLY CREATING MODULE MAP

Explanation: You entered the DUMPSCAN or PRTDUMP command with a CP dump on a writable disk that has no appended module map. A map is being built and appended to the dump file specified in the command.

System Action: Command processing continues.

User Response: None.

Commands That Detect Condition:

```
DUMPSCAN
PRTDUMP
```

341I

DISK nn IS READ ONLY, DYNAMIC MAP NOT CREATED

Explanation: The disk containing the dump file you requested is a read-only disk. The module map cannot be appended to the dump.

System Action: Command processing continues.

User Response: If you do not need the functions of either the FINDMOD subcommand of the DUMPSCAN command or the module map summary of the PRTDUMP command, no response is necessary. If you do need these functions, copy the dump file to a CMS disk where you have write access. Then reissue the DUMPSCAN or PRTDUMP command specifying the dump file on the CMS disk to which you have write access.

Commands That Detect Condition:

```
DUMPSCAN
PRTDUMP
```

342I

INSUFFICIENT DASD SPACE, DYNAMIC MAP NOT CREATED

Explanation: While the dump viewing facility attempted to append the dynamically created module map to the requested dump file, the disk containing the dump became full. The load map is not appended.

System Action: Command processing continues.

User Response: If you do not need the functions of either the FINDMOD subcommand of the DUMPSCAN command or the module map summary of the PRTDUMP command, no user response is necessary. If you do need these functions, delete any unwanted files from the disk containing the dump file to secure space for the appended module map. Then reissue the DUMPSCAN or PRTDUMP command.

Commands That Detect Condition:

```
DUMPSCAN
PRTDUMP
```

350I

THE MODULE OR ENTRY POINT name WAS NOT IN STORAGE AT THE TIME OF THE DUMP

Explanation: The specified module or entry point was not in real storage at the time of the dump.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
FINDMOD subcommand of DUMPSCAN
```

351E

THE ADDRESS address IS NOT IN A MODULE

Explanation: The address specified on the FINDMOD subcommand is not contained in any module in the dump.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
FINDMOD subcommand of DUMPSCAN
```

352I

THE MODULE OR ENTRY POINT name WAS NOT FOUND IN THE MODULE MAP

Explanation: The specified module or entry point name was not found in the module map appended to the dump.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Check if the correct module or entry point name was entered. Correct the name and reenter the subcommand, or enter a new subcommand.

Commands That Detect Condition:

```
FINDMOD subcommand of DUMPSCAN
```

353I **THE MODULE OR ENTRY POINT name IS LOCATED AT ADDRESS address**

Explanation: The specified module or entry point name was found to be in storage at the time of the dump, but the page was not included in the dump.

System Action: This message will be followed by a message indicating why the page was not found.

Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
FINDMOD subcommand of DUMPSCAN
```

400E **subcommand IS NOT A RECOGNIZED SUBCOMMAND**

Explanation: The indicated subcommand you issued is not a known DUMPSCAN subcommand.

System Action: The dump viewing facility waits for you to enter another DUMPSCAN subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
DUMPSCAN
```

401I **READY, DUMP TYPE IS type**

Explanation: This message is issued whenever DUMPSCAN is waiting for a subcommand.

System Action: The dump viewing facility waits for the next subcommand.

User Response: Enter a DUMPSCAN subcommand.

Commands That Detect Condition:

```
DUMPSCAN
```

Note: The “type” value in the dump file will be used.

402E

THE subcommand SUBCOMMAND IS NOT VALID FOR type DUMPS

Explanation: The DUMPSCAN subcommand entered is not valid for the specified type of dumps. *type* is the dump type.

System Action: The subcommand is not processed. After issuing the message, the dump viewing facility waits for you to enter another subcommand.

User Response: You may continue analyzing the dump by issuing a new subcommand.

Commands That Detect Condition:

```
DUMPSCAN
```

403E

THE option OPTION IS NOT VALID FOR type DUMPS

Explanation: The PRTDUMP option entered is not supported for the specified type of dump.

System Action: Command processing is terminated.

User Response: Correct and reenter the request, or enter a new request.

Commands That Detect Condition:

```
PRTDUMP
```

404E

THE 'operand' OPERAND IS NOT VALID FOR type DUMPS

Explanation: When you issued a subcommand of DUMPSCAN, you entered an operand that was not valid for that type of dump.

System Action: The subcommand is terminated. DUMPSCAN waits for you to enter a new subcommand.

User Response: Reissue the subcommand without using the specified operand.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

405E

THE IPCS SUBCOMMAND 'subcommand' IS NOT SUPPORTED FOR type DUMPS

Explanation: When you issued the IPCS subcommand, you entered a subcommand that is not supported for the specified type of dump.

System Action: The subcommand is not processed. After issuing the message, the dump viewing facility waits for you to enter another subcommand.

User Response: Issue a new subcommand to continue analyzing your dump.

Commands That Detect Condition:

```
DUMPSCAN
```

446I **BLOCK INITIALIZATION HAS STARTED, PLEASE BE PATIENT**

Explanation: The BLOCK subcommand is building the control block look-up table.

System Action: BLOCK will process the control block definitions specified in the control block.

User Response: None.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

447I **BLOCK INITIALIZATION HAS COMPLETED SUCCESSFULLY**

Explanation: The BLOCK subcommand has been successful in building the look-up table.

System Action: BLOCK will process the user request to display a control block.

User Response: None.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

448I **BLOCK INITALIZATION HAS FAILED**

Explanation: The BLOCK subcommand has been unsuccessful in the building of the look-up table.

System Action: The subcommand is terminated. You will receive an additional message or messages indicating the cause of the error.

User Response: Correct the error, then reenter the BLOCK subcommand.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

449I **UNABLE TO PROCESS THE BLOCK SUBCOMMAND**

Explanation: The BLOCK subcommand cannot be executed because of an error condition.

System Action: The subcommand is terminated. You will receive a second message to further explain the error.

User Response: Correct the error, then reenter the BLOCK subcommand.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

450I **THE BLOCK TABLE 'tablename' WAS NOT FOUND**

Explanation: The indicated block table name was not found on any disk you have accessed.

System Action: Command processing continues, but the BLOCK subcommand cannot format control blocks contained in the specified table.

User Response: You can terminate the DUMPSCAN command, then access the disk containing the table. Reenter the DUMPSCAN command and then the BLOCK subcommand.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

451I **THE 'cbname' CONTROL BLOCK CANNOT BE FOUND IN ANY BLOCK TABLE**

Explanation: The control block name you specified was not found in any of the BLOCK table files.

System Action: Subcommand processing is terminated.

User Response: Reenter the BLOCK subcommand with a correct control block name.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

452I **THE FIELD 'field name' WAS NOT FOUND**

Explanation: The field name you specified was not found in the block table entry for that control block.

System Action: You will receive message 455A which reprompts you for the correct field name. Any field names already entered that were correct will be saved and used by the BLOCK subcommand.

User Response: Reenter the field name correctly or end the prompt.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

454I **UNKNOWN CONTROL BLOCK ID, RE-ENTER BLOCK SUBCOMMAND USING THE CONTROL BLOCK NAME**

Explanation: You entered an asterisk (*) for a control block name. The dump viewing facility could not find the actual control block name in its internal table.

System Action: The subcommand is terminated and DUMPSCAN waits for you to enter a new subcommand.

User Response: Reenter the BLOCK subcommand using the control block name instead of an asterisk.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

455A

ENTER FIELD NAME(S) TO BE FORMATTED, QUIT TO END SUBCOMMAND OR NULL LINE TO END PROMPT

Explanation: This is a prompt for you to enter the names of specific fields for formatting.

System Action: The system waits for your response.

User Response: You should enter either the field names, separated by blanks, or a null line to indicate that there are no more field names requested. If you enter QUIT, the BLOCK subcommand processing ends. If you enter a null line in response to the first prompt, only entries flagged as default fields are formatted.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

456I

TOO MANY FIELD NAMES SPECIFIED. THE EXTRA ARE IGNORED

Explanation: You have requested more than 512 field names for formatting.

System Action: The first 512 field names will be used. The extra ones are ignored. You will not be prompted for more control block names and processing continues.

User Response: None. If you want to see more than 512 fields, reenter the BLOCK subcommand and specify ALL instead of PROMPT.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

457E

BLOCK SUBCOMMAND NOT SUPPORTED FOR type DUMP

Explanation: The BLOCK subcommand is not supported for the specified type of dump.

System Action: The subcommand is terminated and DUMPSCAN waits for a new subcommand to be entered.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

458E

THE CONTROL FILE filename filetype WAS NOT FOUND

Explanation: The BLOCK subcommand is not supported because a control file was not found for the specified type of dump you requested.

User Response: You should access the disk containing the necessary control file and reenter the DUMPSCAN command, or you should issue another subcommand.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

459E

CANNOT LOCATE ANY OF THE BLOCK TABLE FILES

Explanation: The dump viewing facility cannot locate any of the BLOCK table files used to format control blocks.

System Action: The subcommand is terminated and DUMPSCAN waits for a new subcommand to be entered.

User Response: Access the disk containing the BLOCK tables and reenter the BLOCK subcommand, or issue another subcommand.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

510I

THE &NAME TABLE IS FULL

Explanation: The space available in the &name table is not sufficient for the entry being added.

System Action: The dump viewing facility waits for the next subcommand.

User Response: You may redefine previous entries with shorter entries to make space available in the &name table, or you may enter a new subcommand.

Commands That Detect Condition:

```
&name subcommand of DUMPSCAN
```

511E

&name IS NOT IN THE &NAME TABLE

Explanation: You invoked "&name" and it was not in the table.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter the & subcommand to list all entries in the table and then reenter a valid &name or any other DUMPSCAN subcommand.

Commands That Detect Condition:

```
DUMPSCAN
```

512I

THE &NAME TABLE IS EMPTY

Explanation: There are no entries in the &name table.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Use the &name subcommand to add entries to the &name table or enter another DUMPSCAN subcommand.

Commands That Detect Condition:

```
&name subcommand of DUMPSCAN
```

520I

A LOOP WAS DETECTED IN THE type CHAIN

Explanation: An unexpected loop was detected in the specified control block chain currently being processed.

System Action: If a DUMPSCAN subcommand was issued, subcommand processing is terminated with the detection of the loop. Any output that was completed before the detection of the loop will be displayed.

If the PRTDUMP command was issued, this message is printed in the output to indicate why the particular summary report was not complete. A message will be issued indicating that the summary report being processed is not complete. The printing of other dump data continues.

User Response: If you are in DUMPSCAN mode, you may enter a new subcommand.

Commands That Detect Condition:

Command	*Type*
PRTDUMP	VMDBK
CHAIN subcommand	CONTROL BLOCK
VIOBLOK subcommand	VMDBK
VMDBK subcommand	VMDBK

521I

CONTROL BLOCK AT address POINTS TO CONTROL BLOCK AT address

Explanation: The control block at the address specified points to another control block already in the chain.

System Action: Additional informational messages will be issued. Then subcommand processing terminates.

User Response: None.

Commands That Detect Condition:

```
CHAIN subcommand of DUMPSCAN
```

522I

CONTROL BLOCK #nnnn POINTS TO CONTROL BLOCK #nnnn

Explanation: The control block specified by the variable “nnnn” points to another control block specified by the second “nnnn” variable that is already in the chain.

System Action: An additional informational message may be issued. Then subcommand processing terminates.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
CHAIN subcommand of DUMPSCAN
```

523I **nnnn ENTRIES WERE FOUND IN THE CHAIN**

Explanation: This message tells you how many entries were found in the chain.

System Action: The dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
CHAIN subcommand of DUMPSCAN
```

524I **THE FIRST CONTROL BLOCK POINTS TO ITSELF**

Explanation: The first control block on a chain points to itself.

System Action: Subcommand processing terminates.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
CHAIN subcommand of DUMPSCAN
```

525I **CB # nnnn AT address**

Explanation: The control block specified by the variable "nnnn" is located at the address indicated by the variable "address".

System Action: Subcommand processing continues.

User Response: None.

Commands That Detect Condition:

```
CHAIN subcommand of DUMPSCAN
```

526I **THE LAST CONTROL BLOCK POINTS TO THE FIRST**

Explanation: The last control block on a chain points to the first control block on the same chain. This message appears for noncyclic chains only.

System Action: Subcommand processing terminates.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
CHAIN subcommand of DUMPSCAN
```

527I **THE LAST CONTROL BLOCK POINTS TO ITSELF**

Explanation: The last control block on a chain points to itself.

System Action: Subcommand processing terminates.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
CHAIN subcommand of DUMPSCAN
```

528I

CHAIN MAXIMUM OF 4096 CONTROL BLOCKS REACHED

Explanation: The maximum number of entries is 4096 in a chain for which the CHAIN subcommand is able to detect a loop. That maximum has been reached.

System Action: The system waits for a null line (the null line subcommand) or another subcommand to be entered.

User Response: If you wish to continue CHAIN subcommand processing for this chain, enter a null line (the null line subcommand). Subcommand processing then continues from the last entry found.

Commands That Detect Condition:

```
CHAIN subcommand of DUMPSCAN
```

529I

nnnn ENTRIES - PROCESSING CONTINUES

Explanation: You entered the COUNT operand on the CHAIN subcommand. The subcommand is working.

System Action: System processing continues.

User Response: None.

Commands That Detect Condition:

```
CHAIN subcommand of DUMPSCAN
```

530I

THE STRING string WAS NOT FOUND

Explanation: The string (hexadecimal or EBCDIC) was not found between the starting and ending addresses specified on the input.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
LOCATE   subcommand of DUMPSCAN
LOCATEUP subcommand of DUMPSCAN
```

540E

THE STARTING ADDRESS address MUST BE LESS THAN THE ENDING ADDRESS

Explanation: The starting address specified with the LOCATE subcommand is greater than the ending address.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Correct the input and reenter the subcommand or enter a new subcommand.

Commands That Detect Condition:

```
LOCATE subcommand of DUMPSCAN
```

541E

THE STARTING ADDRESS address MUST BE GREATER THAN THAN THE ENDING ADDRESS

Explanation: The starting address specified with the LOCATE subcommand is less than the ending address.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Correct the input and reenter the subcommand or enter a new subcommand.

Commands That Detect Condition:

```
LOCATEUP subcommand of DUMPSCAN
```

543I

THE BEGINNING OF THE DUMP WAS REACHED

Explanation: The beginning of the dump was reached.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
-        subcommand of DUMPSCAN
BACKWARD subcommand of DUMPSCAN
SCROLLUP subcommand of DUMPSCAN
```

544E

A VALID SCROLLING ADDRESS HAS NOT YET BEEN ESTABLISHED

Explanation: A FORWARD or BACKWARD request must follow some other request that generates a display address. This has not yet happened in this DUMPSCAN session.

System Action: The dump viewing facility waits for a new subcommand.

User Response: Enter an address in the area you wish to examine by using a subcommand such as DISPLAY or LOCATE, or enter a new subcommand.

Commands That Detect Condition:

```
-        subcommand of DUMPSCAN
+        subcommand of DUMPSCAN
BACKWARD subcommand of DUMPSCAN
FORWARD  subcommand of DUMPSCAN
SCROLL   subcommand of DUMPSCAN
SCROLLUP subcommand of DUMPSCAN
```

545E

SCROLLING BEYOND OFFSET offset IS INVALID

Explanation: The OFFSET value has gone beyond 0 or hex FFF0 while using one of the following subcommands: FORWARD, BACKWARD, SCROLL, SCROLLUP, +, or -.

System Action: The dump viewing facility waits for a new subcommand.

User Response: Either set the OFFSET value within the limits by entering the appropriate FORWARD, BACKWARD, SCROLL, SCROLLUP, +, or - subcommand or enter another DISPLAY subcommand.

Commands That Detect Condition:

Command	*Offset (Hex)*
+ subcommand of DUMPSCAN	0 or FFF0
- subcommand of DUMPSCAN	0 or FFF0
BACKWARD subcommand of DUMPSCAN	0 or FFF0
FORWARD subcommand of DUMPSCAN	0 or FFF0
SCROLL subcommand of DUMPSCAN	0 or FFF0
SCROLLUP subcommand of DUMPSCAN	0 or FFF0

560I THE CPU ADDRESS cpuaddr IS NOT IN THE DUMP

Explanation: You tried to display or refer to a CPU address that was not in the dump.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand or use the CPU subcommand to determine which processors are in the dump.

Commands That Detect Condition:

```
CREGS subcommand of DUMPSCAN
GREGS subcommand of DUMPSCAN
REGS  subcommand of DUMPSCAN
TRACE subcommand of DUMPSCAN
VPAIR subcommand of DUMPSCAN
VREG  subcommand of DUMPSCAN
VSTAT subcommand of DUMPSCAN
```

570I VECTOR INFORMATION IS NOT AVAILABLE FOR THIS VIRTUAL MACHINE

Explanation: The dump being viewed by DUMPSCAN contains no vector register contents or status for any virtual CPU. You have requested vector information for a virtual machine where the Vector Facility was one of the following:

- Undefined
- Never used
- Unavailable due to Vector Facility failure.

System Action: The subcommand is terminated and DUMPSCAN waits for a new subcommand to be entered.

User Response: Ensure that the correct dump is being viewed.

Commands That Detect Condition:

```
VPAIR subcommand of DUMPSCAN
VREG  subcommand of DUMPSCAN
VSTAT subcommand of DUMPSCAN
```

571I **VECTOR INFORMATION IS NOT AVAILABLE FOR THIS CPU**

Explanation: Although vector information is available in the dump being viewed, the Vector Facility was not defined or not functional for the processor requested, or no data had been placed in it.

System Action: The subcommand is terminated and DUMPSCAN waits for a new subcommand to be entered.

User Response: Issue the subcommand without specifying a CPU address. Or, correct the address specified. You can use the CPU subcommand of DUMPSCAN to obtain all processor identifiers in the dump.

Commands That Detect Condition:

```
VPAIR subcommand of DUMPSCAN
VREG  subcommand of DUMPSCAN
VSTAT subcommand of DUMPSCAN
```

576E **SPECIFYING AN ODD REGISTER IS INVALID FOR VPAIR**

Explanation: You specified an odd register as the starting or ending register for the VPAIR subcommand. Only even number register pairs can be specified.

System Action: Subcommand processing terminates. The dump viewing facility waits for a new subcommand.

User Response: Reenter the VPAIR subcommand specifying even register numbers as operands.

Commands That Detect Condition:

```
VPAIR subcommand of DUMPSCAN
```

577E **THE STARTING REGISTER IS GREATER THAN THE ENDING REGISTER**

Explanation: You entered a range of registers in which the first register was larger than the last.

System Action: Subcommand processing terminates. The dump viewing facility waits for a new subcommand.

User Response: Reenter the subcommand specifying a valid range of registers.

Commands That Detect Condition:

```
VPAIR subcommand of DUMPSCAN
VREG subcommand of DUMPSCAN
```

578E **THE STARTING ELEMENT IS GREATER THAN THE ENDING ELEMENT**

Explanation: You entered a range of elements in which the first element was larger than the last.

System Action: Subcommand processing terminates. The dump viewing facility waits for a new subcommand.

User Response: Reenter the subcommand specifying a valid range of elements.

Commands That Detect Condition:

```
VPAIR subcommand of DUMPSCAN
VREG subcommand of DUMPSCAN
```

579E **THE REGISTER OPERAND 'operand' IS GREATER THAN number CHARACTERS**

Explanation: When you entered the VPAIR or VREG subcommand, you specified characters in excess of the maximum number allowed for a register operand. For example, you entered "003" when only two characters are permitted.

System Action: Subcommand processing terminates. The dump viewing facility waits for a new subcommand.

User Response: Reenter the subcommand specifying a valid register number.

Commands That Detect Condition:

```
VPAIR subcommand of DUMPSCAN
VREG subcommand of DUMPSCAN
```

580E **THE ELEMENT OPERAND 'operand' IS GREATER THAN number CHARACTERS**

Explanation: When you entered the VPAIR or VREG subcommand, you specified characters in excess the maximum number allowed for an element operand. For example, you entered "0123" when only three characters are permitted.

System Action: Subcommand processing terminates. The dump viewing facility waits for a new subcommand.

User Response: Reenter the subcommand specifying a valid register number.

Commands That Detect Condition:

```
VPAIR subcommand of DUMPSCAN
VREG subcommand of DUMPSCAN
```

600I **NO VALID TRACE ENTRIES WERE FOUND FOR CPU ADDRESS cpuaddr**

Explanation: The value contained in control register 12 for the specified processor pointed to an invalid trace entry that did not contain hex 74 in the first byte.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
TRACE subcommand of DUMPSCAN
```

601I **NO VALID TRACE ENTRY WAS FOUND AT ADDRESS address**

Explanation: The input address pointed to an invalid trace entry that did not contain hex 74 in the first byte.

System Action: If the command was DUMPSCAN, subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

If the command was PRTDUMP, message 602I will be printed in the summary report and processing continues.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
PRTDUMP
TRACE subcommand of DUMPSCAN
```

602I **THE TRACE SUMMARY WILL CONTINUE WITHOUT CPU cpuaddr**

Explanation: An error occurred preventing the trace function of PRTDUMP from locating all or some of the trace entries for the processor indicated.

System Action: The summary printout continues without the identified processor. The summary will contain a list of all merged trace entries up to the point of error, and all except those from the identified processor after the error.

User Response: None.

Commands That Detect Condition:

```
PRTDUMP
```

603I **THE TRACE ENTRY AT address IS NOT IN THE TRACE TABLE FOR CPU cpuaddr**

Explanation: The address specified by the FROM keyword in the command is missing from either of the following:

- The trace table for the processor specified, if the command includes a single processor address
- All trace tables for all processors whose dumps were to be merged, if the command specifies or defaults to ALL.

System Action: Subcommand processing terminates. DUMPSCAN waits for a new subcommand to be entered.

User Response: Correct the FROM address or the CPU address and reenter the subcommand. Or, enter another DUMPSCAN subcommand, HX, END, or QUIT.

Commands That Detect Condition:

```
TRACE subcommand of DUMPSCAN
```

610I **THERE IS NO SELECTION CRITERIA IN EFFECT**

Explanation: You issued SELECT QUERY but you have not previously specified any selection criteria.

System Action: Subcommand processing terminates. DUMPSCAN waits for a new subcommand to be entered.

User Response: Enter any DUMPSCAN subcommand.

Commands That Detect Condition:

```
SELECT subcommand of DUMPSCAN
```

611I **THE USER 'userid' DOES NOT APPEAR IN THE DUMP**

Explanation: The userid you specified is not located in the dump file.

System Action: Subcommand processing for the requested userid is terminated. The userid you specified is not saved for future trace table entry selection.

Subcommand processing is continued for any other userid that may be requested in the same SELECT invocation.

User Response: Correct the input and reenter the SELECT subcommand or a new subcommand.

Commands That Detect Condition:

```
SELECT subcommand of DUMPSCAN
```

620I **THE VMDBK FOR USER userid DOES NOT EXIST IN THE DUMP**

Explanation: The VMDBK for the specified userid did not exist in the dump.

System Action: If a DUMPSCAN subcommand was issued, subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

If the PRTDUMP command was issued, printing of other dump data continues. This message will also be printed in the output to indicate why the virtual machine summary report for this user was not printed.

User Response: If the userid did not exist in the dump, you may issue the VMDBK subcommand of the DUMPSCAN command or you may refer to the printed VMDBK summary report to see what userids are contained in the dump. You may reissue the PRTDUMP command or, if you are in DUMPSCAN mode, you may issue a subcommand.

Commands That Detect Condition:

```
PRTDUMP
VIOBLOK subcommand of DUMPSCAN
VMDBK   subcommand of DUMPSCAN
```

621I **THERE IS NOT ENOUGH STORAGE IN AN INTERNAL TABLE FOR THE nnn ENTRIES FOUND IN THE DUMP**

Explanation: While building the internal dump viewing facility VMDBK table, the number of VMDBKs found in the global cyclic list exceeded the number of available entries in the table.

System Action: Processing continues for the subcommand, if possible.

User Response: If the command was DUMPSCAN, enter a new subcommand. This condition should be reported to your system programmer.

Commands That Detect Condition:

```
PRTDUMP
VIOBLOK subcommand of DUMPSCAN
VMDBK   subcommand of DUMPSCAN
```

622E **'field' IS NOT A RECOGNIZED VMDBK FIELD**

Explanation: The name of a VMDBK field has been specified that the VMDBK subcommand does not recognize.

System Action: The specified field name is not processed. Subcommand processing will continue to display other requested fields.

User Response: Check if the specified field name has been spelled correctly. Then check the table of valid field names supported by the dump viewing facility. Correct and reenter the subcommand or enter a new subcommand.

Commands That Detect Condition:

```
VMDBK subcommand of DUMPSCAN
```

623I **DUPLICATE USERIDS WERE FOUND IN THE VMDBK CHAIN**

Explanation: While processing the VMDBK global cyclic list, at least two VMDBKs were found with the same userid.

System Action: Processing continues, if possible.

User Response: None.

Commands That Detect Condition:

```
PRTDUMP
VIOBLOK subcommand of DUMPSCAN
VMDBK   subcommand of DUMPSCAN
```

624I **USERID userid IS A DUPLICATE USERID**

Explanation: You requested information for the specified userid. More than one VMDBK was found with this userid.

System Action: If you issued the DUMPSCAN subcommand, command processing terminates. If you issued the PRTDUMP command, no report for the requested userid is printed.

User Response: First issue the VMDBK LIST subcommand and determine the address of the VMDBK desired. Then issue the VMDBK FOR address command to display a summary of the VMDBK.

Commands That Detect Condition:

```
PRTDUMP
VIOBLOK subcommand of DUMPSCAN
VMDBK subcommand of DUMPSCAN
```

640I THE VIRTUAL type value FOR USER userid DOES NOT EXIST IN THE DUMP

Explanation: The virtual device for the device number or subchannel number specified did not exist in the dump for this user.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

Subcommand	*Type*
VIOBLOK subcommand of DUMPSCAN	DEVICE
	SUBCHANNEL

641I THE type device DOES NOT EXIST IN THE DUMP

Explanation: The device for the real device or logical device specified did not exist in the dump.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Enter a new subcommand.

Commands That Detect Condition:

Subcommand	*Type*
RIOBLOK subcommand of DUMPSCAN	LOGICAL DEVICE
	REAL DEVICE

642I NO type [FOR USER userid] COULD BE FOUND

Explanation: No I/O device blocks of the specified type could be found in the dump.

System Action: The printing of other dump data continues.

User Response: None.

Commands That Detect Condition:

Command	*Type*
PRTDUMP	REA DEVICES
	VIRTUAL DEVICES
RIOBLOK subcommand of DUMPSCAN	REAL DEVICES
	LOGICAL DEVICES
VIOBLOK subcommand of DUMPSCAN	VIRTUAL DEVICES
	VIRTUAL SUBCHANNELS

643I THE type DEVICE BLOCKS FOR DEVICES nnnn TO mmmm ARE NOT AVAILABLE

Explanation: The I/O device blocks of the specified type in the indicated range could not be found in the dump.

System Action: The printing of other dump data continues.

User Response: None.

Commands That Detect Condition:

Command	*Type*
PRTDUMP	REAL VIRTUAL

660I THE ENTRY AT address IS NOT WITHIN THE FRAME TABLE

Explanation: The address specified by the user is not the address of an entry in the frame table.

System Action: Processing continues and the storage at the frame table entry address is formatted and displayed.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
FRAMETBL subcommand of DUMPSCAN
```

661I THE ENDING ADDRESS OF THE FRAME TABLE IS BEYOND THE DUMP STORAGE SIZE OF size

Explanation: The upper boundary of the frame table exceeds the size of the dump.

System Action: If the command was DUMPSCAN, the entry is displayed, if possible.

If the command was PRTDUMP, the FRAMETBL summary is not complete.

User Response: If you are in DUMPSCAN, enter a new subcommand.

Commands That Detect Condition:

```
PRTDUMP
FRAMETBL subcommand of DUMPSCAN
```

670I

NO SNAP DATA IS PRESENT

Explanation: The soft abend dump you requested for processing contains no SNAPLIST data or snapped save areas.

System Action: DUMPSCAN processing continues.

User Response: Enter any DUMPSCAN subcommand.

Commands That Detect Condition:

```
SNAPLIST subcommand of DUMPSCAN
```

690I

THE DUMP SYMPTOM RECORD IS MISSING FROM FILE filename DUMP filemode

Explanation: The first record of the specified dump file is not a valid dump symptom record. The first two bytes should contain the characters "SR" and they do not.

System Action: If the SYMPTOM subcommand was issued, no summary information is displayed and the subcommand is terminated. The dump viewing facility waits for a new subcommand.

If the PRTDUMP command was issued, you will be prompted to specify whether printing of the dump summary reports should continue.

Note: Results of further processing are unpredictable.

User Response: IBM recommends that you verify that the specified fileid actually contains a valid VM/XA SP CP abend, stand-alone, virtual machine, or soft-abend dump. The user may continue analyzing the dump.

Commands That Detect Condition:

```
DUMPSCAN
PRTDUMP
SYMPTOM subcommand of DUMPSCAN
```

701I

THE type ADDRESS address MUST BE ON A boundary BOUNDARY

Explanation: An address needed for processing was not on the correct boundary for the type of address. The address may be one that was specified on the subcommand line, or may be an intermediary address used in processing.

System Action: If the error was encountered while processing a DUMPSCAN subcommand, subcommand processing continues if possible. If the error was encountered in PRTDUMP, the printing of other dump data will continue.

User Response: If you specified the address on a DUMPSCAN subcommand, you may correct the specified input address and reissue the subcommand, or enter a new subcommand.

Commands That Detect Condition:

Subcommand	*Type*	*Boundary*
PRTDUMP	CHRBK	DOUBLEWORD
	FRAME TABLE	4K
	RDEV	DOUBLEWORD
	SYSTEM COMMON AREA	4K
	TRACE TABLE PAGE	4K
	VDEV	DOUBLEWORD
	VMDBK	4K
FRAMETBL subcommand of DUMPSCAN	FRAME TABLE	4K
	FRAME TABLE ENTRY	16 BYTE
RIOBLOK subcommand of DUMPSCAN	CHRBK	DOUBLEWORD
	RDEV	DOUBLEWORD
	RSPBK	DOUBLEWORD
	SYSTEM COMMON AREA	4K
	VMDBK	4K
	IORBK	DOUBLEWORD
TRACE subcommand of DUMPSCAN	TRACE TABLE ENTRY	32 BYTE
	TRACE TABLE PAGE	4K
VIOBLOK subcommand of DUMPSCAN	CACBK	DOUBLEWORD
	CHRBK	DOUBLEWORD
	IORBK	DOUBLEWORD
	RDEV	DOUBLEWORD
	SYSTEM COMMON AREA	4K
	VDEV	DOUBLEWORD
	VDSBK	DOUBLEWORD
	VMDBK	4K
VMDBK subcommand of DUMPSCAN	VMDBK	4K

702I **THE type ADDRESS address EQUALS OR EXCEEDS THE DUMP STORAGE SIZE OF size**

Explanation: The address to be displayed or referenced equaled or exceeded the storage size of the dump.

System Action: If a DUMPSCAN subcommand was issued, subcommand processing continues, if possible.

If the PRTDUMP command was issued, the printing of other dump data continues.

User Response: If the address specified was on a DUMPSCAN subcommand, correct the specified input address and reenter the subcommand, or enter a new subcommand.

Commands That Detect Condition:

Command	*Type*
ADDMAP	SYSTEM VMDBK SYSTEM SEGMENT TABLE SYSTEM PAGE TABLE
PRTDUMP	ADDRESS OF RADIX TREE CHRBK FRAME TABLE REAL DEVICE BLOCK SYSTEM COMMON AREA TRACE TABLE PAGE VIRTUAL DEVICE BLOCK VMDBK

Subcommands	*Type*
- subcommand of DUMPSCAN	
+ subcommand of DUMPSCAN	
BACKWARD subcommand of DUMPSCAN	
CHAIN subcommand of DUMPSCAN	
DISPLAY subcommand of DUMPSCAN	
FINDMOD subcommand of DUMPSCAN	
FRAMETBL subcommand of DUMPSCAN	FRAME TABLE FRAME TABLE ENTRY
LOCATE subcommand of DUMPSCAN	
LOCATEUP subcommand of DUMPSCAN	
RIOBLOK subcommand of DUMPSCAN	ADDRESS OF RADIX TREE CHRBK IORBK RADIX TREE REAL DEVICE BLOCK RSPBK VMDBK
SCROLL subcommand of DUMPSCAN	
SCROLLUP subcommand of DUMPSCAN	
SNAPLIST subcommand of DUMPSCAN	
TRACE subcommand of DUMPSCAN	TRACE TABLE PAGE
VIOBLOK subcommand of DUMPSCAN	CACBK CHRBK IORBK REAL DEVICE BLOCK SYSTEM COMMON AREA VIRTUAL DEVICE BLOCK VMDBK
VMDBK subcommand of DUMPSCAN	VMDBK

703I THE type ADDRESS IS ZERO

Explanation: The address represented by "type" or mentioned in a previous message is zero.

System Action: If the error was encountered while processing a DUMPSCAN subcommand, subcommand processing continues, if possible.

If the error was encountered in PRTDUMP, the printing of other dump data continues.

User Response: If you specified the address on a DUMPSCAN subcommand, you may correct the specified input address and reenter the subcommand, or enter a new subcommand.

Commands That Detect Condition:

Subcommand	*Type*
ADDMAP	SYSTEM VMDBK SYSTEM SEGMENT TABLE SYSTEM PAGE TABLE
PRTDUMP	ADDRESS OF RADIX TREE CHRBK FRAME TABLE RDEV SYSTEM COMMON AREA VDEV VMDBK
FRAMETBL subcommand of DUMPSCAN	FRAME TABLE
RIOBLOK subcommand of DUMPSCAN	ADDRESS OF RADIX TREE RADIX TREE RDEV RSPBK SYSTEM COMMON AREA
VIOBLOK subcommand of DUMPSCAN	CACBK RADIX TREE SYSTEM COMMON AREA VDEV VDSBK VMDBK
VMDBK subcommand of DUMPSCAN	VMDBK PAGE

720I **THE type CHAIN IS BROKEN**

Explanation: The specified control block chain was not complete in the dump because either an invalid pointer was found or a necessary page was not in the dump.

System Action: If a DUMPSCAN subcommand was issued, subcommand processing is terminated with the detection of a broken chain. Any output that was completed before the detection of the broken chain is displayed.

If the PRTDUMP command was issued, the printing of other dump data continues. A message is displayed indicating that the summary report currently being printed is incomplete. The partial summary report will indicate where the chain was broken.

User Response: If you are in DUMPSCAN, you may enter a new subcommand.

Commands That Detect Condition:

Subcommand	*Type*
RIOBLOK	REAL DEVICE LOGICAL DEVICE
VIOBLOK	VMDBK VIRTUAL DEVICE VIRTUAL SUBCHANNEL
VMDBK	VMDBK

730I THE type PAGE WITH ADDRESS address IS NOT IN THE DUMP

Explanation: In attempting to locate data in the dump, a pointer referenced a page that was not in the dump.

System Action: If the error was encountered while processing a DUMPSCAN subcommand, subcommand processing continues, if possible.

If the error was encountered while in PRTDUMP, the printing of other dump data will continue.

User Response: If you specified the address on a DUMPSCAN subcommand, you may correct the specified input address and reenter the subcommand, or enter a new subcommand.

Commands That Detect Condition:

Command	*Page Type*
ADDMAP	SYSTEM PAGE TABLE
	SYSTEM SEGMENT TABLE
	SYSTEM VMDBK
PRTDUMP	CHRBK
	FRAME TABLE
	REAL DEVICE BLOCK
	SYSTEM COMMON AREA
	TRACE TABLE PAGE
	VIRTUAL DEVICE BLOCK
	VMDBK
Subcommands Page	*Type*
FRAMETBL subcommand of DUMPSCAN	FRAME TABLE
RIOBLOK subcommand of DUMPSCAN	CHRBK
	IORBK
	RADIX TREE
	RDEV
	RSPBK
	SYSTEM COMMON AREA
	VDEV
	VMDBK
TRACE subcommand of DUMPSCAN	TRACE TABLE
VIOBLOK subcommand of DUMPSCAN	CACBK
	CHRBK
	IORBK
	RDEV
	SYSTEM COMMON AREA
	VDEV
	VDSBK
	VMDBK
VMDBK subcommand of DUMPSCAN	VMDBK
All other subcommands of DUMPSCAN	

735I THE DATA IN PAGE address HAS CHANGED SINCE THE SOFT ABEND OCCURRED

Explanation: The soft abend processor found a change in the data on this page between the time the abend was issued and when the page was written to the dump spool file.

System Action: Subcommand processing continues.

User Response: The subcommand SNAPLIST can be used to determine if the data contained in those pages was preserved.

Commands That Detect Condition:

```
Any DUMPSCAN subcommand that displays storage can issue this message.
```

740I THE PREFIX PAGE COULD NOT BE FOUND

Explanation: The pointer to the prefix page was either zero, not on a 4K-byte boundary, or pointed to an address that was not in the dump.

System Action: Processing continues, if possible.

User Response: Enter a new subcommand.

Commands That Detect Condition:

```
ADDMAP
DUMPSCAN
PRTDUMP
CPU      subcommand of DUMPSCAN
FRAMETBL subcommand of DUMPSCAN
REGS     subcommand of DUMPSCAN
RIOBLOK  subcommand of DUMPSCAN
VIOBLOK  subcommand of DUMPSCAN
VMDBK    subcommand of DUMPSCAN
```

741I THE PREFIX PAGE FOR CPU cpuaddr IS NOT AVAILABLE

Explanation: The prefix page for the specified processor was not in the dump, or the page could not be read, or the prefix page could not be located.

System Action: Processing continues, if possible.

User Response: None.

Commands That Detect Condition:

```
PRTDUMP
```

750I DUMPID DATA WAS NOT PROVIDED

Explanation: The "dumpid" operand of the VMDUMP command was not used to provide descriptive text of the dump.

System Action: Subcommand processing is terminated.

User Response: None.

Commands That Detect Condition:

```
DUMPID subcommand of DUMPSCAN
PRTDUMP
```

800I

THE EXTRA OPERAND 'operand' IS IGNORED

Explanation: A valid operand (other than USER) is specified multiple times.

System Action: Printing of other dump data will continue.

User Response: None.

Commands That Detect Condition:

```
PRTDUMP
```

801E

THE OPERAND 'operand' IS NOT VALID

Explanation: The operand specified on the command line was not a recognized operand.

System Action: Command or subcommand processing is terminated, or you are prompted for the correct operand.

User Response: If prompted, enter the correct operand, or correct and reenter the request, or else enter a new request.

Commands That Detect Condition:

```
All commands and subcommands
```

802E

THE REQUIRED operand OPERAND IS MISSING

Explanation: A required operand was not specified on the command line.

System Action: Command or subcommand processing is terminated with the exception of TRACERED selection prompts.

User Response: Correct and reenter the request, or enter a new request.

Commands That Detect Condition:

```
All commands and subcommands
```

803E

TOO {MANY|FEW} FILEID OPERANDS FOR MAPTYPE type

Explanation: You specified either too many or too few fileid operands on the MAP command.

The variable "type" refers to the map type and may be any one of the following:

- CP
- CMS
- PVM
- RSCSNET
- GCS
- RSCSV2.

System Action: Command processing terminates.

User Response: Correct the operands and reenter the MAP command.

804E

REQUIRED OPERAND MISSING FOR 'keyword' KEYWORD

Explanation: An operand value is required following the specified keyword.

System Action: The subcommand is terminated and DUMPSCAN waits for a new subcommand to be entered.

User Response: Enter the HELP subcommand. Or, find the correct subcommand syntax. See "Chapter 3. Command Reference" on page 33 and "Chapter 4. DUMPSCAN Reference" on page 57 of this document for subcommand syntax. Then, correct and reenter the subcommand.

Commands That Detect Condition:

```
VPAIR subcommand of DUMPSCAN
VREG  subcommand of DUMPSCAN
VSTAT subcommand of DUMPSCAN
```

810E

CONFLICT BETWEEN OPERANDS operand1 AND operand2

Explanation: You entered the specified conflicting operands on the command line.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Reenter the subcommand correctly or enter a new subcommand.

Commands That Detect Condition:

Subcommand	*Operand1*	*Operand2*
TRACE	HEX	FORMAT
	FORMAT	HEX
	CPU	ALL
	ALL	CPU
VMDBK	LIST	any operand
	any operand	LIST
	FOR	AT
	AT	FOR
BLOCK	ALL	PROMPT

811E

THE OPERAND 'operand' MUST BE A HEXADECIMAL NUMBER

Explanation: An operand that must be a hexadecimal number contained nonhexadecimal characters.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Reenter the subcommand correctly or enter a new subcommand.

Commands That Detect Condition:

```
All commands and subcommands
```

812E

THE type OPERAND 'operand' IS GREATER THAN MAXIMUM VALUE OF value

Explanation: An operand exceeded its maximum permissible value.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Correct the input and reenter or enter a new subcommand.

Commands That Detect Condition:

```
All commands and subcommands
```

813E

THE type OPERAND 'operand' MUST BE A DECIMAL NUMBER

Explanation: In response to a prompt message for selection criteria, you specified nondecimal characters for an operand that must be a decimal number.

System Action: Your response is ignored. You will be reprompted.

User Response: Reenter the keyword and a valid decimal number. If you want to end the prompt, enter a null line.

Commands That Detect Condition:

```
TRACERED
```

814E

'operand' OPERAND SPECIFIED TWICE

Explanation: You entered the displayed operand twice while issuing a DUMPSCAN subcommand.

System Action: Subcommand processing is terminated and DUMPSCAN waits for you to enter a new subcommand.

User Response: Reenter the subcommand using the specified operand only once.

Commands That Detect Condition:

```
BLOCK subcommand of DUMPSCAN
```

900E

PROCESSING ERROR FROM name, CODE nnn

Explanation: A nonzero return code was returned from the indicated routine or macro "name" and the return code was "nnn". This indicates an internal processing error.

System Action: Command processing continues, if possible.

User Response: If the condition continues, notify your service representative.

Commands That Detect Condition:

```
Any command or subcommand
```

910E **THE VIRTUAL STORAGE CAPACITY IS EXCEEDED**

Explanation: There is not enough virtual storage available for the command or subcommand to complete processing. This may be the result of a system problem, such as storage not being properly released, or the user may have attempted to use the command in a virtual machine that was not of the required virtual size.

System Action: Subcommand processing is terminated and the dump viewing facility waits for a new subcommand.

User Response: Verify that the storage size of the virtual machine is large enough. If it is not, define a larger size and reIPL.

Commands That Detect Condition:

```
ADDMAP
DUMPSCAN
CHAIN     subcommand of DUMPSCAN
VMDBK     subcommand of DUMPSCAN
MAP
PRTDUMP
TRACERED
```

920I **THE DISK nn IS NOT ACCESSED**

Explanation: The disk containing a requested file was not accessed.

System Action: Command processing is terminated, or the user is prompted for the correct file name.

User Response: Verify that the correct disk (filemode) was specified, and that it was linked to the system. If it was correct and linked, use the CMS ACCESS command to access the disk. If prompted, enter the correct file name, or else reissue the command with the correct file name.

Commands That Detect Condition:

```
All commands and subcommands
```

930E **DISK nn IS READ ONLY**

Explanation: You are trying to write to a READ ONLY disk.

System Action: Command processing is terminated.

User Response: Reenter TRACERED and specify a different filemode.

Commands That Detect Condition:

```
TRACERED
```

950E **ERROR READING FILE filename filetype filemode, CODE nn**

Explanation: An error occurred preventing the dump viewing facility from reading the data. FSREAD failed with the CMS return code indicated.

System Action: Processing continues, if possible.

User Response: None.

Commands That Detect Condition:

```
Any command or subcommand
```

960E **ERROR PRINTING DATA, CODE nn**

Explanation: An error occurred preventing the dump viewing facility from printing the requested data. PRINTL failed with the CMS return code indicated.

System Action: Processing continues, if possible.

User Response: Refer to the appropriate system publications for the return code definition.

Commands That Detect Condition:

```
Any command or subcommand
```

970E **ERROR WRITING filename filetype filemode, CODE nnn**

Explanation: An error occurred preventing the dump viewing facility from writing the requested data. FSWRITE failed with the CMS return code indicated.

System Action: Command processing terminates.

User Response: Refer to the appropriate system publications for the return code definitions.

Commands That Detect Condition:

```
ADDMAP
DUMPSCAN
MAP
PRTDUMP
TRACERED
```

Dump Viewing Facility Message Summary by Message Number

010A ENTER TRACE ENTRY SELECTION CRITERIA, NULL LINE TO END SELECTION, OR QUIT TO END TRACERED COMMAND

015A WILL YOU BE MOUNTING ANOTHER TRSAVE TAPE? ENTER YES OR NO

020E 'operand1' AND 'operand2' ARE CONFLICTING KEYWORD OPERANDS

021E TAPE DEVICE ADDRESS INVALID

030I PROCESSING TRSAVE TAPE - VOL SEQ # nn - CREATED mm/dd/yy hh:mm:ss

031I END OF TAPE ENCOUNTERED - CURRENTLY nnnnnnn TRACE ENTRIES FORMATTED

032I WAITING FOR TAPE MOUNT

040E TIME SELECTION RANGE MAY BE SPECIFIED ONLY ONCE

041E A MAXIMUM OF SIXTEEN CPU ADDRESSES MAY BE SPECIFIED

042E A MAXIMUM OF SIXTEEN TRACE CODES MAY BE SPECIFIED

043E A MAXIMUM OF SIXTEEN VMDBK ADDRESSES MAY BE SPECIFIED

044E A MAXIMUM OF SIXTEEN RDEV NUMBERS MAY BE SPECIFIED

045E A MAXIMUM OF SIXTEEN VDEV NUMBERS MAY BE SPECIFIED

046E A MAXIMUM OF SIXTEEN USERIDS MAY BE SPECIFIED

049E LEADING ZEROS REQUIRED FOR CODE SELECTION VALUE 'value'

050E 'value' IS NOT A VALID SELECTION VALUE

051E THE keyword OPERAND 'value' WAS NOT PREVIOUSLY SELECTED

052E CP SELECTION KEYWORD(S) SPECIFIED FOR GUEST TRACE TABLES - KEYWORD(S) IGNORED

060E TAPE WAS NOT CREATED BY THE TRSAVE COMMAND

061E TAPE vdev IS NOT ATTACHED

062E PERMANENT TAPE I/O ERROR

070I NO TRACE ENTRIES FOUND MEETING THE REQUESTED SELECTION CRITERIA

071I ONE OR MORE TRACE ENTRIES MISSING FROM TRSAVE TAPE(S)

073I NO TRACE ENTRIES FOUND MEETING THE REQUESTED SELECTION CRITERIA FOR THE CURRENT INPUT TAPE

074E THE SYSTEM TRACE FILE 'filename' WAS NOT FOUND

076E THE OPERAND GTRACE IS NOT VALID FOR CP TRACE DATA

080E DUPLICATE INPUT SYSTEM TRACE FILES WERE SPECIFIED

081E SYSTEM TRACE FILE spoolid IS INVALID

082E ERROR {OPENING|READING|CLOSING} SYSTEM TRACE FILE spoolid CODE nn

083E PROCESSING SYSTEM TRACE FILES spoolid CREATED MM/DD HH:MM::SS

084I PROCESSING COMPLETE - nnnnnnn TRACE ENTRIES FORMATTED

085E SYSTEM TRACE FILENAME 'filename' HAS FILES FROM MORE THAN ONE ORIGINATOR. RE-ENTER COMMAND AND SPECIFY FILENAME WITH SPOOLID.

086E SYSTEM TRACE FILE spoolid IS CURRENTLY IN USE

088E THE SYSTEM TRACE FILE SPOOLID 'spoolid' WAS NOT FOUND

089E ONLY ONE CP TAPE OR CP SYSTEM TRACE FILE IS ALLOWED

090E DUPLICATE KEYWORDS WERE SPECIFIED

091I ONE OR MORE TRACE ENTRIES MISSING FROM SYSTEM TRACE FILE spoolid

092I TIME ZONES ARE DIFFERENT FOR SYSTEM TRACE FILE spoolid

093I TIME ZONES ARE DIFFERENT FOR TAPE-VOL SEQ# nn. FILE NOT PROCESSED.

094E THE STARTING DATE AND TIME MUST BE LESS THAN THE ENDING TIME AND DATE

095E THE SYSTEM TRACE FILE SPOOLID 'spoolid' WAS NOT FOUND WITHIN FILENAME 'filename'

096E RE-IPL CMS. TRACERED HAS A SEVERE ERROR

110E THE FILE NAME 'filename' IS NOT VALID

120A ENTER DUMP FILE NAME IN THE FORM XXXXXXXX, OR IN THE FORM XXXXXXXX FILEMODE

121A ENTER THE FILENAME FILETYPE FILEMODE OF THE INPUT type LOAD MAP, A NULL LINE, SUBSET OR HX

122A ENTER THE FILENAME FILETYPE FILEMODE OF THE OUTPUT type MODULE MAP, A NULL LINE, SUBSET OR HX

123A ENTER THE MAP TYPE, A NULL LINE, SUBSET OR HX

124A ENTER THE FILENAME FILETYPE FILEMODE OF THE INPUT CMS SEGMENT LOAD MAP, OR ENTER SUBSET, NONE OR HX

130E THE type FILE filename filetype filemode WAS NOT FOUND

135E THE FILE filename filetype filemode ALREADY EXISTS

140E THE FILE filename filetype filemode IS NOT A VALID type

145E THE DUMP FILE REQUESTED IS NOT A VALID DUMP

150I THE type FILE filename filetype filemode IS INCOMPLETE

200I PROCESSING FILE xxxxxxxx DUMP filemode

210I THE SUMMARY REPORTS HAVE BEEN PRINTED FOR FILE xxxxxxxx DUMP filemode

220I THE type SUMMARY REPORT IS NOT COMPLETE; SEE REPORT FOR DETAILS

230A DO YOU WANT PRINTING TO CONTINUE? ENTER YES OR NO

260I THE FORMATTING ROUTINE 'routine name' FOR 'type' COULD NOT BE FOUND

263E SVC 199 IS CURRENTLY IN USE

265I PRINTING FORMATTED STORAGE IS NOT SUPPORTED BY DUMP VIEWING FACILITY

266I THE ROUTINE 'routine name'FOR 'spoolid' CANNOT BE FOUND

300I THE DUMP HAS NO MODULE MAP

301I DUMP filename filetype filemode ALREADY HAS A MODULE MAP

302A DO YOU WANT TO REPLACE THE PREVIOUS MAP? ENTER YES OR NO

304E THE MODULE MAP CANNOT BE APPENDED TO A type DUMP

310E THE FILES filename filetype filemode AND filename filetype filemode ARE INCOMPATIBLE

311E THE type1 MODULE MAP FILE filename filetype filemode DOES NOT MATCH THE type2 DUMP filename filetype filemode

315E THE PVM LOAD MAP filename filetype filemode OVERLAPS A PREVIOUS MAP

320I THE MODULE MAP filename filetype filemode HAS BEEN CREATED FROM LOAD MAP filename filetype filemode

321I THE MAP filename filetype filemode HAS BEEN APPENDED TO DUMP filename

330I THE INPUT type LOAD MAP NAME IS filename filetype filemode

331I MAP TYPE IS CP

332I THE LOAD MAP filename filetype filemode HAS BEEN APPENDED TO MODULE MAP filename filetype filemode

340I DYNAMICALLY CREATING MODULE MAP

341I DISK nn IS READ ONLY, DYNAMIC MAP NOT CREATED

342I INSUFFICIENT DASD SPACE, DYNAMIC MAP NOT CREATED

350I THE MODULE OR ENTRY POINT name WAS NOT IN STORAGE AT THE TIME OF THE DUMP

351E THE ADDRESS address IS NOT IN A MODULE

352I THE MODULE OR ENTRY POINT name WAS NOT FOUND IN THE MODULE MAP

353I THE MODULE OR ENTRY POINT name IS LOCATED AT ADDRESS address

400E subcommand IS NOT A RECOGNIZED SUBCOMMAND

401I READY, DUMP TYPE IS type

402E THE subcommand SUBCOMMAND IS NOT VALID FOR type DUMPS

403E THE option OPTION IS NOT VALID FOR type DUMPS

404E THE 'operand' OPERAND IS NOT VALID FOR type DUMPS

405E THE IPCS SUBCOMMAND 'subcommand' IS NOT SUPPORTED FOR type DUMPS

446I BLOCK INITIALIZATION HAS STARTED, PLEASE BE PATIENT

447I BLOCK INITALIZATION HAS COMPLETED SUCCESSFULLY

448I BLOCK INITALIZATION HAS FAILED

449I UNABLE TO PROCESS THE BLOCK SUBCOMMAND

450I THE BLOCK TABLE 'tablename' WAS NOT FOUND
451I THE 'cbname' CONTROL BLOCK CANNOT BE FOUND IN ANY BLOCK TABLE
452I THE FIELD 'field name' WAS NOT FOUND
454E UNKNOWN CONTROL BLOCK ID, RE-ENTER BLOCK SUBCOMMAND USING CONTROL BLOCK NAME
455A ENTER FIELD NAME(S) TO BE FORMATTED, QUIT TO END SUBCOMMAND OR NULL LINE TO END PROMPT
456I TOO MANY FIELD NAMES SPECIFIED, THE EXTRA NAMES ARE IGNORED
457E BLOCK SUBCOMMAND NOT SUPPORTED FOR type DUMP
458E THE CONTROL FILE fname ftype WAS NOT FOUND
459E CANNOT LOCATE ANY OF THE BLOCK TABLE FILES

510I THE &NAME TABLE IS FULL
511E &name IS NOT IN THE &NAME TABLE
512I THE &NAME TABLE IS EMPTY
520I A LOOP WAS DETECTED IN THE type CHAIN
521I CONTROL BLOCK AT address POINTS TO CONTROL BLOCK AT address
522I CONTROL BLOCK #nnnn POINTS TO CONTROL BLOCK #nnnn
523I nnnn ENTRIES WERE FOUND IN THE CHAIN
524I THE FIRST CONTROL BLOCK POINTS TO ITSELF
525I CB # nnnn AT address
526I THE LAST CONTROL BLOCK POINTS TO THE FIRST
527I THE LAST CONTROL BLOCK POINTS TO ITSELF
528I CHAIN MAXIMUM OF 4096 CONTROL BLOCKS REACHED
529I nnnn ENTRIES - PROCESSING CONTINUES
530I THE STRING string WAS NOT FOUND
540E THE STARTING ADDRESS address MUST BE LESS THAN THE ENDING ADDRESS
541E THE STARTING ADDRESS address MUST BE GREATER THAN THE ENDING ADDRESS
543I THE BEGINNING OF THE DUMP WAS REACHED
544E A VALID SCROLLING ADDRESS HAS NOT YET BEEN ESTABLISHED
545E SCROLLING BEYOND OFFSET offset IS INVALID
560I THE CPU ADDRESS cpuaddr IS NOT IN THE DUMP
570I VECTOR INFORMATION IS NOT AVAILABLE FOR THIS VIRTUAL MACHINE
571I VECTOR INFORMATION IS NOT AVAILABLE FOR THIS CPU
577E THE STARTING REGISTER IS GREATER THAN THE ENDING REGISTER
578E THE STARTING ELEMENT IS GREATER THAN THE ENDING ELEMENT
579E THE REGISTER OPERAND 'operand' IS GREATER THAN number CHARACTERS
580E THE ELEMENT OPERAND 'operand' IS GREATER THAN number CHARACTERS

600I NO VALID TRACE ENTRIES WERE FOUND FOR CPU ADDRESS cpuaddr
601I NO VALID TRACE ENTRY WAS FOUND AT ADDRESS address
602I THE TRACE SUMMARY WILL CONTINUE WITHOUT CPU cpuaddr
603I THE TRACE ENTRY AT address IS NOT IN THE TRACE TABLE FOR CPU cpuaddr
610I THERE IS NO SELECTION CRITERIA IN EFFECT
611I THE USER 'userid' DOES NOT APPEAR IN THE DUMP
620I THE VMDBK FOR USER userid DOES NOT EXIST IN THE DUMP
621I THERE IS NOT ENOUGH STORAGE IN AN INTERNAL TABLE FOR THE nnn ENTRIES FOUND IN THE DUMP
622E 'field' IS NOT A RECOGNIZED VMDBK FIELD
623I DUPLICATE USERIDS WERE FOUND IN THE VMDBK CHAIN
624I USERID userid IS A DUPLICATE USERID
640I THE VIRTUAL type value FOR USER userid DOES NOT EXIST IN THE DUMP
641I THE type device DOES NOT EXIST IN THE DUMP
642I NO type [FOR USER userid] COULD BE FOUND
643I THE type DEVICE BLOCKS FOR DEVICES nnnn TO mmmm ARE NOT AVAILABLE
660I THE ENTRY AT address IS NOT WITHIN THE FRAME TABLE

661I THE ENDING ADDRESS OF THE FRAME TABLE IS BEYOND THE DUMP STORAGE SIZE OF size
670I NO SNAP DATA IS PRESENT
690I THE DUMP SYMPTOM RECORD IS MISSING FROM FILE xxxxxxxx DUMP filemode

701I THE type ADDRESS address MUST BE ON A boundary BOUNDARY
702I THE type ADDRESS address EQUALS OR EXCEEDS THE DUMP STORAGE SIZE OF size
703I THE type ADDRESS IS ZERO
720I THE type CHAIN IS BROKEN
730I THE type PAGE WITH ADDRESS address IS NOT IN THE DUMP
735I THE DATA IN PAGE address HAS CHANGED SINCE THE SOFT ABEND OCCURRED
740I THE PREFIX PAGE COULD NOT BE FOUND
741I THE PREFIX PAGE FOR CPU cpuaddr IS NOT AVAILABLE
750I DUMPID DATA WAS NOT PROVIDED

800I THE EXTRA OPERAND 'operand' IS IGNORED
801E THE OPERAND 'operand' IS NOT VALID
802E THE REQUIRED operand OPERAND IS MISSING
803E TOO {MANY|FEW} FILEID OPERANDS FOR MAPTYPE type
804E REQUIRED OPERAND MISSING FOR 'keyword' KEYWORD
810E CONFLICT BETWEEN OPERANDS operand1 AND operand2
811E THE OPERAND 'operand' MUST BE A HEXADECIMAL NUMBER
812E THE type OPERAND 'operand' IS GREATER THAN MAXIMUM VALUE OF value
813E TIME type OPERAND 'operand' MUST BE A DECIMAL NUMBER
814E 'operand' OPERAND SPECIFIED TWICE

900E PROCESSING ERROR FROM name, CODE nnn
910E THE VIRTUAL STORAGE CAPACITY IS EXCEEDED
920I THE DISK nn IS NOT ACCESSED
930E DISK nn IS READ ONLY
950E ERROR READING FILE filename filetype filemode, CODE nn
960E ERROR PRINTING DATA, CODE nn
970E ERROR WRITING filename filetype filemode, CODE nnn

Appendix A. RIOBLOK Field Names

Following is a list of items that appear on the screen as a result of the RIOBLOK subcommand. Each item is listed with the corresponding field name found in the dump.

There are three types of information that may be included on the screen:

- General device information–information for each device in the dump.
- Device-dependent information–additional information about spooling, direct access storage devices (DASD), and display devices.
- Active IORBK information–information about a device that had active I/O at the time the dump was taken. This information is taken from the I/O request and response block if it is available.

The general real device information displayed is:

Display Field	**Dump Field Name**
REAL DEVICE OR LOGICAL DEVICE	RDEVDEV
SUBCHANNEL	RDEVSUB
RDEV ADDRESS	
CLASS (device)*	RDEVCLAS
TYPE (device)*	RDEVTYPE
USER VMDBK ADDR	RDEVUSER
USER VMDBK USERID	VMDUSER
LOCK OWNER VMDBK ADDR	RDEVLOWN
LOCK OWNER VMDBK USERID	VMDUSER
ACTIVE IORBK ADDR	RDEVAIOR
INTERVENTION REQ'D IORBK ADDR	RDEVNXTW
NEXT IMMEDIATE IORBK ADDR	RDEVNXTI
FEATURE	RDEVFEAT
MODEL	RDEVMODL
DEDICATED VDEV ADDR	RDEVVDEV
WAIT DEVICE CPEBK ADDR	RDEVWTDV
WAITING TASK QUEUE	RDEVTSKQ
RDEVAFLG (allocation flag)	RDEVAFLG
RDEVDFLG (device dependent status)	RDEVDFLG
RDEVRFLG (error recovery control flag)	RDEVRFLG
RDEVSTAT (operation status flag)	RDEVSTAT

*This field may contain a descriptive name in addition to the hexadecimal content from the dump.

The following table lists the descriptive name for each device class along with the HCPRDEV control block bit that corresponds to it. If the device class is not recognized, only the hexadecimal data contained in the field is displayed.

Text	Bit Name	Explanation
DISPLAY	CLASGRAF	Graphic display device
SPOOL	CLASPOOL	Unit record spooling device
TAPE	CLASTAPE	Magnetic tape device
DASD	CLASDASD	Direct access storage device
SPECIAL	CLASSPEC	Special device

The device-dependent information displayed for:

A Spooling Device

Display Field	Dump Field Name
RSPBK ADDR	RDEVRSP
ACTIVE SPFBK ADDR	RSPASFB
CURRENT SPABK ADDR	RSPASFAL
RSPFLAG (device control flag)	RSPFLAG
RELATIVE SPDBK NUMBER	RSPADNUM
SPOOL CLASSES	RSPCLASS
ACTIVE FILE ID	RSPSPID
SILBK ADDR	RSPIL

Note: If the page containing the RSPBK is not in the dump file, "RSPBK INFORMATION IS NOT AVAILABLE" will be displayed in place of the above information.

DASD Devices

Display Field	Dump Field Name
SYSTEM CPVOL ENTRY	RDEVVOL
VOLUME SERIAL ID	RDEVSER
NEXT LOWER IORBK ADDR	RDEVNXTL
NEXT HIGHER IORBK ADDR	RDEVNXTH

For all display devices

Display Field	Dump Field Name
COMBK ADDR	RDEVCON
TRQBK ADDR	RDEVTRQ
RDEVSFLG (Screen control flags)	RDEVSFLG
RDEVTFLG (Terminal operation control flags)	RDEVTFLG

The active IORBK information is displayed for:

Interruption Response Block (IRB)

Display Field	Dump Field Name
FCTL (Function control flag)	IRBFCTL
ACTL (Activity control flag)	IRBACTL
DVST (Device status flag)	IRBDVST
SCST (Subchannel status flag)	IRBSCST
COUNT (Unexpired count in CCW)	IRBCNT
CCW ADDR	IRBCCWA

Operations Request Block (ORB)

Display Field	Dump Field Name
KEY (Key of I/O transfer)	ORBKEY
FPI (Format, prefetch, and response flag)	ORBFPI
LPM (Logical path mask)	ORBLPM
CPA (Channel program address)	ORBCPA

Appendix B. VIOBLOK Field Names

Following is a list of items that appear on the screen as a result of the VIOBLOK subcommand. Each item is listed with the corresponding field name found in the dump.

There are three types of information that may be included on the screen:

- General Device information–information for each device in the dump.
- Device-dependent information–additional information about spooling, direct access storage devices (DASD), and channel-to-channel adapter (CTCA) devices.
- Active IORBK information–information about a device that had active I/O at the time the dump was taken. This information is taken from the I/O request and response block if it is available.

The general virtual device information displayed is:

Display Field	**Dump Field Name**
VIRTUAL DEVICE	VDEVDEV
SUBCHANNEL	VDEVSUB
VDEV ADDRESS	
USER VMDBK USERID	VMDUSER
CLASS (device)*	VDEVCLAS
TYPE (device)*	VDEVTYPE
USER VMDBK ADDR	VDEVUSER
USER MODE	VMDMODE, VMDSTYPE
LOCK OWNER VMDBK ADDR	VDEVLOWN
ACTIVE IORBK ADDR	VDEVAIOR
PENDING INTERRUPT IORBK ADDR	VDEVPIOR
START/RESUME PENDING IORBK ADDR	VDEVNIOR
SENSE IORBK ADDR	VDEVSIOR
REAL DEVICE NUMBER	VDEVRDEV
WAITING TASK QUEUE	VDEVTSKQ
COMPLETION TASK QUEUE	VDEVENDQ
VDEVAFLG (allocation flag)	VDEVAFLG
VDEVDFLG (device dependent status)	VDEVDFLG
VDEVSTAT (operation status flag)	VDEVSTAT
VDEVWAIT (wait status control)	VDEVWAIT

*This field may contain a descriptive name in addition to the hexadecimal content from the dump.

The following table lists the descriptive name for each device class along with the HCPVDEV control block bit that corresponds to it. If the device class is not recognized, only the hexadecimal data contained in the field is displayed.

Text	Bit Name	Explanation
DISPLAY	CLASGRAF	Graphic display device
SPOOL	CLASPOOL	Unit record spooling device
TAPE	CLASTAPE	Magnetic tape device
DASD	CLASDASD	Direct access storage device
SPECIAL	CLASSPEC	Special device

The device-dependent information displayed for:

A Spooling Device

Display Field	Dump Field Name
VDSBK ADDR	VDEVVDS
VPXBK	VDEVVPX
ACTIVE SPABK ADDR	VSPSPA
CURRENT SPFBK ADDR	VSPSPF
VSPBK ADDR	VDEVVSP

Note: If the page containing the VDSBK is not in the dump file, "VDSBK INFORMATION IS NOT AVAILABLE" will be displayed in place of the above information.

DASD Devices

Display Field	Dump Field Name
MINIDISK STARTING CYLINDER	VDEVSCYL
MINIDISK ENDING CYLINDER	VDEVECYL
MDISK ADDRESS	VDEVLINK

For all Channel-to-Channel Adapters (CTCA)

Display Field	Dump Field Name
X-SIDE CACBK ADDR	VDEVCTCA
X-SIDE COMMAND CODE	CACXCMND
X-SIDE LOCKWORD	CACXLOCK
Y-SIDE CACBK ADDR	CACXYCAC
Y-SIDE COMMAND CODE	CACYCMND
Y-SIDE VDEV ADDR	CACYVDEV

Note: If the CTCA does not have a CACBK, only the line with the X-Side CACBK address will be shown containing a value of zero. If the page containing the X-Side CACBK is not available, "CACBK INFORMATION IS NOT AVAILABLE" will appear in place of the above information.

The active IORBK information displayed for:

Interruption Response Block (IRB)

Display Field	Dump Field Name
IRBFCTL (Function control flag)	IRBFCTL
IRBACTL (Activity control flag)	IRBACTL
IRBDVST (Device status flag)	IRBDVST
IRBSCST (Subchannel status flag)	IRBSCST
IRBCNT (Unexpired count in CCW)	IRBCNT
IRBCCWA (CCW address)	IRBCCWA

Operations Request Block (ORB)

Display Field	Dump Field Name
KEY (Key of I/O transfer)	ORBKEY
FPI (Format, prefetch, and response flag)	ORBFPI
LPM (Logical path mask)	ORBLPM
CPA (Channel program address)	ORBCPA

I/O Request and Response Block

Display Field	Dump Field Name
KEY	IORUKEY
FPI	IORUFPI
LPM	IORULPM

Note: If the device does not have a currently active IORBK, "NO ACTIVE IORBK" will appear in place of the above information. Also, if the page containing the active IORBK is not in the dump file, "ACTIVE IORBK INFORMATION IS NOT AVAILABLE" will appear in place of the above information.

Appendix C. Frame Status

The following charts map the bit setting in each of the four types of frame status for the FRAMETBL PRTDUMP summary. In each type, only the first bit found to be set will be summarized, and if none are applicable, that summary field will be left blank. Also, the bits are tested in the order they are shown on the chart.

Queue-Oriented Status

Text	Description
AVAILABLE	On available queue – FRMAVAIL is set
USER-OWNED	On a user owned list – FRMOWNED is set
CP-LOCK-CMD	Locked by CP LOCK command – FRMCPLOK is set
STOR-LOCKED	Locked in real storage – FRMLOCKD is set

Figure 86. Output Text for Queue-Oriented Status

Frame Serialization Status

Text	Description
TRANS	Being translated – FRMTRANS is set
STEAL	Being stolen – FRMSTEAL is set
RELSE	Being released – FRMRELSE is set

Figure 87. Output Text for Frame Serialization Status

Frame Information Status

Text	Description
SHARED	Shared storage frame – FRMSHARE is set
RCP-PAGE	User RCP page – FRMRCP is set

Figure 88. Output Text for Frame Information Status

Miscellaneous Status

Text	Description
IN-ERROR	Frame is in error – FRMERROR is set.
LAST-TRANS	Last translated, frame cannot be stolen – FRMLTRCT is set.
RECLAIM	Being reclaimed – FRMRECLM is set.
REFRESH	Refresh required – FRMFRSH is set.
NO-GIVE	Frame is being manipulated and must not be given out to. anyone by the available list manager – FRMNOGIV is set.
READ-ONLY	Read only frame – FRMRONLY is set.

Figure 89. Output Text for Miscellaneous Status

Appendix D. Dumping the Abend Dump to Tape

The following procedure outlines the commands to dump the VM/XA System Product to tape to send to IBM:

1. Use the CMS DUMPLOAD command to place the dump on your A-disk.
2. Rename the dump filename with the CMS RENAME command. For example, when you issue the DUMPLOAD command, the dump file may be named PRB00001 DUMP A. You may rename this file. For example:

   ```
   RENAME PRB00001 DUMP A CPDUMP01 DUMP A
   ```

3. Finally, copy the dump to tape using the CMS TAPE DUMP command. The format for this command is:

   ```
   TAPE DUMP filename filetype filemode
   ```

Note: For further information about CMS Commands, refer to the *VM/XA SP CMS Command Reference.*

Appendix E. The Dump Viewing Facility HCSTBL Table

How the Dump Viewing Facility Uses the HCSTBL Table

The VM/XA SP dump viewing facility has duplicated the VM/SP IPCS (Interactive Problem Control System) capability of using the component/program product-supplied formatting routines to view or print the data particular to the respective component/program product dump. To do this, the dump viewing facility has constructed a table, HCSTBL, which is similar to the IPCS supported product table.

The ability to use these formatting routines to view the VM dump is made possible by the dump viewing facility's use of the HCSTBL table. The HCSTBL table has fields associated with each supported component/program product dump type. For each particular dump type there is a field that contains the formatting routine lead module name for DUMPSCAN and another field with the name for PRTDUMP. The interface from the dump viewing facility to the formatting routines is similar to the interface provided by IPCS for the same purpose.

In addition to the formatting routine information, the HCSTBL also holds vital data for the BLOCK subcommand of DUMPSCAN. This data is contained in one of the HCSTBL fields. The field contains the 4-character identification for the control file that the BLOCK subcommand uses for each of the particular dump types. For instance, the block control file that is shipped with the dump viewing facility is for use with CP dumps. Its filename is HCS$*CP1*; the 4-character identification in this case is 'CP1 '.

The HCSTBL table is contained within a file named HCSTBL ASSEMBLE and is shipped with the dump viewing facility.

Note: The dump viewing facility uses a similar format for its HCSTBL table as the table used in VM/XA SP IPCS. This does not necessarily mean that the dump viewing facility uses all the fields in the exact manner of IPCS.

The Format of the HCSTBL Table

The table is organized by dump types supported by the dump viewing facility as follows:

- CP
- CMS
- RSCSNET
- RSCSV2
- PVM
- GCS.

For each dump type there are seven fields of data relating to that particular dump type as follows:

1. ID field: The character identification that was specified at the time of the dump (it can be up to 8 characters).
2. IDLEN field: The length of the ID minus one
3. EXT field: Not supported for the dump viewing facility

4. DMP field: The name of the lead DUMPSCAN formatting routine for the dump type
5. PRT field: The name of the lead PRTDUMP formatting routine for the dump type
6. MAP field: Not supported for the dump viewing facility
7. TYPE field: The block control file 4-character identification for the dump type.

Note: Supported dump types whose data fields contain “00” indicate that the dump viewing facility will ignore that field, that is, the PRT field for the PVM dump type is “00” and therefore the dump viewing facility assumes no PRTDUMP formatting routine specific to a PVM dump type exists.

The HCSTBL Table (part of the HCSTBL ASSEMBLE file)

Note: Modification of the CPDMP and CPPRT fields is not supported. The dump viewing facility uses the its own DUMPSCAN and PRTDUMP function to format CP dumps; no exit can be taken.

```
*********************   START OF 'CP'  ENTRY   ************************
CPID     DC    CL8'CP'             CP ID
CPIDLEN  DC    F'1'                LENGTH OF CP ID MINUS 1
CPEXT    DC    XL8'00'             CP EXTRACT ROUTINE
CPDMP    DC    XL8'00'             CP DUMPSCAN ROUTINE
CPPRT    DC    XL8'00'             NO CP PRTDUMP ROUTINE
CPMAP    DC    XL8'00'             CP MAP DEFAULT NAME
CPTYPE   DC    CL4'CP1 '           BLOCK ID FOR CP DUMPS
ENTRYEND EQU   *
*********************    END OF 'CP' ENTRY     ************************
*********************   START OF 'CMS' ENTRY   ************************
CMSID    DC    CL8'CMS'            CMS ID
CMSIDLEN DC    F'2'                LENGTH OF CMS ID MINUS 1
CMSEXT   DC    CL8'DMMCMS'         CMS EXTRACT ROUTINE
CMSDMP   DC    CL8'DMMDCM'         CMS DUMPSCAN ROUTINE
CMSPRT   DC    XL8'00'             NO CMS PRTDUMP ROUTINE
CMSMAP   DC    CL8'CMSIPCS'        CMS MAP DEFAULT NAME
CMSTYPE  DC    CL4'CMS '           BLOCK ID FOR CMS DUMPS
*********************    END OF 'CMS' ENTRY    ************************
********************* START OF 'RSCSNET' ENTRY ************************
RSCSID   DC    CL8'RSCSNET'        RSCSNET ID
RSCSIDLN DC    F'6'                LENGTH OF RSCSNET ID MINUS 1
RSCSEXT  DC    CL8'DMTZEX'         RSCSNET EXTRACT ROUTINE
RSCSDMP  DC    CL8'DMTZDS'         RSCSNET DUMPSCAN ROUTINE
RSCSPRT  DC    XL8'00'             NO RSCSNET PRTDUMP ROUTINE
RSCSMAP  DC    CL8'RSCSIPCS'       RSCSNET MAP DEFAULT NAME
RSCSTYPE DC    CL4'RSCS'           BLOCK ID FOR RSCSNET DUMPS
*********************  END OF 'RSCSNET' ENTRY  ************************
********************* START OF 'RSCSV2' ENTRY  ************************
RSCSV2ID DC    CL8'RSCSV2'         RSCSV2 ID
RSCSV2LN DC    F'5'                LENGTH OF RSCSV2 ID MINUS 1
RSCSV2EX DC    CL8'CSIIEX'         RSCSV2 EXTRACTION ROUTINE
RSCSV2DM DC    CL8'CSIIDS'         RSCSV2 DUMPSCAN ROUTINE
RSCSV2PR DC    CL8'CSIIPR'         RSCSV2 PRTDUMP ROUTINE
RSCSV2MA DC    CL8'GCSIPCS'        RSCSV2 MAP DEFAULT NAME
GCSRTYPE DC    CL4'GCRS'           BLOCK ID FOR RSCSV2 DUMPS
*********************  END OF 'RSCSV2' ENTRY   ************************
*********************   START OF 'PVM' ENTRY   ************************
PVMID    DC    CL8'PVM'            PVM ID
PVMIDLEN DC    F'2'                LENGTH OF PVM ID MINUS 1
PVMEXT   DC    CL8'DVMZEX'         PVM EXTRACT ROUTINE
PVMDMP   DC    CL8'DVMZDS'         PVM DUMPSCAN ROUTINE
PVMPRT   DC    XL8'00'             NO PVM PRTDUMP ROUTINE
PVMMAP   DC    CL8'PVMIPCS'        PVM MAP DEFAULT NAME
PVMTYPE  DC    CL4'PVM '           BLOCK ID FOR PVM DUMPS
*********************    END OF 'PVM' ENTRY    ************************
*********************   START OF 'GCS' ENTRY   ************************
GCSID    DC    CL8'GCS'            GCS ID
GCSIDLEN DC    F'2'                LENGTH OF GCS ID MINUS 1
GCSEXT   DC    CL8'CSIIEX'         GCS EXTRACTION ROUTINE
GCSDMP   DC    CL8'CSIIDS'         GCS DUMPSCAN ROUTINE
GCSPRT   DC    CL8'CSIIPR'         GCS PRTDUMP ROUTINE
GCSMAP   DC    CL8'GCSIPCS'        GCS MAP DEFAULT NAME
GCSTYPE  DC    CL4'GCS '           BLOCK ID FOR GCS DUMPS
*********************    END OF 'GCS' ENTRY    ************************
```

Modifying the HCSTBL Table

In order to change the HCSTBL table, you need to do the following:

1. Place the HCSTBL ASSEMBLE file on your A disk or other writable disk.
2. Change the data in a particular fields, but do not change the format or order of the fields.
3. Reassemble the HCSTBL ASSEMBLE file.
4. Place the HCSTBL TEXT file on your A disk.
5. Regenerate the DUMPSCAN and PRTDUMP modules.

Appendix F. Dump Viewing Facility SVC 199 Services

The dump viewing facility uses the VM/SP Interactive Problem Control System (IPCS) SVC 199 type of communication facility. This facility provides the interface between the dump viewing facility and the IPCS formatting routines used to support:

- Component/program product unique subcommands for DUMPSCAN
- Formatting routines for PRTDUMP.

The SVC services may be used by any code written to change the formatting routines to enable interaction with the dump viewing facility.

There are thirteen codes associated with SVC 199 under IPCS. However, not all are supported by the dump viewing facility.

- **Code = 10**: Not supported by the dump viewing facility

- **Code = 20**: Not supported by the dump viewing facility

- **Code = 30**: Request a work buffer

  ```
  PLIST  DS   0F
         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 4Kb 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   INSUFFICIENT STORAGE
     =8   INVALID REQUEST
  ```

- **Code = 31**: Free 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
  ```

  Frees storage previously obtained with SVC subcode 30.

  Return codes are:

  ```
  R15=0   ALL OK
     =4   ADDRESS INVALID
     =8   INVALID REQUEST
  ```

- **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 the 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
   =100 READ ERROR
```

Notes:

1. The next subcode call (40 or 41) will overlay the buffer returned by the previous invocation of subcode 40 or 41.

2. When the requested address exceeds dump storage size, R15 is set to 4 and the address of the 12K buffer returned is set to X'000000FF'.

- **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
   =100 READ ERROR
```

Notes:

1. The next subcode call (40 or 41) will overlay the buffer returned by the previous invocation of subcode 40 or 41.

2. When the requested address exceeds dump storage size, R15 is set to 4 and the address of the 12Kb buffer returned is set to X'000000FF'.

- **Code = 50**: Not supported by the dump viewing facility

- **Code = 60**: Request PRTDUMP to print a buffer that has been translated

```
PLIST  DS   0F
       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
   =500 PRINT FAILURE
```

- **Code = 70**: Request PRTDUMP to print the registers and PSWs and the appended load map.

```
PLIST  DS   0F
       DC   AL4(*-*)  RESERVED
       DC   H'70'     CODE FIELD
```

Return codes are:

```
R15=0   ALL OK
   =100 READ ERROR
   =500 PRINT FAILURE
```

Note: When the map is not appended, register 15 equals zero and the dump viewing facility message 300I is issued.

- **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
   =100 READ ERROR
   =500 PRINT FAILURE
```

Note: When the map is not appended, register 15 equals zero and the dump viewing facility message 300I is issued.

- **Code = 80**: Not supported by the dump viewing facility.

- **Code = 90**: Return to user a module and an 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
   =100 READ ERROR
```

- **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
```

The address is returned to caller.

Return codes are:

```
R15=0    ALL OK  ADDRESS RETURNED
   =2    MAP NOT PRESENT
   =4    NAME NOT FOUND IN MAP
   =100  READ ERROR
```

Appendix G. Table of Publications for IPCS DUMPSCAN Subcommands

The following table shows the supported dump types that can have IPCS DUMPSCAN subcommands issued against them and where you can find more information as how to use them:

VM Dump Type	**Publication**
CMS	Appendix H of this manual
GCS	Appendix I of this manual
PVM	*VM/SP Pass-Through Facility Logic*
RSCSNET	*VM/SP RSCS Networking Diagnosis Reference*
RSCSV2	*VM/SP RSCS Networking Version 2 Diagnosis Reference*

Appendix H. IPCS DUMPSCAN Subcommands for CMS Dumps

The following IPCS DUMPSCAN subcommands are supported by the dump viewing facility for CMS dumps.

Note: The IPCS DUMPSCAN subcommand USERMAP is not supported.

CMSPOINT Subcommand

Use the CMSPOINT subcommand to display the formatted contents of 17 pointers from CMS NUCON.

The format of the CMSPOINT subcommand is as follows:

CMSPoint	

Usage Notes: None.

Responses:

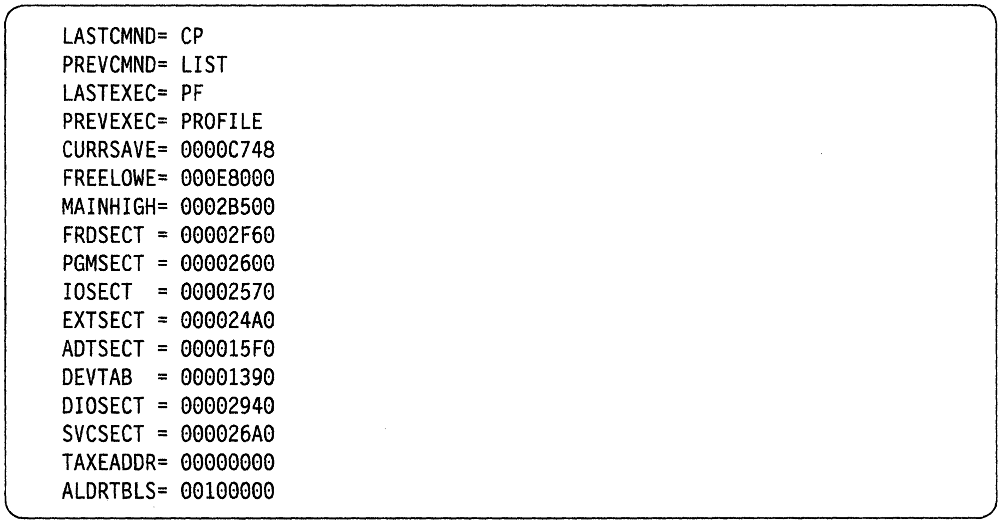
```
LASTCMND= (last command executed)
PREVCMND= (previous command executed)
LASTEXEC= (last EXEC executed)
PREVEXEC= (previous EXEC executed)
CURRSAVE= (address of current system svc
           save area)

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:

```
DMMDCC7081 PAGE 'page' NOT FOUND IN DUMP
DMMDCM863E INVALID OPERAND - operand
```

Sample Output:

This is 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
```

Figure 90. Output of the CMSPOINT Subcommand

DOSPOINT Subcommand

Use the DOSPOINT subcommand to display the formatted contents of five pointers used by DOS simulation.

The format of the DOSPoint subcommand is as follows:

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:

```
DMMDCD7281 DOS SIMULATION NOT IN EFFECT
DMMDCD708I Page 'page' NOT FOUND IN DUMP
DMMDCM863E INVALID OPERAND - operand
```

Sample Output:

```
DMMDCD728I  DOS SIMULATION NOT IN EFFECT
BGCOM  = 00000DB8
SYSCOM = 00000CA0
LTASAVE= 00001180
ACBLIST= 00000000
DOSSECT= 00000000
```

OSPOINT Subcommand

Use the OSPOINT subcommand to display the formatted contents of three pointers used in OS simulation.

The format of the OSPoint subcommand is as follows:

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
DMMDC0708I  PAGE 'page' NOT FOUND IN DUMP
```

Sample Output:

The following is an example of output of the OSPOINT subcommand.

```
CVTSECT = 00000DB8
FCBSECT = 00000000
OPSECT  = 00003BA0
```

Appendix I. IPCS DUMPSCAN Subcommands for GCS Dumps

The following IPCS DUMPSCAN subcommands are supported by the dump viewing facility for GCS dumps.

IUCV Subcommand

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.

The format of the IUCV subcommand is as follows:

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 IUCV anchor block ptr is zero.  Can't find IUCV path table
CSIIIU544I IUCV PATH table ptr is zero
```

Sample Output:

```
USER-ID-BLOCK  EXIT-ADDR  USER-WORD  TASK-BLOCK  PATH-STATUS

HHHHHH         HHHHHH     HHHHHHHH   HHHHHH      HHHH
```

TACtive Subcommand

Use the TACtive subcommand to display the task's active program list.

The format of the TACtive subcommand is as follows:

TACtive	[*taskid* ALL]

Where:

taskid
identifies the task you want information about. The 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 the 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 the OSRUN command.
- SVC blocks represent calls to the SVC interrupt handler.
- Asynchronous exit blocks exist for asynchronous exits scheduled for this task.

For every state block on the task's active program list, this subcommand also displays:

- The address of the state block
- The type of state block (link block, SVC block, or asynchronous exit block)
- The name and entry-point address of the program that the block represents
- The 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 PRT is zero.  Can't find state block

CSIIAL547I  State block PRT is zero
CSIIAL548I  Taskid table PRT is zero.  Can't find state block
```

Sample Output:

```
TASK   BLOCK   COMPLETION   BLOCK   TYPE   PROGRAM   ENTRY
 ID   ADDRESS     CODE     ADDRESS           NAME   ADDRESS

HHHH    HHHHHH   HHHHHH     HHHHHH   HH     EEEEE     HHHHHH

0001    002438   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=00009F28 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     0035B0    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
```

TLOADL Subcommand

Use the TLOADL subcommand to display the task load list.

The format of the TLOADL subcommand is as follows:

```
TLoadl     [ taskid   ALL ]
```

Where:

taskid
identifies the task you want information about. The format is nnnn.

ALL
requests information for all tasks. ALL is the default.

Usage Notes: None.

Responses:

Displays for each program loaded by this task:

- The address of the control block that contains information as to where the program is loaded
- The associated program name
- The 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  TASKID table PRT is zero.  Can't find task load list

CSIITL537I  Task block PRT is zero.  Can't find task load list
CSIITL538I  Task block list PRT is zero
```

Sample Output:

The following is an example of the output of the TLOADL subcommand.

```
TASK-ID  TASK-BLOCK  LOAD-BLOCK  PROGRAM-NAME  LOAD-COUNT

HHHH     HHHHHH      HHHHHH      EEEEEEEE      HHHH
 .          .           .           .            .
 .          .           .           .            .
 .          .           .           .            .
```

TSAB Subcommand

Use the TSAB subcommand to display the subpool map and chain header of a task.

The format of the GREGS subcommand is as follows:

TSab	[*taskid ALL*]

Where:

taskid
identifies the task you want information about. The format is nnnn.

ALL
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 subpools owned by the task
- 256-bit map of the subpools owned by the task.

Error Messages:

```
CSIITL031S  Insufficient free storage is available
CSIITL504I  Page 'nnnnnnnn' not found in dump
CSIIAL505I  TASKID 'xxxxxxxx' invalid
CSIITA539I  NUCON extension PRT is zero.  Can't find task storage anchor
CSIITA540I  TASKID table PRT is zero.  Can't find task storage anchor blo
CSIITL537I  Task block PRT is zero.  Can't find task load list
CSIITL538I  Task block list PRT is zero
```

Sample Output:

The following is an example of the output of the TSAB subcommand. The first 32 bytes of the TSAB contain the 256 bit map of the subpools owned by the task.

```
TASK-ID  TASK-BLOCK  TASK-STORAGE-ANCHOR-BLOCK  CHAIN-HEADER

HHHH     HHHHHH      HHHHHH                      HHHHHH

SUBPOOL-MAP: (CONSISTING OF 64 HEX DIGITS)
```

VMLOADL Subcommand

Use the VMLOADL subcommand to display information about all programs loaded in this virtual machine.

The format of the GREGS subcommand is as follows:

VMLoadl	

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 IDENTIFY-specified 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  The virtual machine load list is empty.
```

Sample Output:

The following is 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

MAJOR-NUCCBLK  ENTRY-NAME ENTRY-ADDRESS   TYPE

      HHHHHH   EEEEEEEE   HHHHHH          EEEEEEEE
        .          .          .              .
        .          .          .              .
        .          .          .              .
```

Summary of Changes

Second Edition

Note: Due to the extensive changes made to this book, vertical bars have not been used in the left-hand margin to denote new or changed information.

Form of Publication: SC23-0359

Level of Product: VM/XA System Product Release 2

Date of Publication: November 1988

Changes to this publication:

- **Group control system (GCS)**

  GCS is a new component that allows you to implement a native SNA communication network or run RSCS Version 2.

  This manual reflects the added ability to process GCS and RCSC Version 2 load maps (with the MAP and ADDMAP commands) as well as the load maps currently supported.

- **Soft abend dump enhancements**

  VM/XA System Product Release 2 enhances existing soft abend dump support by:

  - Providing snapshot information about abending modules' save areas and associated data areas.

    This manual documents the use of the SNAPLIST subcommand of DUMPSCAN which allows you to view a summary snaplist in a soft abend dump.

- **Dump viewing facility enhancements**

  VM/XA System Product Release 2 enhances the dump viewing facility to allow you to:

  - Format any CP control block found in a CP dump or any virtual machine control block found in a VM/XA SP dump. You can format all or just selected fields as well as specify the level of formatting detail (at the bit or byte level) you want to see. This manual documents the BLOCK subcommand of DUMPSCAN, which formats control blocks within the dump.
  - Select which trace table entries you wish to view in a CP, stand-alone, or soft abend dumps. The SELECT subcommand of DUMPSCAN documents the ability to select trace table entries you want to see and to reduce these entries through several options.
  - Process data trace information as well as CP trace table data
  - Use the XEDIT interface to extract data from dumps for use in a REXX EXEC.
  - Write your own DUMPSCAN macros using the DUMPSCAN macro subcommands FINDSTRG, READSTRG, and NOTE. Those subcommands are documented in the DUMPSCAN subcommand section of the book. The

creation of macros with these macro subcommands is documented in Chapter 2.
  - Use the new SCAN subcommand of DUMPSCAN to process a PF key assignment or command string to the system product interpreter
  - Collect GCS guest trace records in a simulated OS QSAM file. This is documented in the TRACERED command.
  - Process dumps with any CMS file name.
  - View dump data via the XEDIT interface.
  - Use an IBM-supplied EXEC to search a dump for a hung user. This is documented under the FINDUSER subcommand of DUMPSCAN.

- **New and changed publications**

  VM/XA System Product Release 2 adds two new books to the VM/XA SP library and combines two existing books into one book. The two new books are:

  - *VM/XA System Product: Group Control System Command and Macro Reference*, SC23-0433
  - *VM/XA System Product: Group Control System Diagnosis Reference*, LY27-8060.

  The combined book is *VM/XA SP Release 2 Planning and Administration*. This book replaces the *VM/XA SP Release 1 Administration* (SC23-0353) and *VM/XA SP Release 1 Planning* (GC23-0378) manuals.

- **Programming enhancements**

  VM/XA System Product Release 2 provides new and changed DIAGNOSE and IUCV functions.

Glossary

A

automatic software re-IPL. The process by which the control program attempts to restart the system after abnormal termination. This process does not involve the hardware IPL process. See also virtual = real machine recovery.

C

CCS. Console communication services.

CCW. Channel command word.

channel command word (CCW). A doubleword structure that directs an I/O operation on a device or channel and includes pointers to any storage areas associated with the operation. One or more CCWs make up a channel program.

CMS. Conversational monitor system.

console communication services (CCS). A group of CP routines that interface with the VTAM service machine, providing full VM/XA™ SP console capabilities for SNA/CCS terminal users.

control program (CP). The component of VM/XA SP that manages the resources of a single System/370-Extended Architecture system so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of either a System/370 computing system or a System/370-Extended Architecture computing system.

conversational monitor system (CMS). The component of VM/XA SP that, as a virtual machine operating system, provides interactive time-sharing. CMS allows users to communicate with the system and with each other, to create and edit files, and to develop and run application programs. It operates in either System/370 mode or 370-XA mode under the control of CP.

CP. Control program.

D

DCSS. Discontiguous saved segment.

directory. A CP disk file that includes an entry for each user in the system. The entry defines the characteristics of the user's initial virtual machine configuration. These characteristics include the userid, the password, normal and maximum allowable virtual storage, virtual device definitions, the privilege class, the dispatching priority, logical line editing characters, and the account number.

discontiguous saved segment (DCSS). A saved segment that occupies one or more architecturally-defined segments. It begins and ends on segment boundaries. It is accessed by its own name. Contrast with member saved segment. See also saved segment, segment, segment space.

dump viewing facility. A VM/XA SP component that allows users to display, format, and print data interactively from CP hard and soft abend, stand-alone, and virtual machine dumps, and to process CP trace table data stored on tape or in a system trace file.

dynamic paging area. The area of real storage allocated by CP for V = V machine paging. This area also contains CP nonresident modules, CP control blocks, CP trace tables, free storage pages, and the alternate processor's prefix storage areas.

E

Expanded Storage. Optional integrated high-speed storage. In VM/XA SP, Expanded Storage may be shared by CP and one or more virtual machines. It may also be dedicated to CP or to a particular virtual machine.

VM/XA is a trademark of the International Business Machines Corporation.

F

full-pack minidisk. A virtual disk that contains all of the addressable cylinders of a real DASD volume.

full-screen mode. In VM/XA SP, the environment in which an entire 3270 display screen is under the control of a program running in a virtual machine.

G

GCS. Group control system.

group control system (GCS). The component of VM/XA SP that, as a virtual machine supervisor, executes in a group of System/370 virtual machines under CP control to provide an interface that helps support a native Systems Network Architecture (SNA) network.

guest. An operating system running in a virtual machine managed by the VM/XA SP control program. Contrast with host.

guest real storage. The storage that appears real to the operating system running in a virtual machine. Contrast with guest virtual storage, host real storage, and host virtual storage.

guest virtual storage. The storage that appears virtual to the operating system running in a virtual machine. Contrast with guest real storage, host real storage, and host virtual storage.

H

host. The VM/XA SP control program in its capacity as manager of a virtual machine in which another operating system is running. Contrast with guest.

host real storage. The storage that appears real to the control program. If VM/XA SP is running native, this is real storage; if VM/XA SP is running in a virtual machine, this is virtual storage. Contrast with guest real storage, guest virtual storage, and host virtual storage.

host virtual storage. The storage that appears virtual to the control program. Contrast with guest real storage, guest virtual storage, and host real storage.

I

image library. A set of modules, contained in a system data file, that define the spacing, characters, and copy modification data that a 3800 printer uses to print a spool file or that define the spacing and character set that an impact printer uses to print a spool file. See also system data file.

inter-user communication vehicle (IUCV). A generalized CP interface that facilitates the transfer of data among virtual machines.

IUCV. Inter-user communication vehicle.

M

member saved segment. A saved segment that begins and ends on a page boundary. It belongs to up to 64 segment spaces and is accessed either by the segment space name or by its own name. Contrast with discontiguous saved segment. See also saved segment, segment, segment space.

message repository file. A type of system data file that contains a set of VM/XA SP messages translated into a national language.

missing interrupt handler. A CP function for detecting and dealing with real I/O operations that do not complete within a specified time.

multiple preferred guests. A VM/XA SP facility that supports up to six preferred virtual machines when the Processor Resource/Systems Manager™ (PR/SM™) feature is installed in the real machine. See also preferred virtual machine.

Processor Resource/Systems Manager and PR/SM are trademarks of the International Business Machines Corporation.

N

named saved system (NSS). A copy of an operating system that a user has named and retained in a system data file. The user can load the operating system by its name, which is more efficient than loading it by device number. See also discontiguous saved segment, member saved segment, saved segment, segment space, system data file.

NSS. Named saved system.

P

pageable virtual machine. Synonymous with virtual=virtual machine.

preferred virtual machine. A virtual machine that runs in the V=R area. CP gives this virtual machine preferred treatment in the areas of performance, processor assignment, and I/O interrupt handling. See also multiple preferred guests, virtual=fixed machine, virtual=real area, virtual=real machine.

Processor Resource/Systems Manager (PR/SM). A separately orderable feature available with 3090E processors that provides for logical partitioning of the real machine and support of multiple preferred guests. See also multiple preferred guests.

PR/SM. Processor Resource/Systems Manager.

R

real system operator. Any user who loads and runs VM/XA SP in the real machine. Contrast with virtual machine operator.

S

saved segment. One or more pages of storage that have been named and retained in a system data file. See also discontiguous saved segment, member saved segment, segment, segment space, system data file.

segment. In System/370 architecture, 64 kilobytes of storage. In 370-XA architecture, 1 megabyte of storage. See also saved segment.

segment space. A saved segment composed of up to 64 member saved segments accessed by a single name. A segment space occupies one or more architecturally-defined segments; it begins and ends on segment boundaries. A user with access to a segment space has access to all of its members. See also discontiguous saved segment, member saved segment, saved segment, segment.

service virtual machine. A virtual machine that provides system services. These services include accounting, error recording, monitoring, and those provided by supported licensed programs.

SMSG function. A CP function that allows a virtual machine to send a special message to another virtual machine programmed to accept and process the message. See also special message.

SNA. Systems Network Architecture.

SNA/CCS terminal. Any terminal accessing VM/XA SP that is managed by a VTAM service machine.

special message. A data transmission, made up of instructions or commands, sent from one virtual machine to another via the SMSG function. A special message is processed by the receiving virtual machine and does not appear on the receiver's console. See also SMSG function.

spool file. A collection of data along with CCWs for processing on a unit record device. Contrast with system data file.

SVC 76. In VM/XA SP, a supervisor call instruction that records the error incidents encountered by certain operating systems running in virtual machines. When a virtual machine operating system issues an SVC 76, VM/XA SP translates the virtual storage and I/O device addresses to real addresses, records the information on the VM/XA SP error recording virtual machine, and returns control to the issuing virtual machine. This interface bypasses the virtual machine's own error recording routine, and avoids duplicate error recording.

System/370 mode. A virtual machine operating mode in which System/370 functions are simulated. Contrast with 370-XA mode.

system data file. A collection of data associated with a particular function. Types of system data files include saved segments, NSSs, UCR files, image libraries, message repository files, and system trace files. Because a system data file contains no CCWs, it cannot be processed on a unit record device. Contrast with spool file.

system hold status. A spool file status that prevents a file from being printed, punched, or read until the real system operator releases it. Contrast with user hold status.

system trace file. A type of system data file that contains CP or virtual machine trace data.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks.

U

UCR file. User class restructure file.

unit record device. A reader, a printer, or a punch.

user class restructure file (UCR file). A type of system data file that contains information used to override the IBM-defined privilege class structure of CP commands, DIAGNOSE instruction codes, and certain CP system functions.

user directory. See directory.

user hold status. A spool file status that prevents a file from being printed, punched, or read until the file owner releases it. Contrast with system hold status.

V

Vector Facility (VF). A hardware feature that provides synchronous instruction processing for high-speed manipulation of fixed-point and floating-point data.

VF. Vector Facility.

V = F machine. Virtual = fixed machine.

virtual = fixed machine (V = F machine). A preferred virtual machine with a fixed, contiguous area of host real storage that does not start at page 0. CP provides performance enhancements for this virtual machine. See also multiple preferred guests, preferred virtual machine, virtual = real area, virtual = real machine, virtual = virtual machine.

virtual machine. In VM/XA SP, a functional equivalent of either a System/370 computing system or a System/370-Extended Architecture computing system. Each virtual machine is controlled by an operating system. VM/XA SP controls the concurrent execution of multiple virtual machines on an actual System/370-Extended Architecture system.

Virtual Machine/Extended Architecture™ System Product (VM/XA SP). An operating system that allows multiple IBM System/370 and 370-XA operating systems to run simultaneously on a single 370-XA processor. The multiple systems may be used for production, testing, developing application programs, maintenance, and migration. VM/XA SP also provides a high-capacity interactive environment. There are four components: the control program (CP), the conversational monitor system (CMS), the dump viewing facility, and the group control system (GCS).

virtual machine operator. Any user who loads and runs an operating system in a virtual machine. Contrast with real system operator.

virtual = real area (V = R area). A fixed, contiguous section of real storage, starting at page 0, in which preferred virtual machines execute. CP does not page this storage. See also preferred virtual machine, virtual = fixed machine, virtual = real machine.

virtual = real machine (V = R machine). A preferred virtual machine with a fixed, contiguous area of host real storage that starts at page 0. CP provides performance enhancements and an automatic recovery facility for this virtual machine. See also multiple preferred guests, preferred virtual machine, virtual = real area, virtual = real machine recovery, virtual = virtual machine.

virtual = real machine recovery (V = R machine recovery). A CP function that allows the V = R machine to resume operation after most CP abnormal terminations. When possible, the facility reestablishes the V = R machine environment,

Virtual Machine/Extended Architecture is a trademark of the International Business Machines Corporation.

allowing the operating system running in that virtual machine to perform its own recovery processes. See also automatic software re-IPL.

virtual = virtual machine (V = V machine). A virtual machine that runs in the dynamic paging area. CP pages this virtual machine's guest real storage in and out of host real storage. See also dynamic paging area, virtual = fixed machine, virtual = real machine.

virtual supervisor state. A condition, controlled by a virtual machine's current PSW, during which the control program allows the virtual machine to issue input/output and other privileged instructions. When these instructions are not emulated, the control program intercepts these instructions and simulates their functions for the virtual machine.

virtual wait time. The period during which the control program suspends the processing of a program while a required resource is unavailable.

VM/XA SP. Virtual Machine/Extended Architecture System Product.

VTAM service machine. A collection of networking programs running in a virtual machine that, together with the CP console communication services (CCS) routines, provide full VM/XA SP console capabilities for SNA/CCS terminal users. A VTAM service machine contains either (1) VM/VTAM with VSCS running as an application under control of GCS, or (2) VM/VCNA running as a VTAM application under control of the VSE or VS1 operating system.

V = R area. Virtual = real area.

V = R machine. Virtual = real machine.

V = R machine recovery. Virtual = real machine recovery.

V = V machine. Virtual = virtual machine.

Numerics

370 mode. Synonym for System/370 mode.

370-XA mode. A virtual machine operating mode in which System/370-Extended Architecture functions are simulated. Contrast with System/370 mode.

Bibliography

This bibliography gives the names and order numbers of microfiche and publications about VM/XA System

VM/XA System Product Microfiche

You can order microfiche listings that contain code. The order numbers for the microfiche are:

Order No.	**Description**
LYC7-0330	VM/XA System Product: CP listings
LYC7-0331	VM/XA System Product: CMS listings
LYC7-0332	VM/XA System Product: GCS listings
LYC7-0334	VM/XA System Product: dump viewing facility listings.

VM/XA System Product Publications

The publications are shown in Figure 91 on page 268. You can order any of them by their individual order numbers or you can order most of them as a group by using a single order number, SBOF-0260. SBOF-0260 provides:

- All unlicensed publications (order numbers that do not begin with LY)
- Enough three-ring binders to hold the publications
- Spine and cover inserts for the binders.

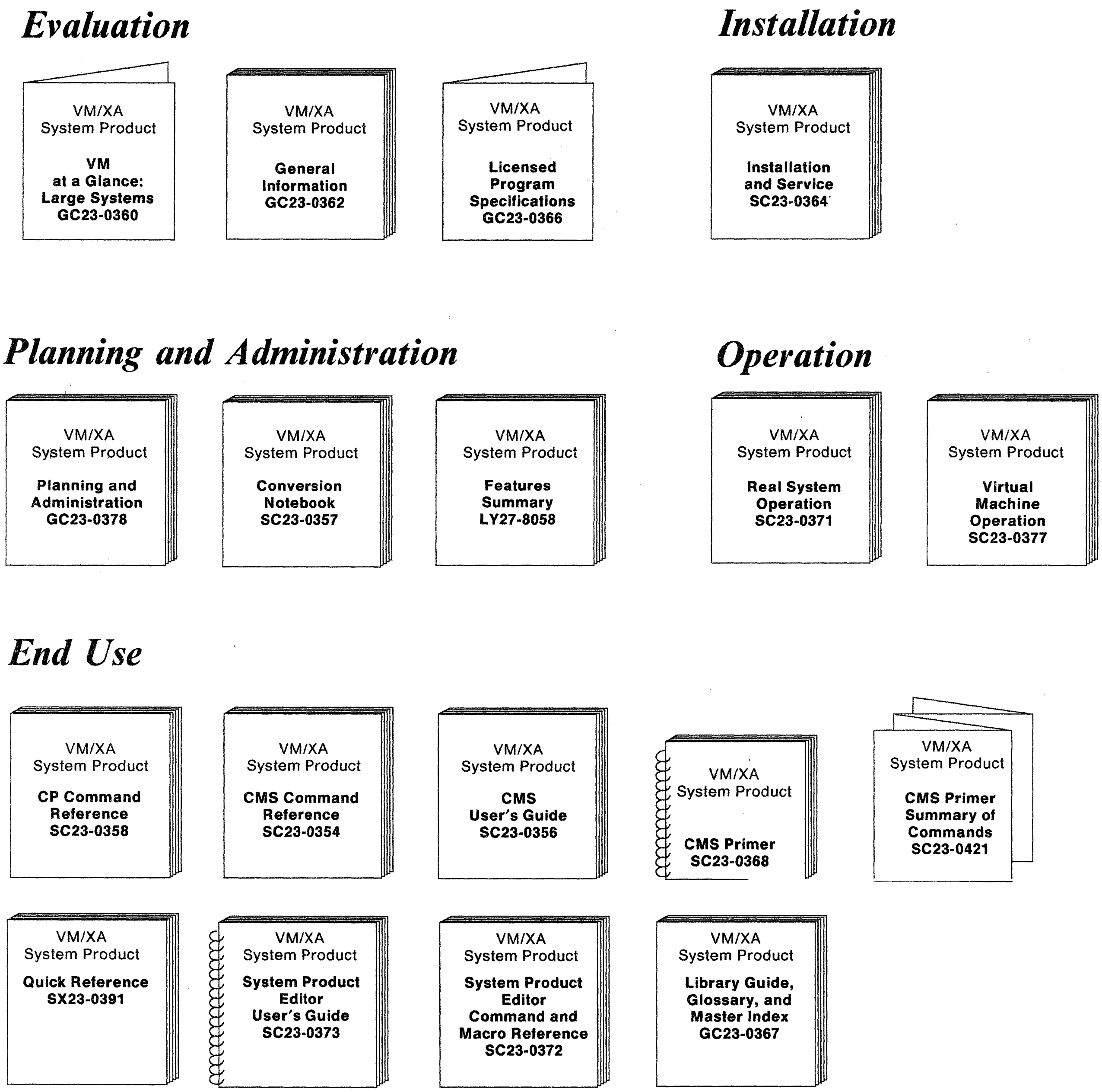
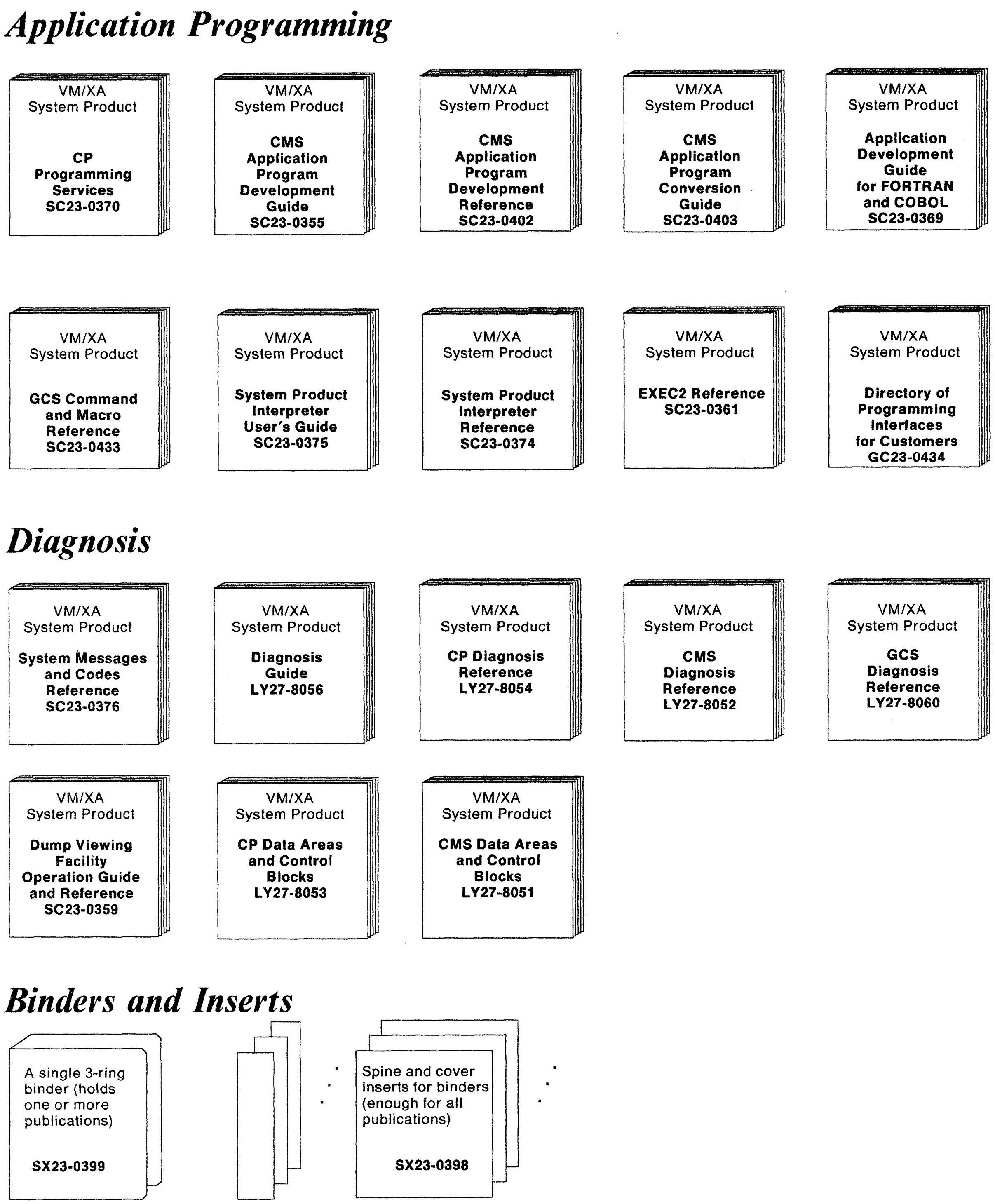

Evaluation
VM/XA System Product VM at a Glance: Large Systems GC23-0360
VM/XA System Product General Information GC23-0362
VM/XA System Product Licensed Program Specifications GC23-0366
Installation
VM/XA System Product Installation and Service SC23-0364
Planning and Administration
VM/XA System Product Planning and Administration GC23-0378
VM/XA System Product Conversion Notebook SC23-0357
VM/XA System Product Features Summary LY27-8058
Operation
VM/XA System Product Real System Operation SC23-0371
VM/XA System Product Virtual Machine Operation SC23-0377
End Use
VM/XA System Product CP Command Reference SC23-0358
VM/XA System Product CMS Command Reference SC23-0354
VM/XA System Product CMS User's Guide SC23-0356
VM/XA System Product CMS Primer SC23-0368
VM/XA System Product CMS Primer Summary of Commands SC23-0421
VM/XA System Product Quick Reference SX23-0391
VM/XA System Product System Product Editor User's Guide SC23-0373
VM/XA System Product System Product Editor Command and Macro Reference SC23-0372
VM/XA System Product Library Guide, Glossary, and Master Index GC23-0367

Figure 91 (Part 1 of 2). VM/XA System Product Publications



Application Programming

VM/XA System Product — CP Programming Services SC23-0370

VM/XA System Product — CMS Application Program Development Guide SC23-0355

VM/XA System Product — CMS Application Program Development Reference SC23-0402

VM/XA System Product — CMS Application Program Conversion Guide SC23-0403

VM/XA System Product — Application Development Guide for FORTRAN and COBOL SC23-0369

VM/XA System Product — GCS Command and Macro Reference SC23-0433

VM/XA System Product — System Product Interpreter User's Guide SC23-0375

VM/XA System Product — System Product Interpreter Reference SC23-0374

VM/XA System Product — EXEC2 Reference SC23-0361

VM/XA System Product — Directory of Programming Interfaces for Customers GC23-0434

Diagnosis

VM/XA System Product — System Messages and Codes Reference SC23-0376

VM/XA System Product — Diagnosis Guide LY27-8056

VM/XA System Product — CP Diagnosis Reference LY27-8054

VM/XA System Product — CMS Diagnosis Reference LY27-8052

VM/XA System Product — GCS Diagnosis Reference LY27-8060

VM/XA System Product — Dump Viewing Facility Operation Guide and Reference SC23-0359

VM/XA System Product — CP Data Areas and Control Blocks LY27-8053

VM/XA System Product — CMS Data Areas and Control Blocks LY27-8051

Binders and Inserts

A single 3-ring binder (holds one or more publications) SX23-0399

Spine and cover inserts for binders (enough for all publications) SX23-0398

Figure 91 (Part 2 of 2). VM/XA System Product Publications

Index

Virtual Machine/
Extended Architecture
System Product
Release 2

SYSTEM USABILITY COMMENTS

Please use this form to communicate your comments about the usability of the VM system, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the Product Usability Department for appropriate review and action, if any. Comments may be written in your own language; English is not required.

System Information

If you answer **No**, please explain.

	Yes	No
• Does the VM system meet your needs?	□	□
• Is it easy to use and understand?	□	□
• Are the commands/messages easy to understand and use?	□	□
• Are the HELP facilities appropriate?	□	□

Customer Information

- What is your occupation? ____________________
- How long have you been in this occupation? ____________________
- How long have you been using VM? ____________________
- Indicate the tasks your job involves:

Evaluation	□	Planning	□
Installation	□	Administration	□
Customization	□	Operations	□
Diagnosis	□	End Use	□
Other	□		

Your Comments:

We appreciate your comments.

If you would like a reply, please supply your name and address on the reverse side of this form. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

System Usability Comments

Fold and Tape　　Please Do Not Staple　　Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS　PERMIT NO. 40　ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 47U MS 914
Neighborhood Road
Kingston, New York 12401

Fold and Tape　　Please Do Not Staple　　Fold and Tape

If you would like a reply, *please print:*

Your Name ______________________

Company Name ______________ *Department* ________

Street Address ______________________

City ______________________

State ______________ *Zip Code* ________

IBM Branch Office serving you ______________________

IBM®

PRINTED IN U.S.A.

Virtual Machine
Extended Architecture
System Product
Release 2

Dump Viewing Facility
Operation Guide and Reference

READER'S COMMENT FORM

Order No. SC23-0359-1

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

How did you use this publication?

[ ] As an introduction
[ ] As a reference manual
[ ] For another purpose (explain) ____________________

[ ] As a text (student)
[ ] As a text (instructor)

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

What is your occupation? ____________________

Newsletter number of latest Technical Newsletter (if any) concerning this publication: ________

If you wish a reply, give your name and address: ____________________

IBM branch office serving you ____________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

SC23-0359-1

Reader's Comment Form

Fold and Tape　　Please Do Not Staple　　Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS　PERMIT NO. 40　ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 52Q MS 458
Neighborhood Road
Kingston, New York 12401

Fold and Tape　　Please Do Not Staple　　Fold and Tape

IBM®

PRINTED IN U.S.A.

Virtual Machine
Extended Architecture
System Product
Release 2

READER'S COMMENT FORM

Dump Viewing Facility
Operation Guide and Reference

Order No. SC23-0359-1

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

How did you use this publication?

[ ] As an introduction

[ ] As a reference manual

[ ] For another purpose (explain) ____________________

____

[ ] As a text (student)

[ ] As a text (instructor)

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

What is your occupation? ____________________

Newsletter number of latest Technical Newsletter (if any) concerning this publication: ________

If you wish a reply, give your name and address: ____________________

____

____

IBM branch office serving you ____________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

SC23-0359-1

Reader's Comment Form

Fold and Tape **Please Do Not Staple** **Fold and Tape**

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 52Q MS 458
Neighborhood Road
Kingston, New York 12401

Fold and Tape **Please Do Not Staple** **Fold and Tape**

IBM®

PRINTED IN U.S.A.

Program Number
5664-308

File Number
S370-37

SC23-0359-1

Printed in U.S.A.