LY20-0898-5
File No. S370-37

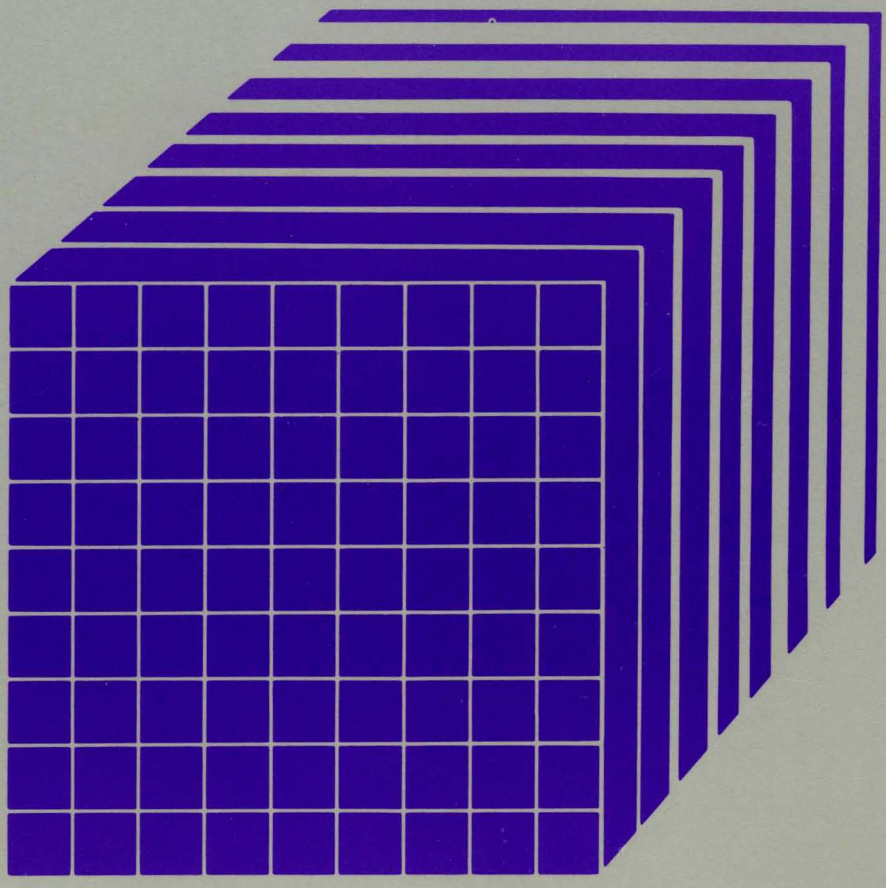
IBM

Virtual Machine/ System Product High Performance Option

Service Routines Program Logic

Release 5

LY20-0898-5



LY20-0898-5
File No. S370-37

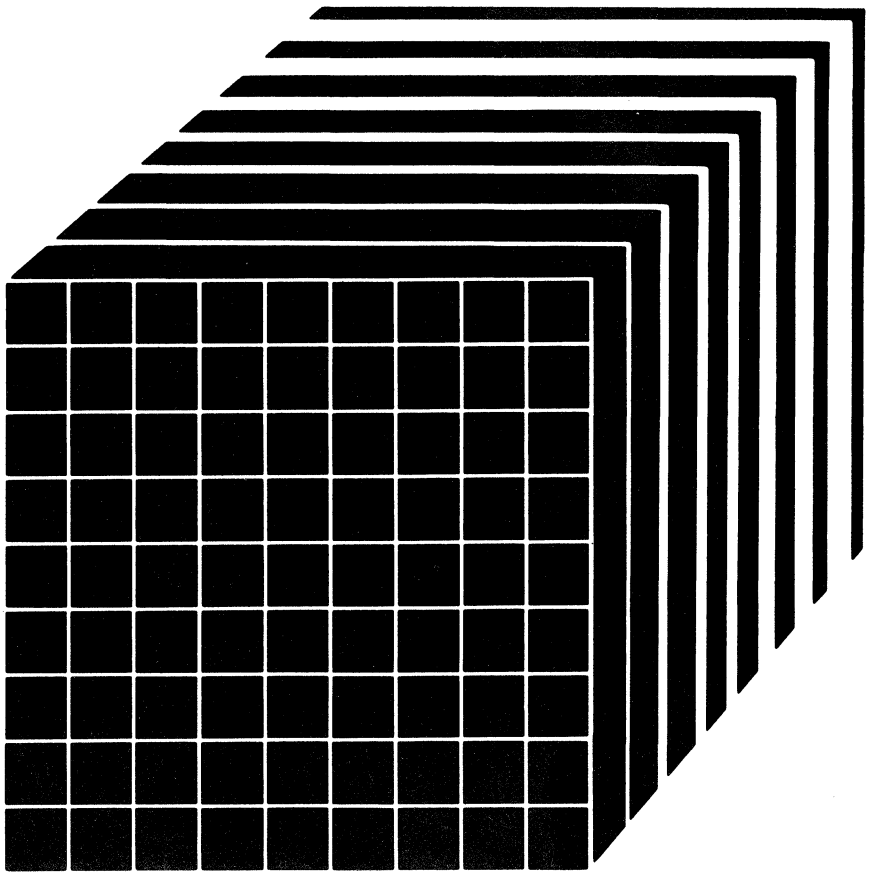
IBM

Virtual Machine/ System Product High Performance Option

Service Routines Program Logic

Release 5

LY20-0898-5



The term "VM/SP High Performance Option" applies to the VM/SP High Performance Option Licensed Program when used in conjunction with the VM/System Product Licensed Program.

Sixth Edition (August 1987)

This is a major revision of LY20-0898-4. See the Summary of Changes following the Contents for the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Release 5 of IBM Virtual Machine/System Product High Performance Option (Program Number 5664-173), and to later releases and modifications until otherwise indicated in new editions or Technical Newsletters.

To order the previous edition that still applies to Release 4.2, use the following temporary order number:

Release 4.2 Fifth Edition LT00-1913

Changes are made periodically to the information herein; before using this publication to operate IBM systems, consult the latest *IBM System/370, 30xx, and 4300 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 at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department 52Q, Neighborhood Rd., 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.

Summary of Changes

To obtain editions of this manual that pertain to earlier releases of VM/SP HPO, you must order using the pseudo-number assigned to the respective edition. For:

Release 3.6, order LY00-1624

Release 4.0, order LT00-1741

Release 4.2, order LT00-1913

Summary of Changes
for LY20-0898-5
as updated August 1987
for VM/SP HPO Release 5

NEW 'NOVF' PARAMETER ON THE 'OPTION' DIRECTORY ENTRY

New: Programming Support

Specifying the NOVF parameter on a user's OPTION control statement in the directory will deny that user access to the Vector Facility.

4381 PROCESSOR COMPLEX MODELS 11, 12, 13, AND 14

Changed: Hardware Support

The 4381 Processor Complex Models 1, 2, and 3 are replaced and extended by the Models 11, 12, 13, and 14.

NEW MODELS OF THE 3090 PROCESSOR COMPLEX

Changed: Hardware Support

In addition to supporting the 3090 Processor Complex Model 200, VM/SP HPO now supports the 3090 Processor Complex Models 150, 150E, 180, 180E, 200E, 400, and 400E (the 400 and 400E are supported in partitioned processing mode only). VM/SP HPO does **not** support the 300E and 600E.

NATIONAL LANGUAGE SUPPORT

New: Programming Support

VM/SP HPO now supports a variety of national languages. Updates have been made to modules and data areas providing this support, specifically, those handling CP messages.

DOCUMENTATION CHANGES

Minor editorial and technical changes have been made throughout this publication.

Summary of Changes for LY20-0898-4 for VM/SP HPO Release 4.2

AUTO-DEACTIVATION OF RESTRICTED PASSWORDS AND DIRECTORY ENHANCEMENTS

New: Programming Support

Adds support to enhance system integrity by minimizing the exposure of unauthorized system access through the use of restricted passwords. The directory enhancements removes the restriction on the number of USER entries that can be defined in the directory. Also, directory PROFILE support provides a means by which installations can optimize the number of commonly repeated control statements in USER entries in the source directory.

ACCESS VERIFICATION ROUTINES

New: Programming Support

While VM/SP HPO provides many security functions, added support for access verification routines provides a standard interface to the RACF/VM Support PRPQ or user-written routines that can provide a higher level of security. Although the access verification routines support does not by itself provide security functions, it allows you to install software that does.

For example, to increase security of minidisk accesses, logon passwords, and movement of spool files, you can install access verification routines with the Resource Access Control Facility (RACF) (Program Number 5740-XXH) and RACF/VM Support PRPQ (Program Number 5767-002).

Summary of Changes for LY20-0898-3 for VM/SP HPO Release 4

Note: Release 4 does not *support 3090 processors. 3090 processors are supported by Release 3 Modification 6. For information on Release 3 Modification 6, order the manual using the pseudo number shown above.*

DDR COMPACT OPTION

New: Programming Support

VM/SP HPO now supports a new option, COMPACT, for the DDR function.

3480 MAGNETIC TAPE SUBSYSTEM

New: Hardware Support

VM/SP HPO now supports the 3480 Magnetic Tape Subsystem.

3800 MODEL 3 PRINTING SUBSYSTEM

New: Hardware Support

VM/SP HPO now supports the 3800 Model 3 Printing Subsystem.

EXPANSION OF USER CLASSES

Changed: Programming Support

The user class structure has been modified such that the user may now define up to 32 privilege classes, beyond (or in place of) the seven IBM defined privilege classes. DMKOVR is documented here in a new chapter.

DOCUMENTATION CHANGES

Minor technical and editorial changes have been made throughout this publication.

Summary of Changes
for LY20-0898-2
As Updated March 31, 1985
for Release 3.6

VM/SP HIGH PERFORMANCE OPTION RELEASE 3.6

New: Expanded Programming Support

VM/SP Release 3, or an equivalent program product, is the prerequisite program product for VM/SP High Performance Option Release 3.6. VM/SP High Performance Option Release 3.6 operates with VM/SP Release 3 and incorporates the new and expanded programming facilities, features and support provided by VM/SP Release 3.

3090 Processor Support

New: Hardware Support

This release supports the 3090 as a dyadic processor.

Extended Channel Support

New: Programming Support

Extended channel support lets an installation configure its system resources over 48 channels for a 3090 Processor. As a result, many CP commands, messages, and macros now accept or return a four-digit real device address.

Paging Storage Support

New: Programming Support

Paging Storage is optional storage that can be installed on the 3090 Processor. Paging Storage support lets an installation use this storage as a high-speed, system-owned paging area. Monitor records, the DASD Dump Restore program, the page migration routine, several commands, and the SYSPAG macro have been enhanced for this support.

Control Switch Assist Extensions to Preferred Machine Assist

New: Programming Support

This support, available on 308X, 4381, and the 3090 processors, allows an MVS/SP V=R virtual machine guest (Release 1 Enhancement or later) to use IUCV, many DIAGNOSE instructions, and some Service Call instructions. It also reduces line timeout problems for such guests by letting CP reflect virtual I/O interruptions to the guest.

IBM 3880 Model 21 Storage Subsystem

New: Programming Support

The IBM 3880 Model 21 storage subsystem is similar to the Model 11, but has a larger cache, two storage directors that can be used simultaneously, and improved data transfer. The Model 21 has a different addressing scheme than the Model 11. Support for this device modifies block paging by providing sequential access. Now both the Model 11 and Model 21 can use block paging.

Miscellaneous

Changed: Documentation Only

Various technical and editorial changes have been made throughout the publication.

Preface

This publication explains the program logic for each of the VM/SP High Performance Option service routines. Because the service routines are unrelated, they are discussed separately. One chapter of this publication is dedicated to each service routine (or logical group of service routines). The Introduction describes the format of the publication in more detail.

This publication is intended for system programmers and operators whose responsibility it is to maintain a VM/SP High Performance Option system.

The "Introduction" describes the format of this publication, with special emphasis on using the method of operation diagrams.

The second chapter of this publication, "DMKIMG and DMKNMT — IEBIMAGE Interface" describes the utility programs required to dynamically change the character arrangement tables, graphic modifications, copy modifications, and FCBs for the 3800 Printing Subsystem.

The "IPCS—Interactive Problem Control System" chapter describes the logic for the commands that track and report both CP and non-CP problems.

The program that formats system residence, spooling and paging disks is described in the "Form/Allocate Service Program" chapter.

The "DMKDIR—The Directory Program" chapter describes the program that creates the directory.

The "DASD Dump Restore Program" chapter describes the program that dumps, restores, and copies system disk files.

The "Installation Verification Procedure" chapter describes the EXEC procedure that checks the accuracy of the starter or newly generated system.

The "Procedures for Generating and Updating CP and CMS" chapter describes the EXEC procedures and modules that apply updates to the system, load the system, and generate new macro libraries.

The "VM/SP HPO Starter System" chapter describes the system that is distributed to be used for system generation.

The "3704/3705 Service Program" chapter describes the programs that perform generation and service functions for the control program for the IBM 3704/3705 Communications Controllers.

The "ZAP Service Program" chapter describes the program that modifies and dumps MODULE, LOADLIB, and TXTLIB files.

The "DMSIFC and DMSREA—EREP/Error Recording Interface" chapter describes the modules that interface between CMS and the OS/VS EREP program.

The "DMKMSS—The MSS Communicator" chapter describes the program that operates in a virtual machine under OS/VS and interfaces between VM/SP and the MSS Mass Storage Control.

VM/SP High Performance Option Library

To understand the interrelationships of the publications in the VM/SP HPO library, see Figure P-1, following.

The VM/SP HPO Library

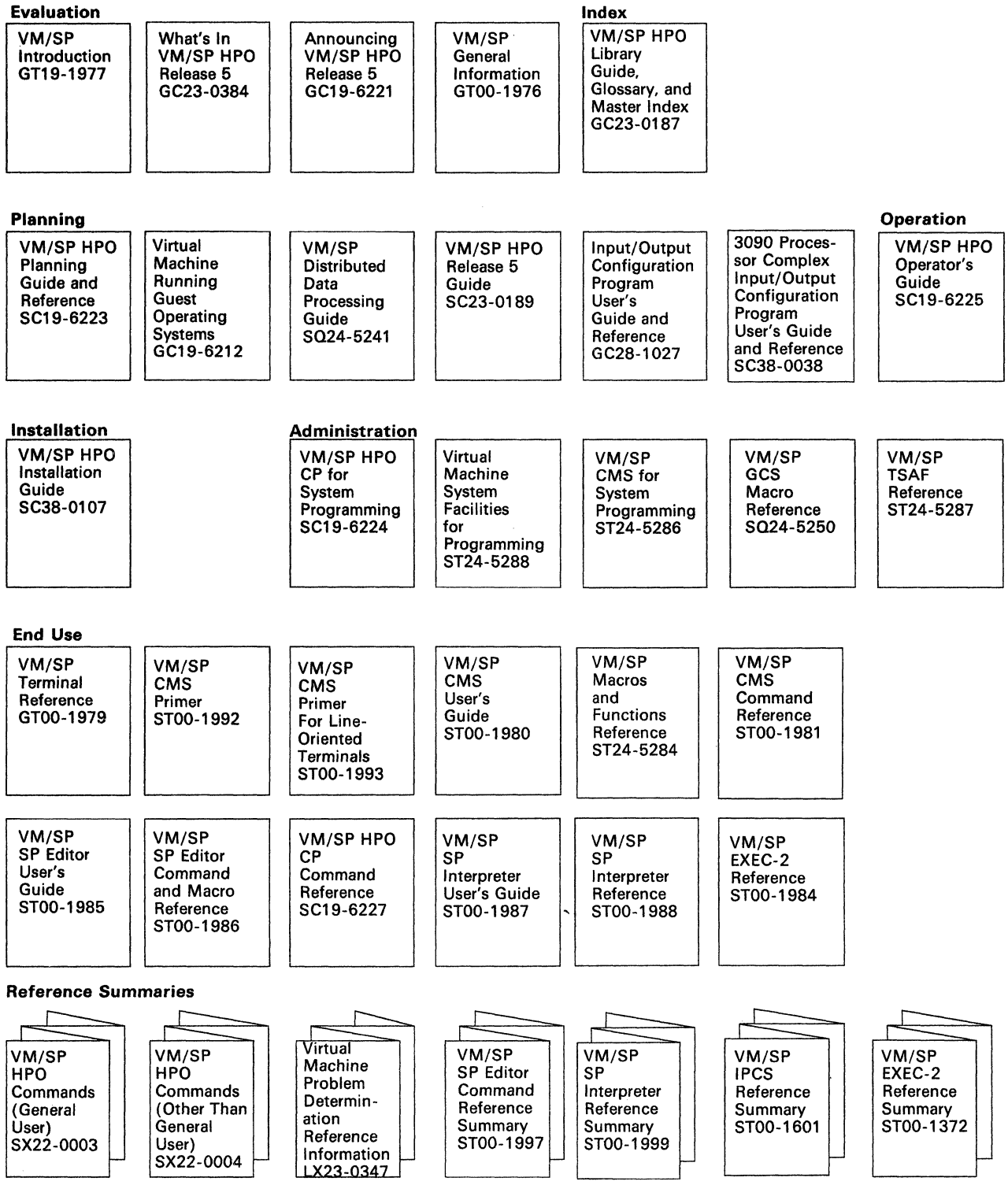


Figure P-1 (Part 1 of 2). VM/SP High Performance Option Library

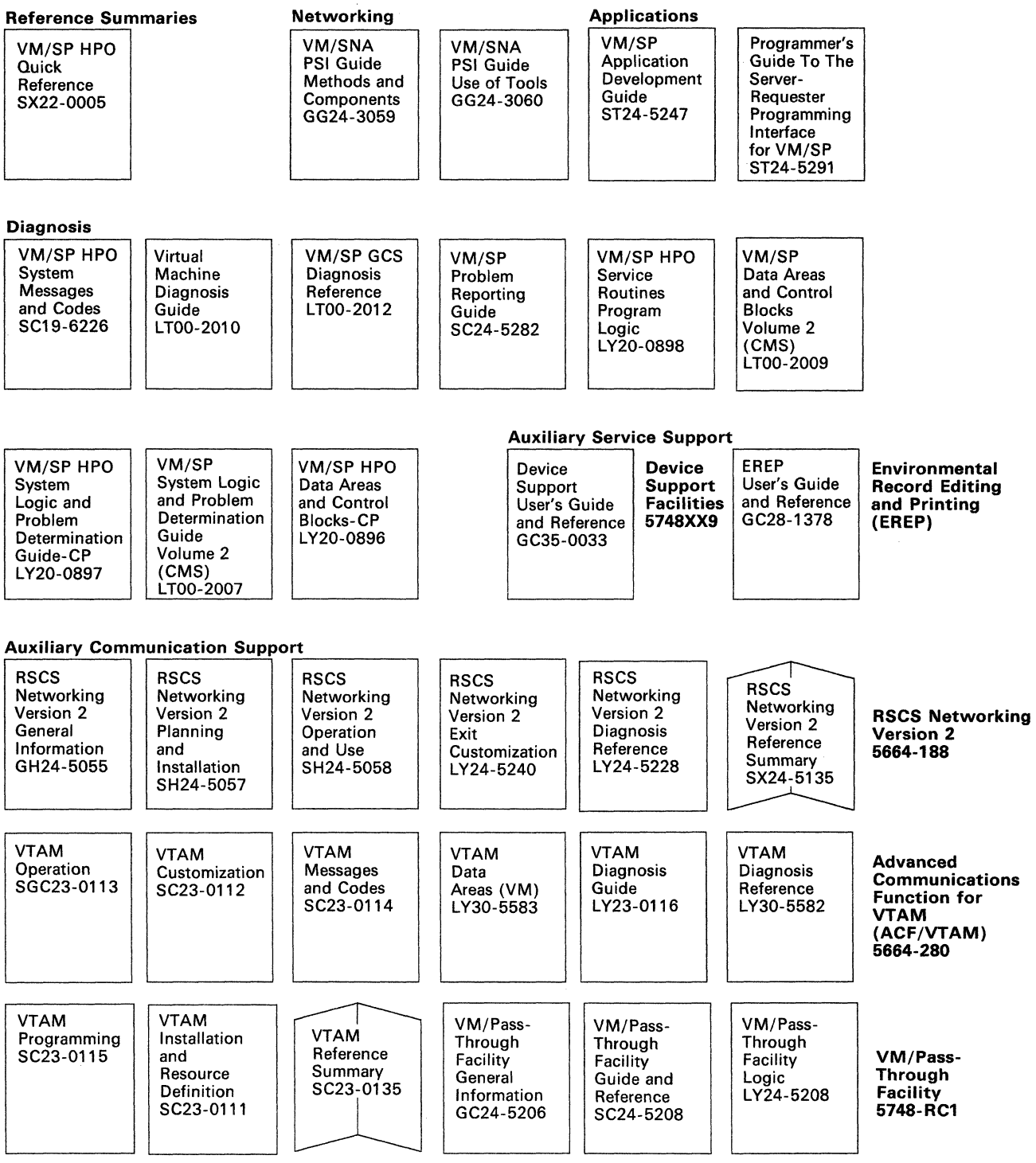


Figure P-1 (Part 2 of 2). VM/SP High Performance Option Library

Contents

Figures

Diagrams

Chapter 1. Introduction

This publication explains the program logic for each of the VM/SP HPO service routines. Because the service routines are unrelated, they are discussed separately. One chapter of this publication is dedicated to each service routine (or logical group of service routines).

Each chapter is structured similarly. The following sections, where they are applicable, are included in each chapter:

- Introduction
- Method of Operation
- Program Organization
- Directory
- Data Areas
- Diagnostic Aids.

The first section, the "Introduction," gives a brief description of the service routine. This section explains what functions the service routine performs and tells how the program can be executed.

The second section, "Method of Operation," describes the program logic for the service routine. Diagrams describe the functions that the service routine performs and the "Notes" section of each diagram relates the function performed to the coding in the program. The labels of the related program sections are identified so that you can easily find the area in the program listing.

The "Program Organization" section contains a variety of information, such as entry points, data areas, and register usage. If the service routine is complex, there is a synopsis of the program modules or program routines.

The "Directory" lists all the program labels that are mentioned in the method of operation diagrams with a cross reference list indicating the diagram on which they appear. Also, there is a brief description of the function performed at the point in the program corresponding to each label. If the service routine contains more than one module, the correct module is indicated. The "Directory" is intended to help you quickly locate section of the chapter that describes a particular function.

The "Data Areas" section contains detailed descriptions of the control blocks and data areas used by the service routine.

The last section, "Diagnostic Aids," contains a cross-reference list of the messages issued by the service routine. The message number and text are included with a label in the program reasonably close to the point where the message is issued. Messages are usually helpful when debugging a program problem.

Illustrations

There are two types of illustrations in this publication:

- Figures
- Diagrams.

Figures

All general illustrations, such as data areas and relationship drawings, are called "Figures". Figures may appear in any section of this publication.

Diagrams

The method of operation drawings are called "Diagrams". Diagrams consist of a drawing and, very often, complementary notes. The drawing has three distinct parts:

- Process
- Input
- Output.

The process block describes the action taken by the service routine. The input block shows the necessary input, such as data areas and control statements. The output block shows the resulting output, such as initialized disks or copied files. The process block is found in the center of the drawing with the input block on the left and the output block on the right. The Notes section appears below the drawing; it consists of a detailed comment, the module name (if the service routine consists of more than one module), the related program label, and a reference to any additional information (where appropriate).

Each step in the process block has a numbered key (1, 2, 3, ...) and each substep has an alphabetic key (A, B, C, ...). The related comment in the Notes section has the same key. The key that relates the processing step to a note is inside a box, and the key that relates a processing substep to a note is indented so that it is easily visible.

Illustration Numbering

Figures and diagrams are separately numbered. The format of the numbering system is:

Figure X-nn
Diagram X-nn

where X designates the chapter (one through fourteen) and nn designates the relative position of the figure or diagram within the chapter. For example,

Figure 2-3

is the third figure in Chapter 2.

Diagram 3-1

is the first method of operation diagram in Chapter 3.

This publication is intended to acquaint the system programmer, and those programmers responsible for updating VM/SP HPO service routines, with the operation of these service routines.

Refer to the *VM/SP HPO Library Guide, Glossary and Master Index*, order number GC23-0187, for unfamiliar terms used in this publication.

Note: The conversational monitor system (CMS) component of VM/SP provides a wide range of conversational and time-sharing facilities. Using CMS, you can create and manage files; and compile, test, and execute problem programs. When you install VM/SP HPO in conjunction with VM/SP, it becomes a functional operating system that provides extended features to the VM/SP control program. VM/SP HPO adds no additional functions to the CMS component of VM/SP.

Chapter 2. 3800 Utility Programs

Introduction

The GENIMAGE, IMAGELIB, and IMAGEMOD commands allow an installation to maintain 3800 printer character sets and image libraries. The GENIMAGE command (DMKIMG) creates character arrangement tables, library character sets, graphic modification modules, copy modification modules, and forms control buffers. The IMAGELIB and IMAGEMOD commands load these modules into an image library.

DMKIMG

DMKIMG, invoked by the GENIMAGE CMS command, uses the IEBIMAGE program to create TEXT images that will be used by the 3800 Printer Model 1 or Model 3.

DMKNMT

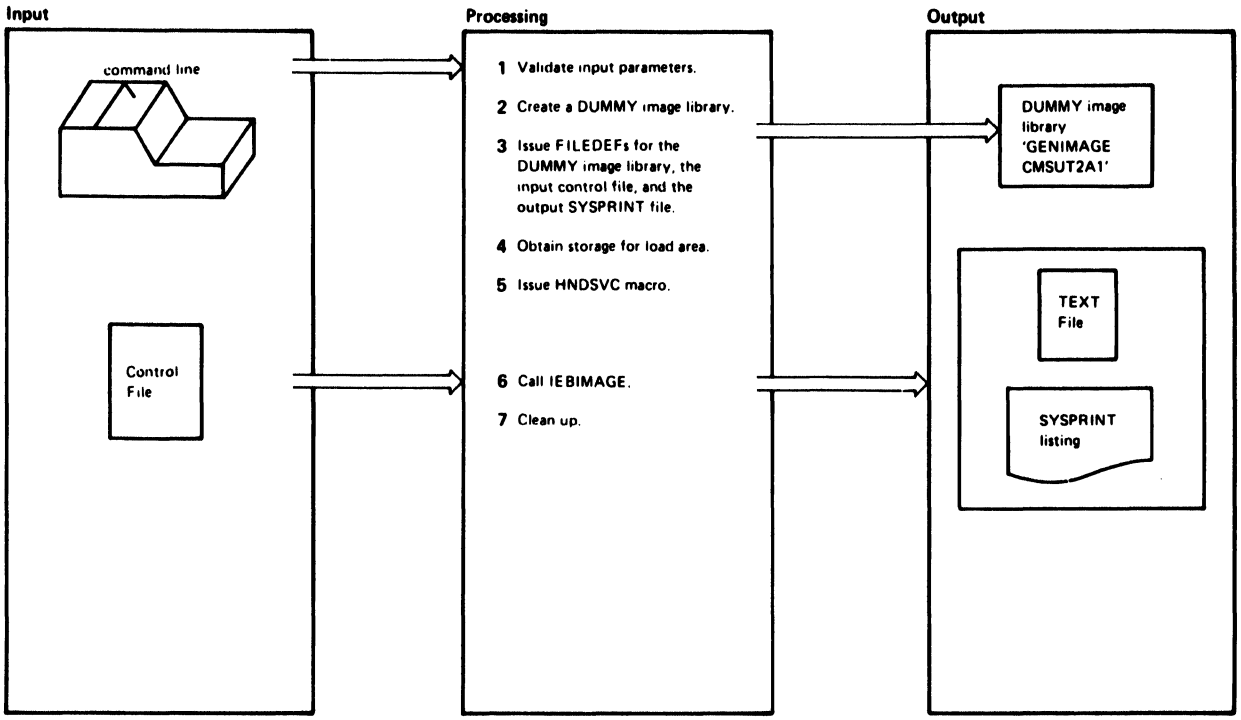
The IMAGELIB program (module DMKNMT) invoked by the IMAGELIB command, loads the necessary TEXT decks into the named system allocated at system generation time.

An installation can modify an existing 3800 named system using the IMAGEMOD command. This command is described in *VM/SP System Logic and Problem Determination Guide Volume 2 (CMS).*

Note: Due to the change in pel density, customized 3800 Model 1 character sets are not interchangeable with the 3800 Model 3 character sets. Users may recode customized 3800 Model 1 character sets and build new modules through the use of the GENIMAGE command. The MVS Character Conversion Aid may also be used to convert existing customized character sets to the 3800 Model 3 pel density.

Method of Operation

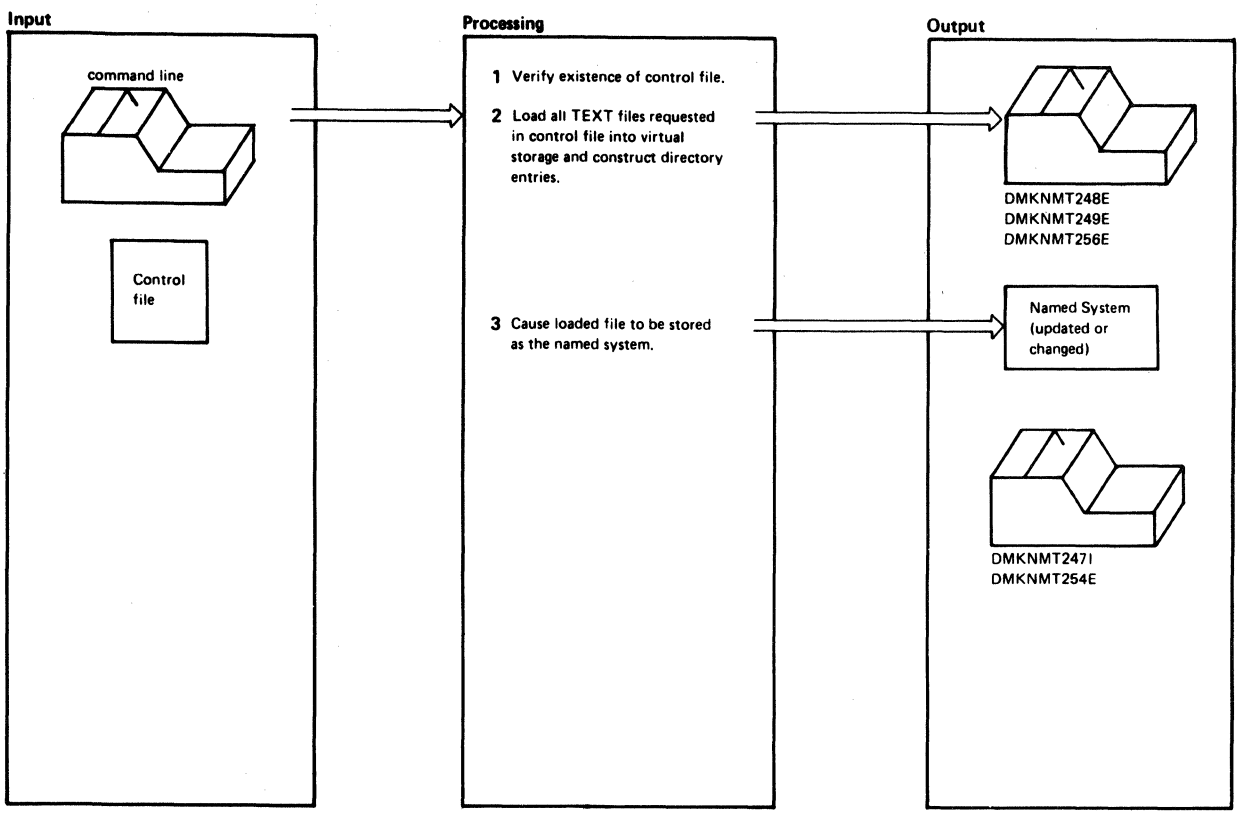
This section describes modules DMKIMG and DMKNMT. Diagrams 2-1 and 2-2 describe the functions of these modules and serve as a guide to the program listings. The labels shown indicate the closet label to the function being documented. Use the Directory and Program Organization sections to find the labels in the program listings for any routines that are not shown in the Method of Operation section.



Note	Module	Label	Ref
1 GENIMAGE command parameters are validated. If a parameter is invalid, issue return code 100.	DMKIMG	LOOP1 LOOP2 PARMERR	
2 Create a DUMMY image library.		ENDPARMS	
3 Issue a FILEDEF command with the AUXPROC option for the DUMMY image library created in Step 2; this traps all READ and WRITE operations on that data set. If any FILEDEF errors occur, issue a return code of 104.		FILEBAD	
4 Issue a GETMAIN for a 73,000 byte area for simulating OS LOAD macros.			
5 Issue HNDSVC macro to handle the following SVCs: • SVC 8 (LOAD) • SVC 18 (BLDL) • SVC 21 (STOW)			
6 Call IEBIMAGE • Issue a CMS STATE command for the TEXT file being searched for and set appropriate return codes in SVC save area.		BLDLRTN	
• Issue CMS LOAD for requested module and return the address of the area loaded into, to the issuer of the LOAD command.		LOADRTN	
• Use CMS LOAD command to get module into LOAD area and move data into user-supplied buffer for the READ.		READRTN	
• Treat as no-op and return to issuer.		WRITERTN READEXIT	

Notes	Module	Label	Ref
• Simulate operation of STOW macro by locating the module data in the IEBIBLKS work area and create a TEXT deck from it. 1) Create ESD (external symbol directory) card and write to TEXT file (GENIMAGE CMSUT1) 2) Create all necessary TXT cards and write to TEXT file (GENIMAGE CMSUT1) 3) Create an END card and write to TEXT file (GENIMAGE CMSUT1)		STOWRTN TXTLOOP	
7 Erase old TEXT file (if one existed) and rename GENIMAGE CMSUT1 to a TEXT file named IEBIMAGE.			

Diagram 2-1. DMKIMG



Notes	Module	Label	Ref
1 Verify the existence of the control file. If it doesn't exist, give a return code of 4.	DMKNMT	IMAGELIB ERR004	
2 Create a DUMMY directory that will be used to hold the number of entries in the named system.			
Read a record from the control file and verify the existence of the indicated TEXT file. If it doesn't exist, issue message DMKNMT248E.		RDLOOP AFTERRD NOTEXT	
Load the TEXT file into the CMS transient area. If a LOAD error occurs, issue message DMKNMT256E.		LDERR	
Move the file from the transient area to the core image area if sufficient storage exists. If not, issue message DMKNMT256E.			
Create a new directory entry for this TEXT file and return to RDLOOP. If no more entries, close the control file, compress the core image, and adjust the displacements in the directory.		RANOUT RDEOF DSPLOOP	
3 Issue DIAGNOSE X'74' to cause the named system to be saved. If successful, issue message DMKNMT2471; if not successful, issue message DMKNMT254E.		DIAGERR	

Notes	Module	Label	Ref

Diagram 2-2. DMKNMT

Program Organization

This section includes program descriptions of modules DMKIMG and DMKMNT.

DMKIMG

Provides a CMS interface for the VS-based IEBIMAGE program by handling certain SVCs issued by IEBIMAGE and translating them into CMS terms.

Entry Point

DMKIMGBG

Routines Called

FSSTATE - Determines if control file exists.
HNDSVC - Traps certain SVCs issued by IEBIMAGE.
GETMAIN - Gets area for simulating OS LOAD SVC.
FREEMAIN - Releases OS LOAD area.
FILEDEF - Issues FILEDEFs needed by IEBIMAGE.
LOAD - Simulates OS LOAD and QSAM READ.
FSWRITE - Creates a new TEXT file (STPW simulation).

Attributes

Disk resident, loaded into CMS user area, called via SVC 202, serially reusable.

Registers at Entry

R1: Standard CMS PLIST
R14: Return address
R15: Address of GENIMAGE

Registers at Exit

R15: Return code 100 for normal IEBIMAGE execution
R15: Return code 100 if error in input parameters
R15: Return code 104 if error during FILEDEF

External References

MAINHIGH - Saves and restores its value between loads.

DMKNMT

Constructs an image library from TEXT files on user disks and creates or replaces that image library via DIAGNOSE code X'74'.

Entry Point

DMKNMTBL

Routines Called

FSSTATE - Determines if CNTRL and TEXT files exist.
FSREAD - Reads in the control file.
CMS LOAD - Loads the TEXT file into the transient area.

Attributes

Disk resident as "IMAGELIB", loaded into CMS user area, called via SVC 202, serially reusable.

Registers at Entry

R1: Standard CMS PLIST

Register at Exit

Register 15 contains a return code:

Return Code	Meaning
0	Image library updated successfully
4	Control file not found or in error
8	Specified image non-existent
12	Specified image caused LOAD error
16	Insufficient virtual storage
20	Image library is currently active
100+	Error in FSREAD

Register Usage

R0: Temporary work register
R1: PLIST register and temporary work register
R2: Source address for MVCL
R3: Source length for MVCL
R4: Target address for MVCL
R5: Target length for MVCL
R6: Current end of image library in storage
R7: Pointer to next available directory entry
R8: Running counter for number of directory entries
R9: Starting address of the image library in storage
R12: DMKNMT module base
R14: BALR return address and scratch register
R15: BALR branch address and scratch register

External References

None

Directory

Figures 2-1 and 2-2 list, in alphabetical order, the major labels in modules DMKIMG and DMKNMT respectively. The figures indicate the associated method of operation diagrams and describe the operation performed at the point in the program associated with each label.

Label	Diagram	Description
BLDL2	2-1	Checks for file.
BLDL3	2-1	
BLDRET	2-1	Return to user key.
-ENDPARMS	2-1	Creates DUMMY image library.
-FILEBAD	2-1	Issues FILEDEF error.
GETSEQ	2-1	Obtains current value of sequence number.
LOADRTN	2-1	Simulates LOAD functions.
LOOP1	2-1	Validates parameter list.
LOOP2	2-1	Validates options.
MOVETXT	2-1	
OPTIONS	2-1	Scans through options.
PARMERR	2-1	Gives return code 100 for parameter error.
READEXIT	2-1	Issue return codes from READ.
READRTN	2-1	Simulates READ functions.
RETURN	2-1	Saves return code.
STOWRTN	2-1	Simulates STOW functions.
TXTLOOP	2-1	Creates TXT cards.
WRITERTN	2-1	Simulates WRITE functions.

Figure 2-1. DMKIMG Label Directory

Label	Diagram	Description
AFTERRD	2-2	Saves the name of the control file.
DIAGERR	2-2	Issue error message DMKNMT254E.
DSPLOOP	2-2	Adjusts old displacement in directory entries.
ERR004	2-2	Issues return code of 4.
LDERR	2-2	Issues error message DMKNMT249E.
NOTEXT	2-2	Issues error message DMKNMT248E.
RANOUT	2-2	Issues error message DMKNMT256E.
RDEOF	2-2	Saves file name for CLOSE.
RDERR	2-2	Checks for end-of-file.
RDLOOP	2-2	Points to file name.
RETURN	2-2	Obtains return address.

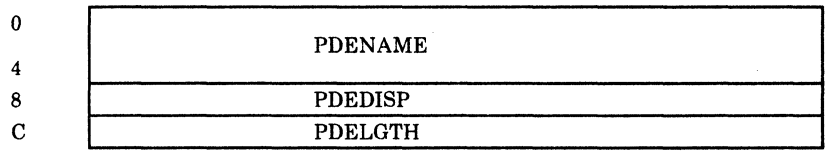
Figure 2-2. DMKNMT Label Directory

Data Areas

The following data areas are used by DMKIMG:

- Data Control Block (DCB)
- Data Extent Block (DEB)
- Data Extent Control Block (DECB)

The above data areas, except PDEBLOK, are described in the *OS/VS2 System Programming Library: Debugging Handbook, Volume 2,* Order No. GC28-0988. PDEBLOK is described in Figure 2-3.



Displacement Hex	Dec	Field Name			Description
0	0	PDENAME	DS	CL8	Member name
8	8	PDEDISP	DS	1F	RBA of start of member
0C	12	PDELGTH	DS	1F	Length of member in bytes

Figure 2-3. PDEBLOK Directory Entry for Named System

Diagnostic Aids

Figure 2-4 lists the messages issued by DMKNMT. The nearest label and the associated method of operation diagram are identified.

Message Code	Label	Diagram	Message Text
DMKNMT247I	RETURN	2-2	3800 NAMED SYSTEM imag 3800 CREATED
DMKNMT248E	NOTEXT	2-2	SPECIFIED IMAGE image NON-EXISTENT
DMKNMT249E	LDERR	2-2	ERROR LOADING IMAGE image
DMKNMT254E	DIAGERR	2-2	ERROR SAVING imag 3800 - RC = (return code)
DMKNMT256E	RANOUT	2-2	INSUFFICIENT VIRTUAL STORAGE

Figure 2-4. DMKNMT Messages

Chapter 3. IPCS—The Interactive Problem Control System

Introduction

The VM/Interactive Problem Control System Extension (VM/IPCS Extension) licensed program can be ordered separately. It is not to be misconstrued with the Interactive Problem Control System (IPCS) component of VM/370. VM/IPCS Extension provides installations with expanded facilities for reporting and diagnosing software failure. If you have installed this program, see the *VM/Interactive Problem Control System Extension User's Guide and Reference,* Order No. SC34-2020.

The Interactive Problem Control System (IPCS) is a group of CMS commands which track and report both CP and non-CP problems. The IPCS commands are:

DUMPSCAN—which allows you to inspect CP dumps that the VMFDUMP command has converted to CMS files. It prompts you for the dump number and filemode, and it lets you enter subcommands to display specific parts of the dump and to locate data and addresses.

PRB—which allows you to update the status, last update function, severity, and PTF (Program Temporary Fix) files of the symptom summary record for a problem.

PROB—which allows you to describe a problem that is not a CP abend, or to add information to an existing problem report (whether or not it is a CP abend). It prompts you for all the necessary information about the problem.

STAT—which allows you to produce a list of the status of all problems that you can print or type. You can also request the status of a single problem or a subset of problems and display it at the terminal.

VMFDUMP—which allows you to convert CP dumps into CMS files, create problem reports, and search for duplicate problems.

All that is necessary to use the IPCS commands is that the command modules be installed on your VM system and that the modules and IPCS files be available to the appropriate users.

IPCS Report Files

Usually, all IPCS files reside on the A-disk of the user responsible for maintaining your system. All files associated with a given problem (such as a dump or supplementary files) are of the form:

PRBnnnnn filetype

The number assigned to the problem by IPCS is indicated by nnnnn and the filetype is one of the following:

DUMP—a CMS file; the output of the VMFDUMP command.

REPORT—the report generated by the PROB command or the VMFDUMP command. (One exists for each problem known to the system.)

Other IPCS Files

Other IPCS files include the NUC MAP file, the STATALL LOCAL file, the summary record, and the symptom summary.

NUC MAP is the nucleus load map of the CP dump being analyzed. It contains every module name and entry point in the CP nucleus and is required by the VMFDUMP command for successful analysis of the dump. An abbreviated version of the NUC MAP is appended to the VMFDUMP and is used by the DUMPSCAN command.

The STATALL LOCAL file contains the status of all problems known to the system and is created by the STAT command when entered with the ALL operand.

The summary record contains the next available problem number. It is a single 80-character record that is assigned to a problem when it is reported. The number is then increased by 1 and the summary record is rewritten.

The symptom summary contains the symptoms and status of each problem known to the system. There is one symptom summary control record for each problem that is created and placed in this file by the PROB and VMFDUMP commands. These records are displayed by the STAT command and updated by the PRB command. They are also used to identify possible duplicate problems as they are added to this file.

CP Abend Dumps

During system generation a user is designated to receive CP abend dumps. If an abend occurs when SET DUMP AUTO is in effect, and sufficient contiguous space is available in the CP paging area, the abend will appear in the designated user's virtual reader. The user can then use the VMFDUMP command to read the spool file, create a CMS file containing the dump, and print it.

After a CP dump is created, any user who has access to the IPCS commands and files can use them to examine the dump, the problem report, and the status of the problem. However, an IPCS file must be on the user's A-disk for him to update it.

Method of Operation

This section describes Interactive Problem Control System (IPCS). Diagrams describe the five IPCS functions. Figure 3-1 shows the relationship of these diagrams.

Diagram 3-1 shows how the DUMPSCAN command and its subcommands enable the user to interactively examine a CMS dump file created by CP.

Diagram 3-2 shows how the PRB command updates the status of problems in the symptom summary file.

Diagram 3-3 shows how the PROB command creates problem reports and adds information to existing problem reports.

Diagram 3-4 shows how the STAT command lists the current status of a given problem.

Diagram 3-5 shows an overview of how the VMFDUMP command creates a problem report by extracting pertinent data from a CP abend dump.

Diagram 3-6 shows how the nucleus load map is compressed.

Diagram 3-7 shows how a program check is handled.

Diagram 3-8 shows how a coded abend is handled.

Diagram 3-9 shows how an operator initiated dump is handled.

Diagram 3-10 shows how the preliminary information is printed.

Diagram 3-11 shows how the control blocks are formatted and printed.

Diagram 3-12 shows how the storage protection keys and dump file are printed.

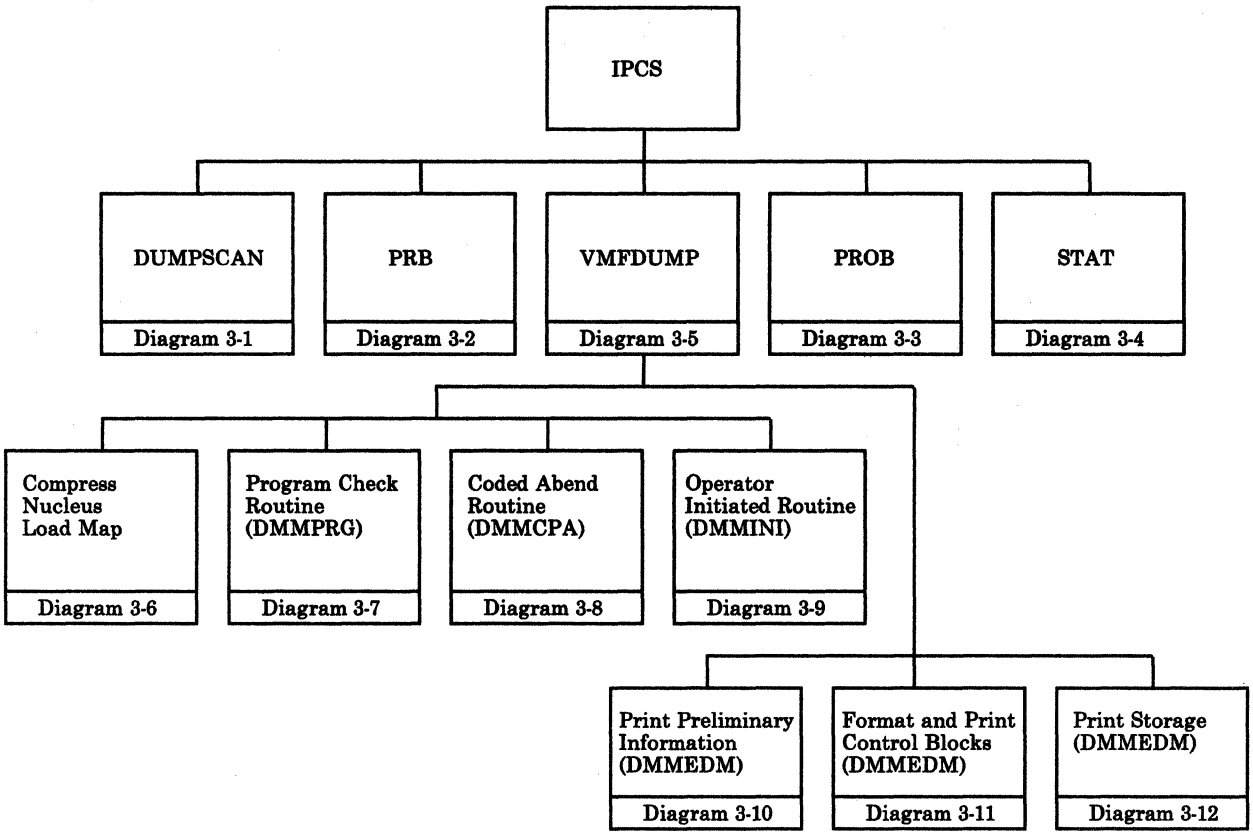
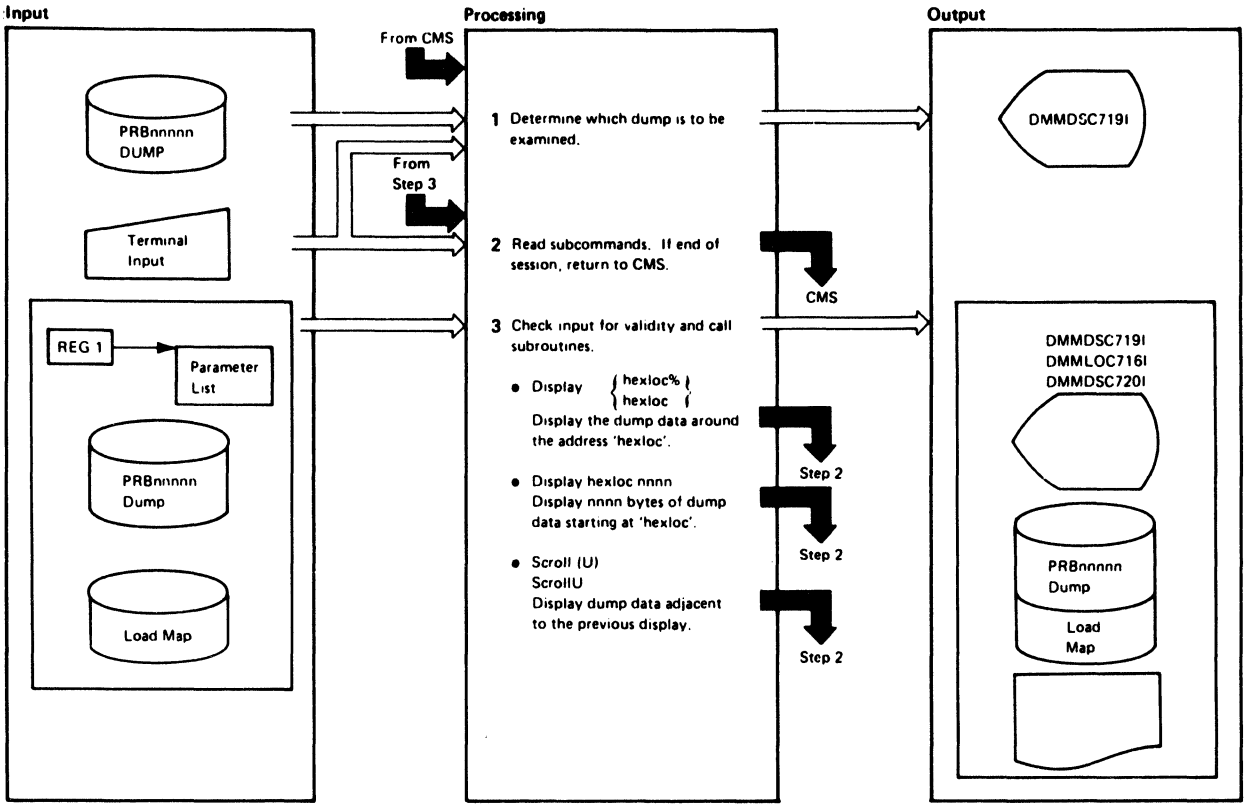


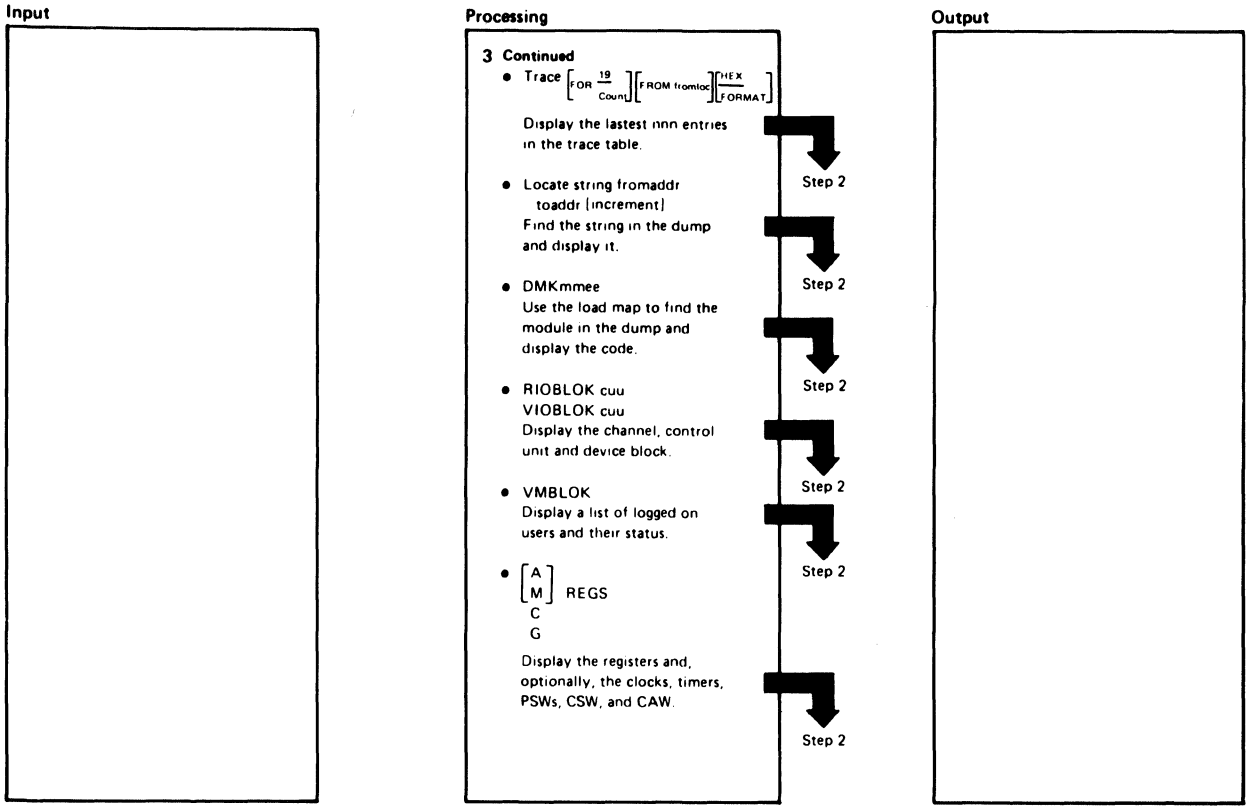
Figure 3-1. Key to Interactive Problem Control System Method of Operation Diagram



Notes	Module	Label	Ref
1 Ask if the user needs help. If yes, show 'HELP' pages. Prompt for the dump file number and mode. Check that the dump exists. If not, issue message: ERROR IN FSSTATE	DMMDSC	GETDUMP ENTER STATERR	
2 Read subcommands from the terminal. Truncate the input to 8-byte words and place in parameter list. If end of session ('HX', 'QUIT', or 'END'), return to CMS.	DMMDSC	READY TOKEN INCHECK	
3 Scan the parameter list and check subcommand for validity. Pass control to the appropriate subroutine.	DMMDSC	PLISTSCN	
• Convert 'hexloc' to hexadecimal and call DMMGET to fetch the dump data into storage. Check for indirect request (%). Call DMMDIR to format and display the data.	DMMFEX	TRANADD NOROUND INDIR DIRECT	
• Translate the address of 'hexloc' and the byte count nnnn to hexadecimal. Fetch the data into storage, format the data, convert to EBCDIC, add addresses, and write the data to the terminal.	DMMFED	DMMFED TRANADD CTRAN DIRECT SCRNFULL	
• Increase (decrease if 'U') the last displayed address by X'130', checking for an end of page condition. Fetch that data into storage. Invoke the format and display routine (DMMDIR)	DMMSCR	DMMSCR DIRECTIT	

Notes	Module	Label	Ref

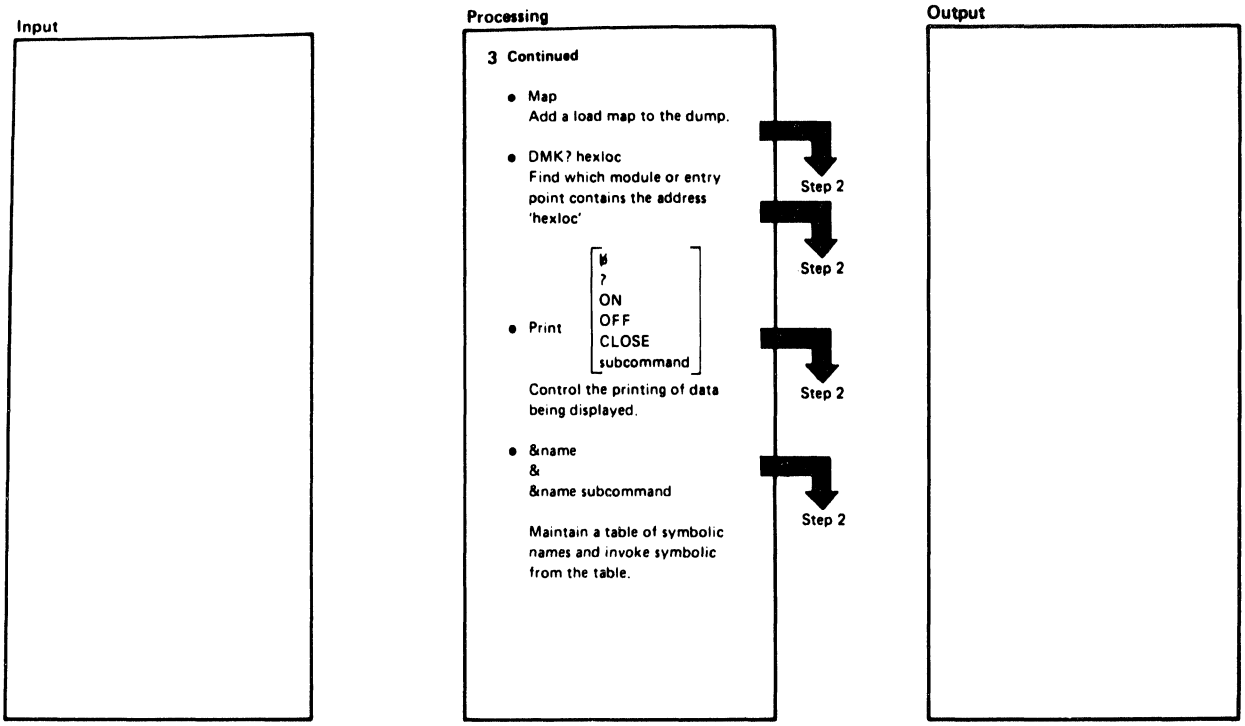
Diagram 3-1. DUMPSCAN IPCS Command (Part 1 of 3)



Notes	Module	Label	Ref
3 Continued			
• Find the trace table from 'traccurr' in the PSA. Convert the count to a byte count. Submit it to DMMFED as display 'hexloc' nnnn.	DMMTRC	DMMTRC	
• Fetch the page containing the 'fromaddr' into storage. Compare the string against the data at 'fromaddr'. If not equal, increase the data pointer by the 'increment' parameter and compare again. Continue until the comparison is equal then display the area containing the equal compare or until the 'toaddr' is reached then issue message: STRING 'string' NOT FOUND	DMMLOC	DMMLOC EXECUTOR GOGOFEX MOREMSG	
• Read the load map from the end of the dump and scan it for this label. Submit the address of the label to DMMFEX to display.	DMMMOD	DMMMOD MAPRED	
• Separate the channel block, control unit block, and device block for the given real or virtual device address. Display the blocks.	DMMIOB DMMFED	DMMIOB COMPRCUB UIO DMMFED	
• Get the system VMBLOK pointer from the PSA. Follow the pointer to the chain of VMBLOKs Print a list of the active VMBLOKs with the userid and selected status bytes.	DMMVMB	DMMVMB BALGET MOVEL	
• Check for AP or UP dump. Select the appropriate set of registers and if the subcommand is not 'C' or 'G', also display the PSW and clocks.	DMMREG	DMMREG	

Notes	Module	Label	Ref

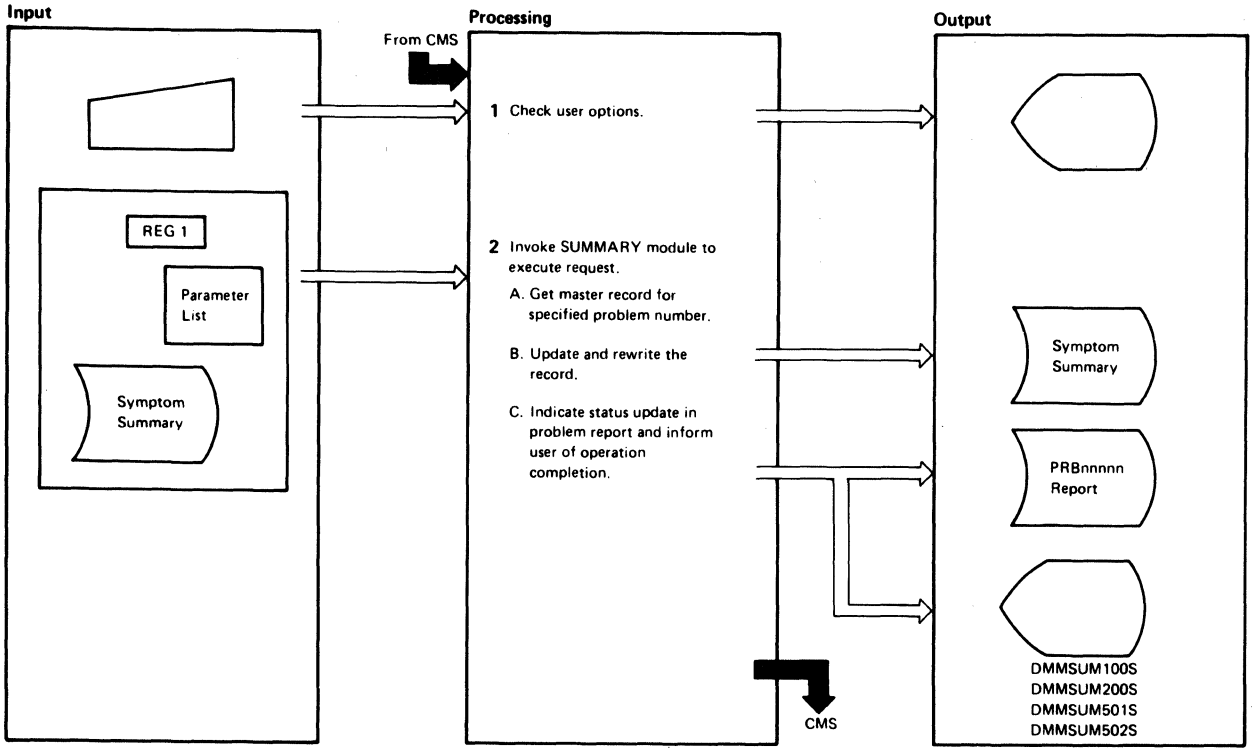
Diagram 3-1. DUMPSCAN IPCS Command (Part 2 of 3)



Notes	Module	Label	Ref
3 Continued			
• Check that the dump does not already have a load map. If it does, issue the message: LOAD MAP ALREADY PRESENT If it does not, call DMMMAP to add the load map to the dump. (See Diagram 3-6 for a description of DMMMAP processing.)	DMMDSC	MAPCHECK TWOMAPS	
• Read the load map from the end of the dump. Scan for the address closest to, and before the given address. See if the module is pageable. If it is, find its loaded address at dump time. Display the entry point name and displacement.	DMMMOD	READ QREQUEST PAGEMOD	
• Turn PRINT 'ON' or 'OFF' as requested. ? Display the current print status. CLOSE Issue CP DIAGNOSE '08'. subcommand. Issue subcommand and turn printing 'ON' for subcommand. PRINT Reissue the previous subcommand and print the output.	DMMDSC	CHECKTWO SHOWPSW CLOSEPRT SUBCOM RESUBCOM	

Notes	Module	Label	Ref
• &name Call a names subcommand from the table. & Display a list of the entries in the table. &name subcommand Add the subcommand into a table of subcommands.	DMMDSC	FOUNDAMP SHOWTAB NOTINTAB	

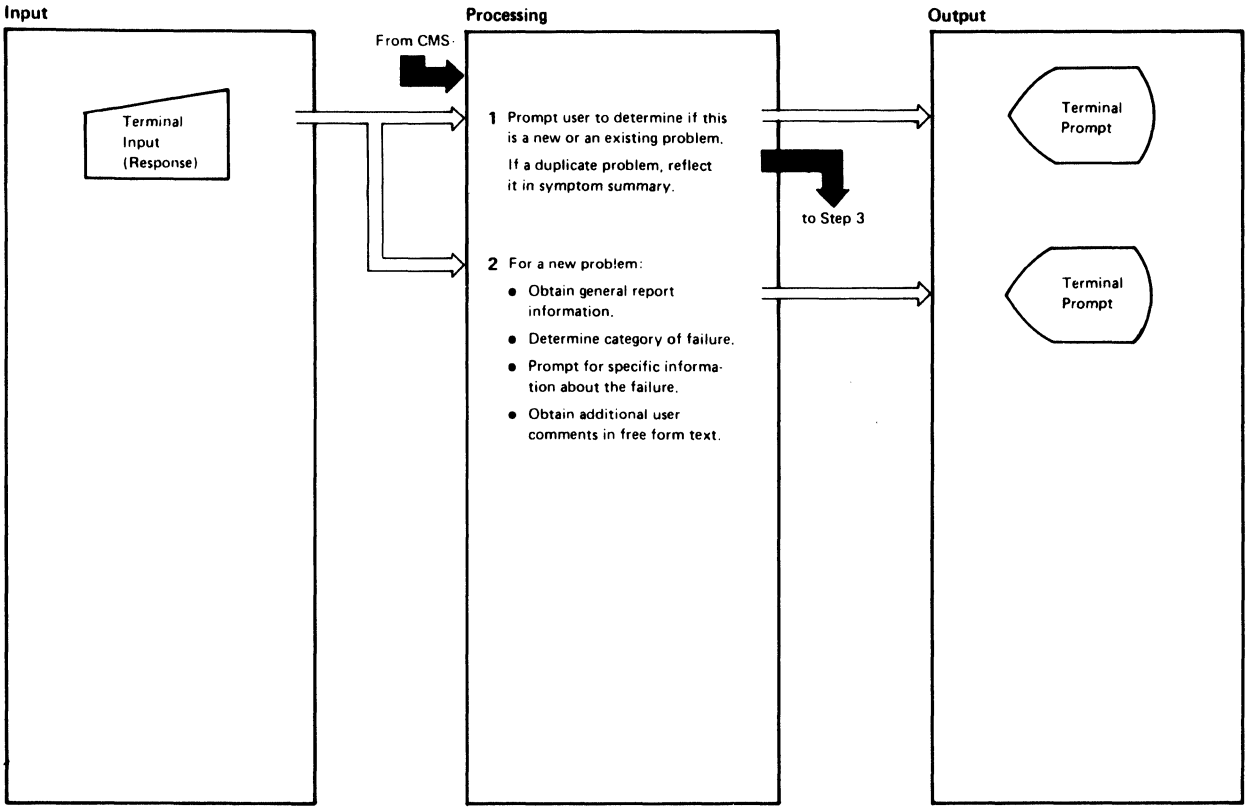
Diagram 3-1. DUMPSCAN IPCS Command (Part 3 of 3)



Notes	Module	Label	Ref
1 Check the first two operands (which may be in any order) as entered, swap them and check again. Issue error messages for invalid input.	PRB	–RETRY	
2 Call SUMMARY module (DMMSUM). The following routines supply information appropriate to the user's request: –PTFON –DSPLY –PTFIS –IBM –CLOSE –USER –DUPOF –NEEDINFO –APAR –HELP –SEV			
A. Pass input to parameter list pointed to by register 1. The parameter list is: **Byte** **Contents** 1-8 Not used 9-16 PRBnnnnn (problem number) 17-24 Request type: UPSTAT – Update status UPFUNCT – Update last function UPSEV – Update severity UPPTF – Update PTF information UPDUP – Update duplicate information UPAPAR – Update APAR information 25-32 Update specific information.	DMMSUM	START	
B. All activities cause "LAST" date to be updated with the current date.			

Notes	Module	Label	Ref
C. After the status is updated in the symptom summary file, append the date and time and new status to the problem report for history purposes. The SUMMARY module supplies a return code which is checked. If zero, an informational reply is issued indicating successful completion. If the completion code is not zero, an informational reply is issued indicating that the update was unsuccessful.	DMMSUM PRB	REPORT1	

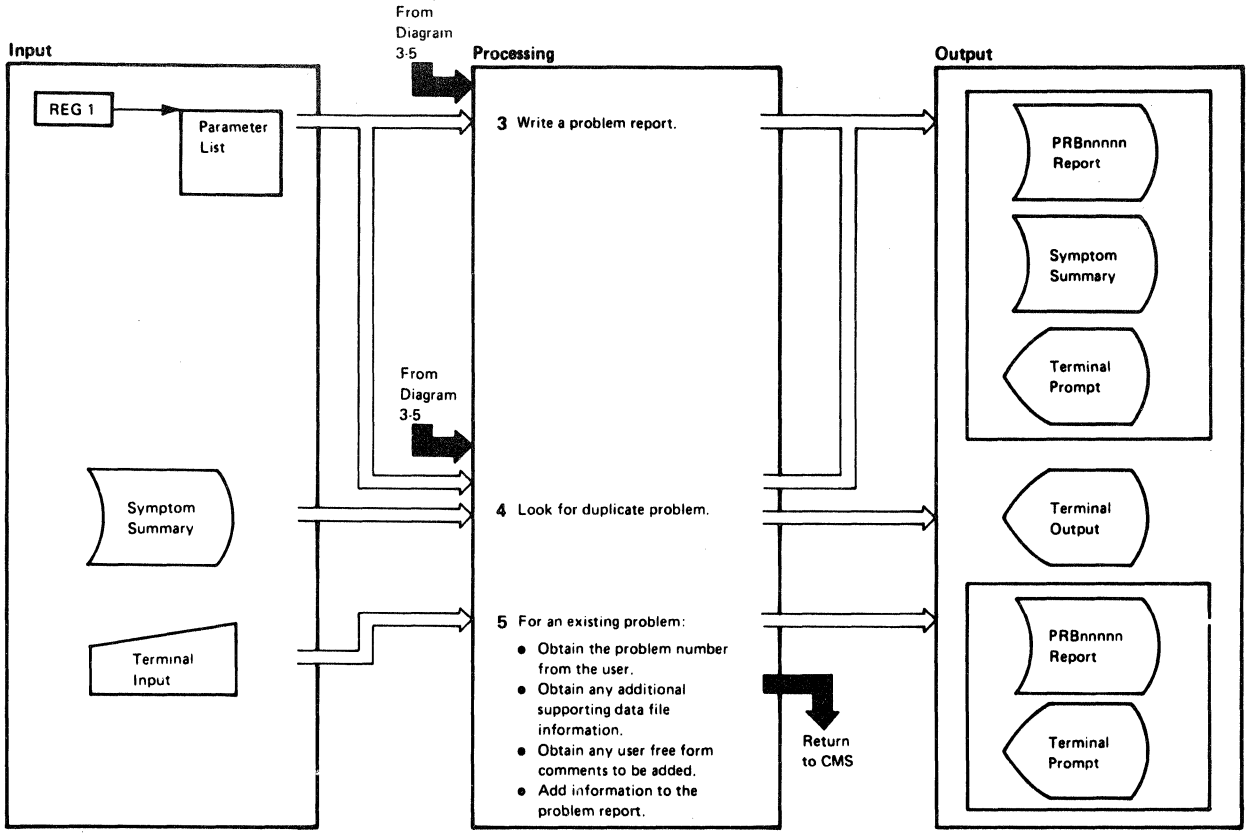
Diagram 3-2. PRB IPCS Command



Notes	Module	Label	Ref
1 Prompt user to find out if this activity is to create a new problem report or to update an existing report.	DMMPRO	EXIST	
2 A response of 'NO' to this prompt indicates this is a new problem. • The user is prompted for the date and time of the failure, the SCP, CPU type, CPU serial, and other general information.	DMMPRO	MAINLINE	
• The user is prompted for the category of the problem; for example, abend message or loop.		GETFAIL	
• The user is prompted for detailed information, depending on the type of failure			
• The user is prompted for additional user comments.		TEXTENTR	

Notes	Module	Label	Ref

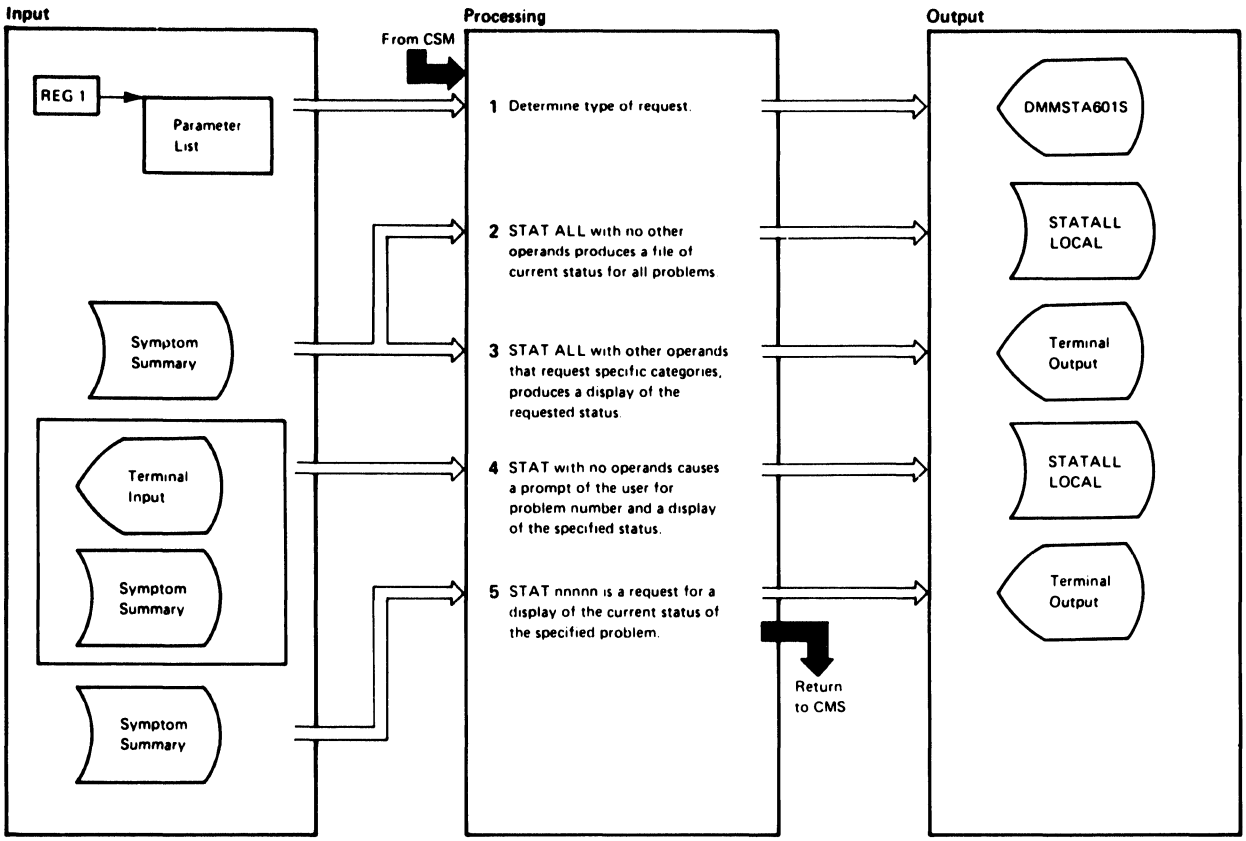
Diagram 3-3. PROB IPCS Command (Part 1 of 2)



Notes	Module	Label	Ref
3 The date and time stamps are supplied from CMS low storage. The FSWRITE routine writes the first two records, which contain general information about the problem. The third record, containing a start of keyword area indicator, is written.	DMMWRT	INTOUT	
• Keyword data is passed in variable blocked format. The data is extracted and moved to the output buffer one entry per 80 character record. FSWRITE adds this data and an end of keywords record.		KEYOUT	
• Supplementary data file names are added to the problem report file if supplied.		SUPPOUT	
• Textual descriptions of the problem are added to the file if supplied.		TEXTOUT	
• Data from INTSECT (the internal data area) low storage (time and date) and the initial status fields are moved to the 80-byte output area and FSWRITE adds the data to the symptom summary file.		CNTRLOUT	
• The keyword data is rounded up to a multiple of 80 bytes and the information is added to the symptom summary file.		CNTRLOUT	

Notes	Module	Label	Ref
4 Look for duplicate problem:			
• The keyword data for the new problem is compared to that of all existing problems and any exact matches are considered duplicates. The search is terminated when the newly created problem is encountered.	DMMSEA	START	
• The user is notified (at the terminal) as each duplicate problem is encountered. Up to 10 duplicate problems may be displayed for a search.		PUTOUT	
• If duplicate was found, DMMSUM is called to record the first encountered duplicate problem number in both the symptom summary control record for the problem, and the problem report.		ENDRTN	
5 A response of 'YES' to the prompt indicates that this is an update to an existing problem:	DMMPRO	OLDPROB	
• The user is prompted for the number of the problem, and its existence is verified.			
• The user is prompted for any additional data file names.		GETSDATA	
• The user is prompted for free form comments to be added to the report.		TEXTENTR	
• The new information, with a date and time stamp is added to the problem report.		OLDADD	

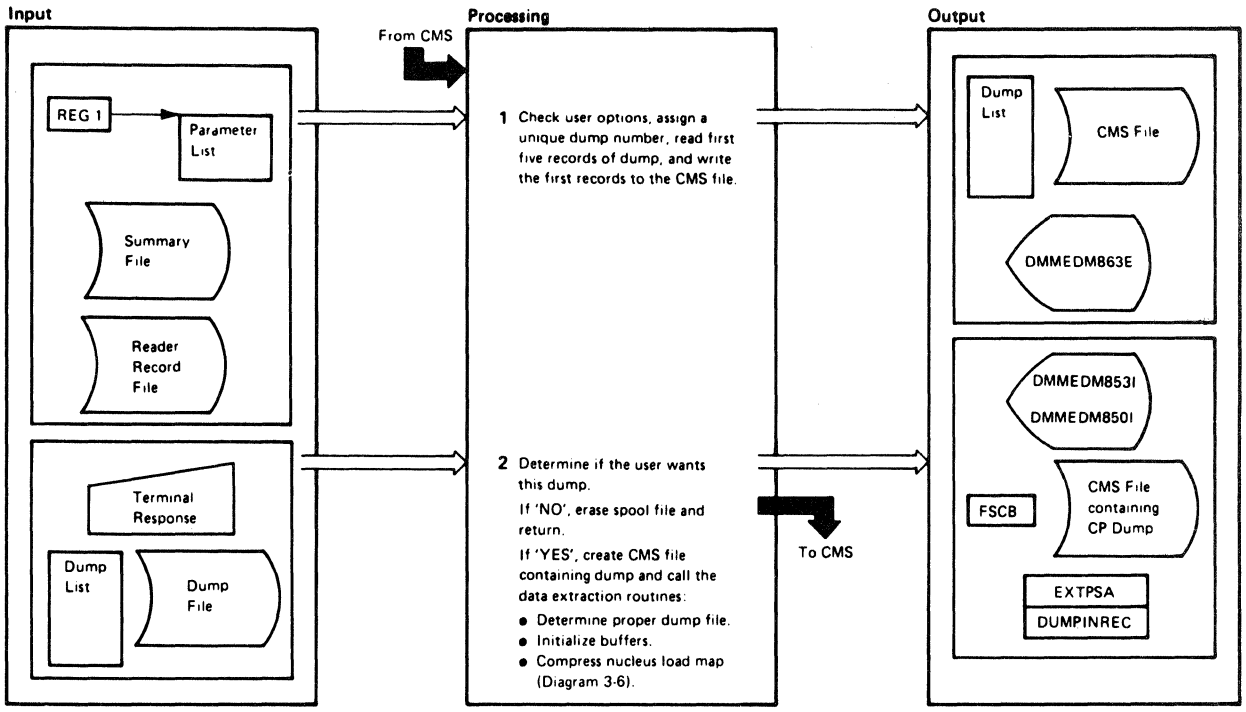
Diagram 3-3. PROB IPCS Command (Part 2 of 2)



Notes	Module	Label	Ref
1 If an operand is not recognized, issue message: OPERAND NOT RECOGNIZED, STATALL ASSUMED	DMMSTA	START	
		CK2CONT	
2 STAT ALL: Set switch (LALLSW) and erase any old copy of STATALL LOCAL file. Heading line is written followed by all the symptom summary control records and the file is closed.		STALLOC	
3 STAT ALL oper: If any additional operands are not recognized, issue message: OPERAND oper NOT RECOGNIZED • If operands are valid, the entire symptom summary file is searched and each control record is matched with the specified operands. • If a match is found, the control record is presented to the user on the terminal and the search continues.		STATSRCH	

Notes	Module	Label	Ref
4 STAT: The user is prompted for the number of the problem whose status he wishes. • If he enters 0000, STAT ALL is assumed (see Step 2). • If he enters a number other than 0000, that number is checked for validity and the symptom summary file is searched for the requested problem.		SPNUM	
5 STAT nnnnn: The problem number nnnnn is checked for validity. • If the number is in the correct format, the symptom summary file is searched for the requested problem. • The status is displayed when found. • If the problem is not found, issue message: PROBLEM NOT FOUND IN SYMPTOM SUMMARY FILE		STATRDY	

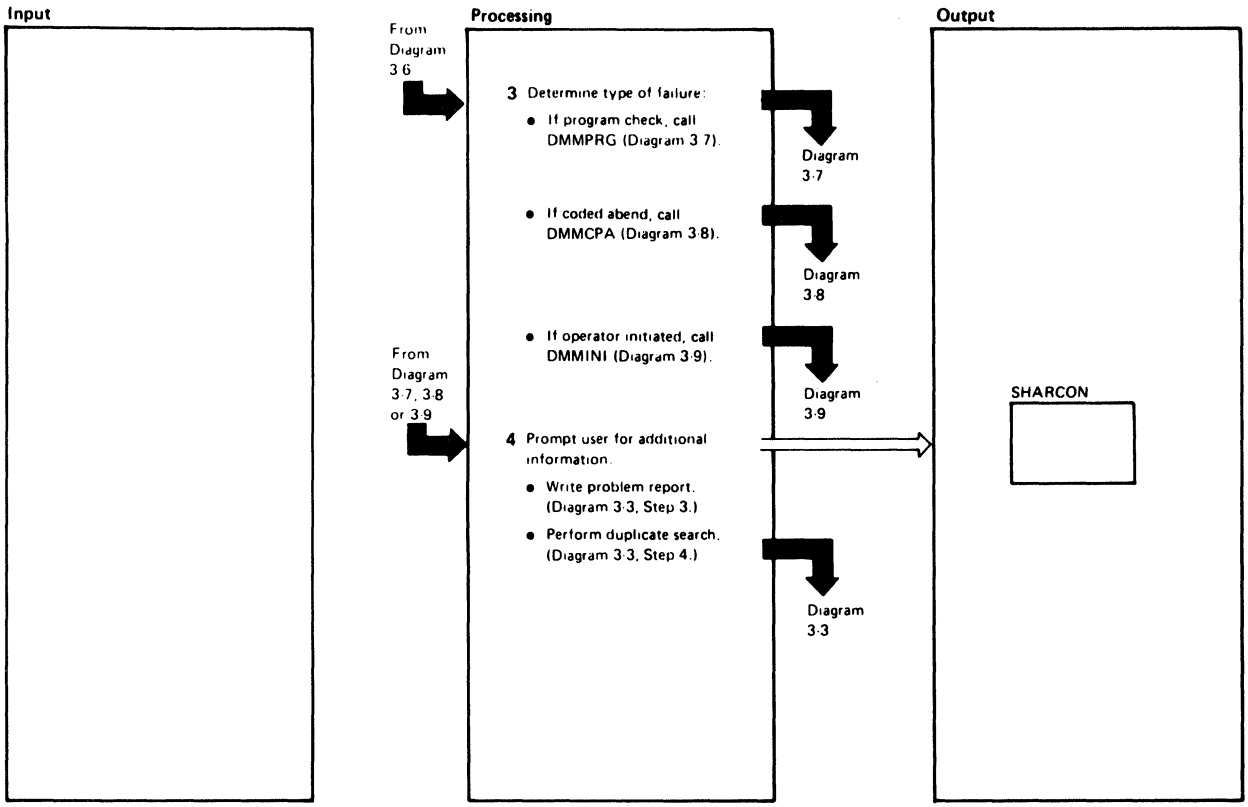
Diagram 3-4. STAT IPCS Command



Notes	Module	Label	Ref
1 If there are no options specified, the defaults are MAP, FORMAT and HEX. If an invalid option is specified, issue message: INVALID PARAMETER parm PAGE REFERENCED NOT AVAILABLE	DMMEDM DMMINI	CHKOPT ERRFND PRBDUMNO	
Read the next sequential number from the summary record file, and append it to the dump prefix 'PRB'. The dump name will always be PRBnnnnn, where 'nnnnn' is the unique dump number.	DMMEDM	RDUMP	
Read the records by branching and linking to the READCPR routine. The spool records contain the following: **Record** **Contents** 1 Symbol Table 2 Dump Information record 3-4 Storage Protection Keys at time of dump 5 First page of storage dumped (0)			
Save the following from the information record (2): • First 256 bytes of storage • General and floating point and control registers • TOD clock and comparator • Address of the terminating prefix storage area • Abend code		NXTWD	
Create the item table from the bit map in record 2.		RDUMP	
Write the records to the CMS file that will contain the CP dump file (if requested) by branching and linking to the WTREC routine.			

Notes	Module	Label	Ref
2 To determine if the user wants this dump, issue the message: VM/370 SYSTEM ABEND xxxx DATE (date) TIME (time) DO YOU WANT THIS DUMP?			
If the user responds 'NO', erase the spool file and return to CMS.		REREAD	
If the user responds 'YES', write a record by branching and linking to the WRT Routine, read another record by branching and linking to the READCD routine, and so on until the read returns a non-zero condition code: **Code** **Meaning** 1 End-of-file, 'DUMP PRBnnnnn CREATED' message is issued 2 Issue message 'NO DUMP FILES EXIST' and return to CMS 3 Issue message 'UNABLE TO READ DUMP FROM READER' and return to CMS		LOOP DMPEND1 NODMP LOOP	
• Examine the dump list that resides in DMMEDM as a constant to determine the file name assigned to the dump; move this name to the read 'FSCB' to facilitate subsequent reads to the dump file.	DMMINI		
• Initialize buffers EXTPSA (terminating PSA in the dump) and EXTINREC (record 2 from the dump) by issuing FSREADs to the dump file.		EXTREAD	
• Call the nucleus load map module (DMMMAP) to compress the nucleus load map. (See Diagram 3-6.)		EXTREND	

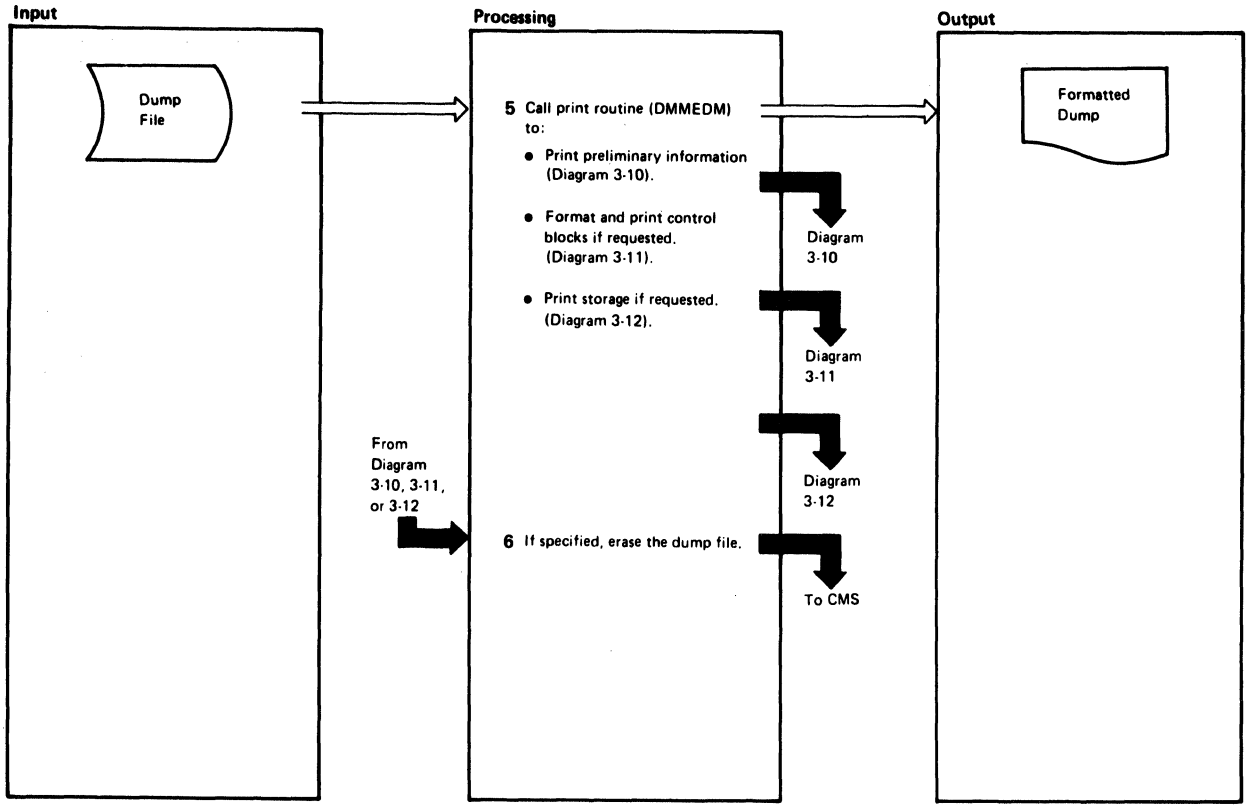
Diagram 3-5. VMFDUMP IPCS Command (Part 1 of 3)



Notes	Module	Label	Ref
3 Examine the DMPABEND field of the dump information record (EXTINREC) to determine the failure type: • If the failure type is a program check (PRGxx), call the program check routine (DMMPRG). See Diagram 3-7.	DMMINI	EXTPSWCK	
• If the failure type is other than PSA02, call the coded abend routine (DMMCPA). See Diagram 3-8.		EXTSVCHK	
• If the failure type is PSA02, handle within DMMINI. See Diagram 3-9.		EXTPSCHK	
4 Prompt the user for the severity code, examine the previously set switches in the SHARECON data area to determine the failure. If it was system detected (CP abend or program check), request the file name and file type of any supporting documentation, and a free form entry description of the problem.	DMMPRM	GETSEV PRMTYPSW PRMSUPP	
Note. The prompting sequence for operator initiated dumps depends on the user's response to the query: THE DUMP INFORMATION IS INCONCLUSIVE ENTER LOOP, PERFORMANCE OR OTHER		PRMLPPER	

Notes	Module	Label	Ref
• All information necessary to create the problem report has been gathered. Call module DMMWRT to order the data and create the problem report. See Diagram 3-3, Step 4. • Call module DMMSEA to search for duplicate problems. See Diagram 3-3, Step 4.		NORMEXIT	

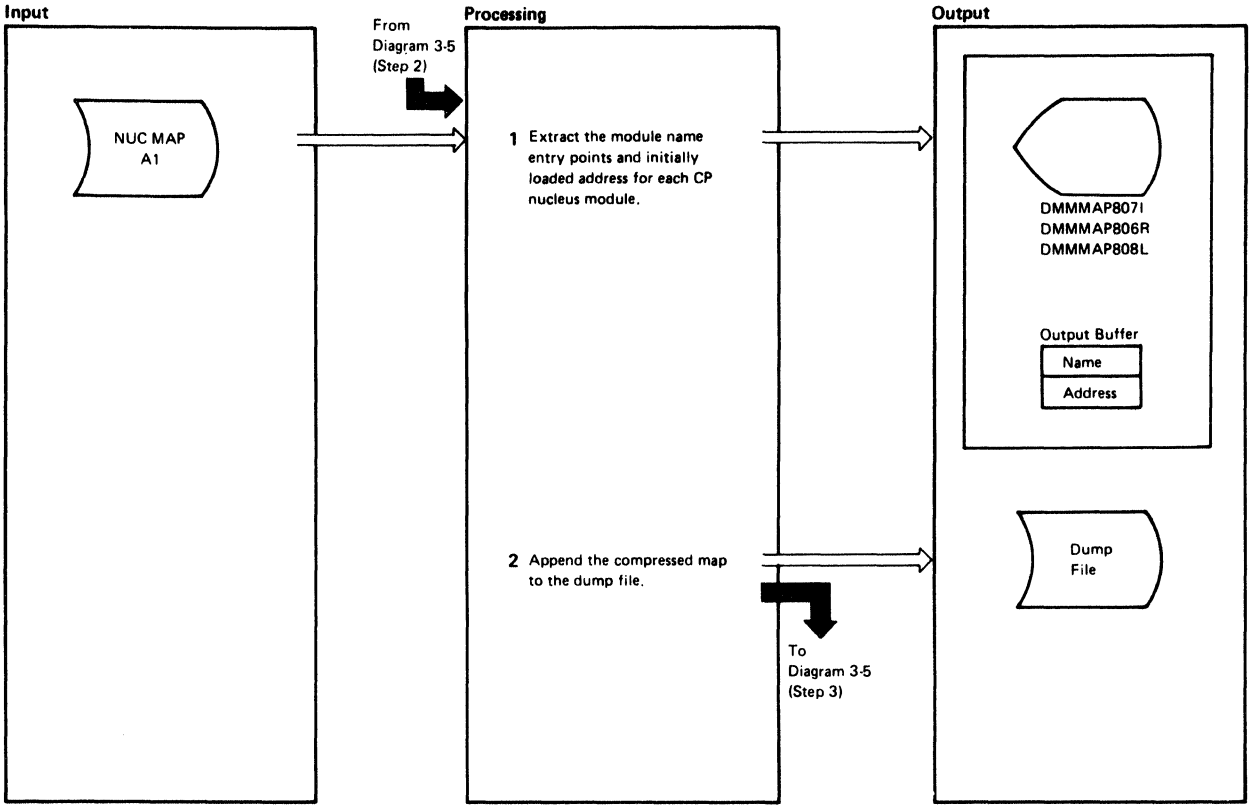
Diagram 3-5. VMFDUMP IPCS Command (Part 2 of 3)



Notes	Module	Label	Ref
5 Pass control to the print routine (DMMEDM) to print the dump:	DMMPRM	EXIT	
• Read record1 (symbol table) and record2 (dump information) from the dump file and print the preliminary information. See Diagram 3-10.	DMMEDM	EDITDUMP	
• If the NOFORM option was omitted, format and print the control blocks. See Diagram 3-11.		RCHFORM	
• If the NOHEX option was omitted, print storage. See Diagram 3-12.		HEXDUMP	
6 If the ERASE option was omitted, keep the dump file. If specified, erase the dump file. In either case, return control to CMS.		RETN	

Notes	Module	Label	Ref

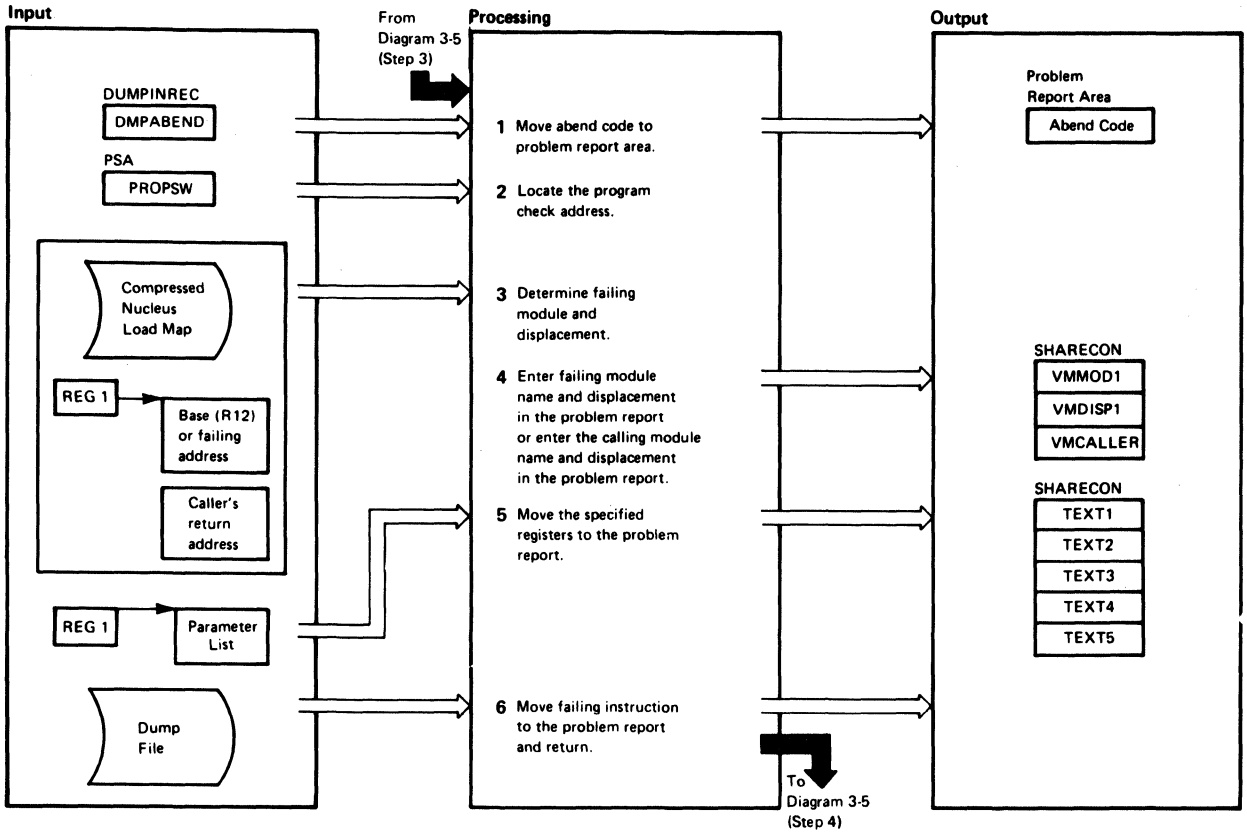
Diagram 3-5. VMFDUMP IPCS Command (Part 3 of 3)



Notes	Module	Label	Ref
1 Attempt a read to NUC MAP A1.	DMMAP	READ	
Note: The nucleus load map is assumed to reside on the IPCS user's A-disk. If NUC MAP A1 cannot be found, issue message:		READERR	
UNABLE TO LOCATE NUC MAP A1		STATERR	
Follow this message with message:		MAPNAME	
ENTER fn ft fn OF THE NUCLEUS LOAD MAP			
If the load map is successfully located, compare the address of the constant DMKCPEND in the symbol table (dump record1) to the address of DMKCPEND in the load map. If the addresses do not compare, issue message:		XCK	
		MAPERROR	
NUCLEUS MAP INVALID 'file id'			
If the map is valid, read each line of the map into a buffer. If it contains a module or entry point name, move this name and associated address (12 bytes) to an output buffer.		READ	
2 When end-of-file is reached sort the output buffer by ascending entry point address, write the output buffer and append it to the dump file.		WRTOUT	

Notes	Module	Label	Ref

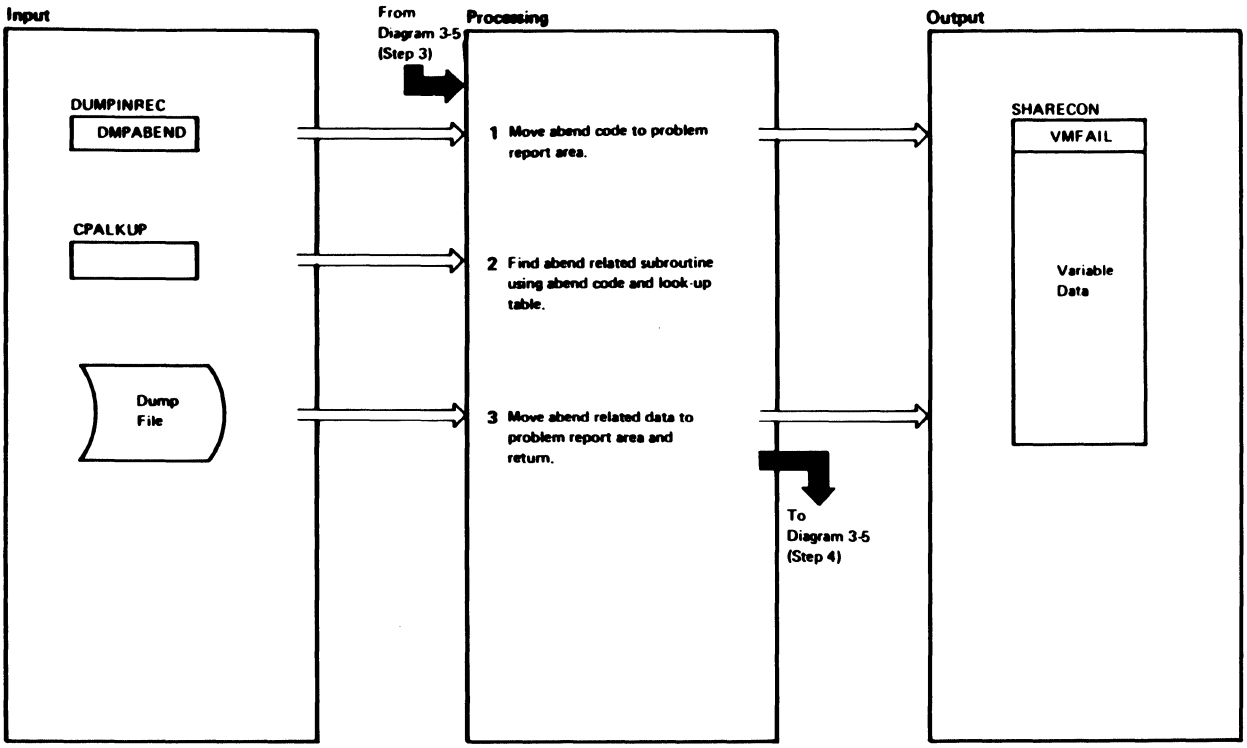
Diagram 3-6. Compress the Nucleus Load Map



Notes	Module	Label	Ref
1 Move the abend code from DMPABEND into the problem report area.	DMMPRG		
2 Identify the program check address in the PSA Program Old PSW.			
3 Call DMMIDM to identify the failing module and displacement.			
If entered from DMMPRG, the failing address will be in the fixed nucleus portion of the dump or in a pageable module.	DMMIDM	MODREAL	
		MODPRGCK	
If entered from DMMCPA, the caller's base (R12) will be in the fixed nucleus or in a pageable module.		MODABND	
		MODPAGE	
Using the addresses provided, and the compressed nucleus map, calculate the displacement of the failing or calling module.			
4 Enter the name of the failing module in the problem report or enter the name of the calling module in the problem report.	DMMIDM	MODPRGCK	
		MODGOOD	

Notes	Module	Label	Ref
5 Call DMMRMV to move the register set indicated by the pointer passed in register 1. It can be one of the following: • general registers • BALSAVE registers • FREESAVE registers • SAVEAREA registers • LOKSAVE registers • SWTSAVE registers	DMMRMV		
6 Move the failing instruction to the problem report and return to the user prompting routine.	DMMPRG	PTGINSTR	
		PRGMORCD	

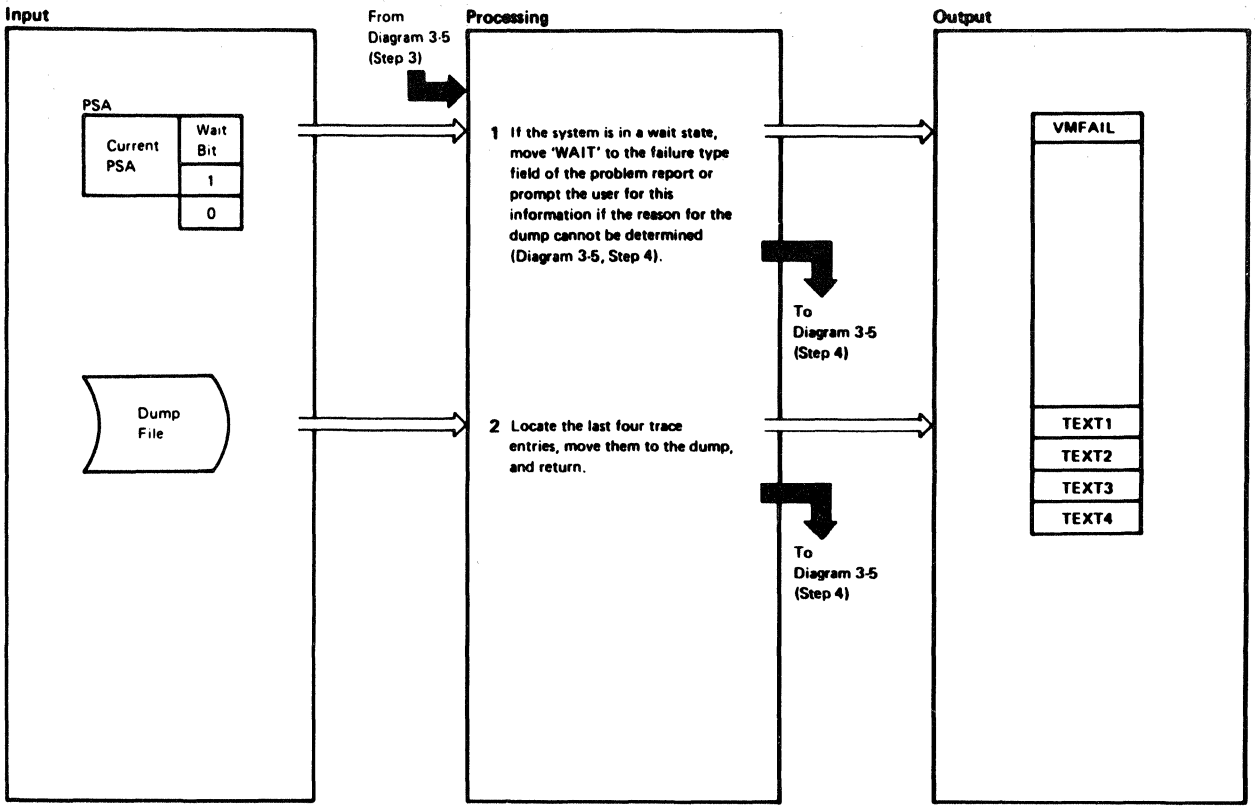
Diagram 3-7. Program Check Routine (DMMPRG)



Notes	Module	Label	Ref
1 Move the abend code from the DMPABEND field of the dump information record (record 2) into the problem report area.	DMMCPA		
2 Compare the prefix of the abend code to the entries in the abend look up table; in this table an entry exists for every known abend and it includes a prefix code followed by a sub-routine address.		CPACNTIN	
3 When the appropriate subroutine is located, move the predetermined data to the problem report. *Notes:* The amount and type of data extracted will vary according to the abend, but will generally fall into one of the following categories: • TEXT 1 through TEXT 5 contain the IOBLOK. • TEXT 1 through TEXT 5 contain an information type entry; for example, THE CLOCK IS IN ERROR OR NOT OPERATIONAL • TEXT 1 through TEXT 5 contain one of the six register save areas (reserved); VMDISP is the displacement within the caller; and VMCALLER is the module calling the failing module. The last category requires calling DMMIDM and DMMRMV. (See Diagram 3-7, Steps 4 and 5.)		CPAGETAD	
Call the user prompting routine.	DMMINI	CPACALL	

Notes	Module	Label	Ref

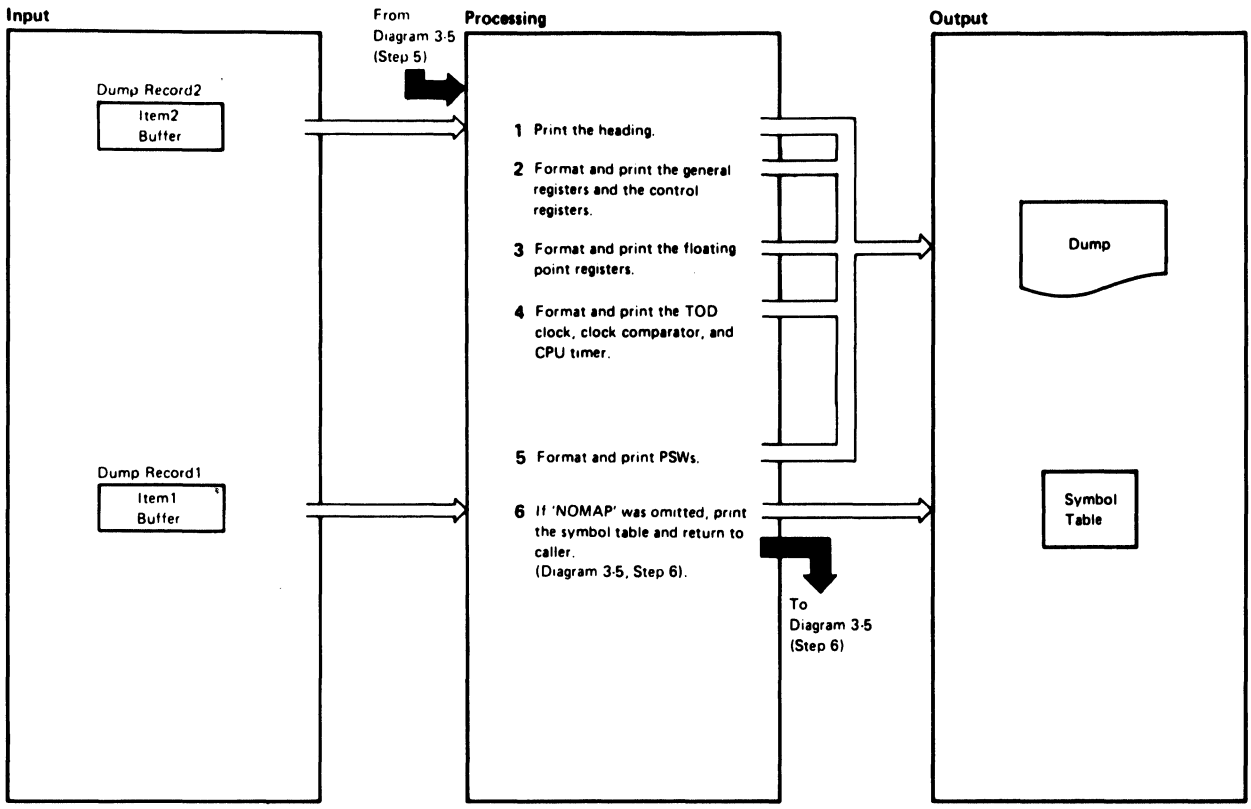
Diagram 3-8. Coded Abend Routine (DMMCPA)



Notes	Module	Label	Ref
1 Examine the wait bit in the current PSW. If on, the system is assumed to be waiting when the operator depressed the SYSTEM RESTART key. Move WAIT to the problem report failure area and re-examine the current PSW to check for the presence of a wait code. If one exists, move it to the problem report also. If off, consider the dump information inconclusive and prompt the IPCS user for a failure code. (See Diagram 3-5, Step 4.)	DMMINI	EXTLPWT	
2 Locate the last four trace entries for all operator initiated dumps, move them to the problem report area, and return to the user prompting routine (Diagram 3-5, Step 4).		EXTTRTAB EXTLEAV	

Notes	Module	Label	Ref

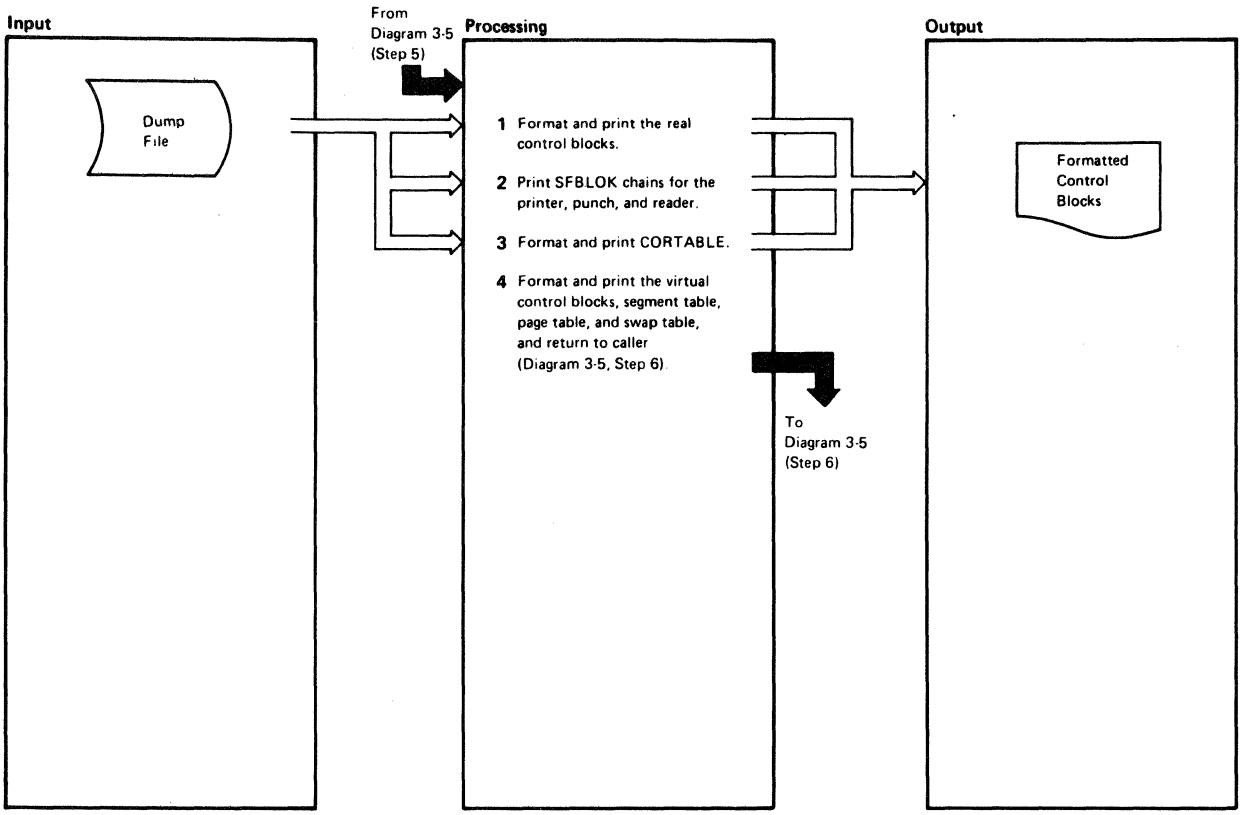
Diagram 3-9. Operator Initiated Routine (DMMINI)



Notes	Module	Label	Ref
1 Print the heading line which contains the time, date, abend code and cause.	DMMEDM	PREREC	
2 Unpack the general registers and the control registers by branching and linking to the transmit routine, move the data by branching and linking to the MVSBRTN routine and print it by branching and linking to the PRINTA routine.		PRELIM4	
3 Print the floating point registers as in Step 2, above.		PRELIMB	
4 Unpack and print the TOD clock, clock comparator and CPU timer as in Step 2, above.			
5 Translate low storage and format and print the PSWs.		PRELIM11	
6 If NOMAP was omitted, print the symbol table.			

Notes	Module	Label	Ref

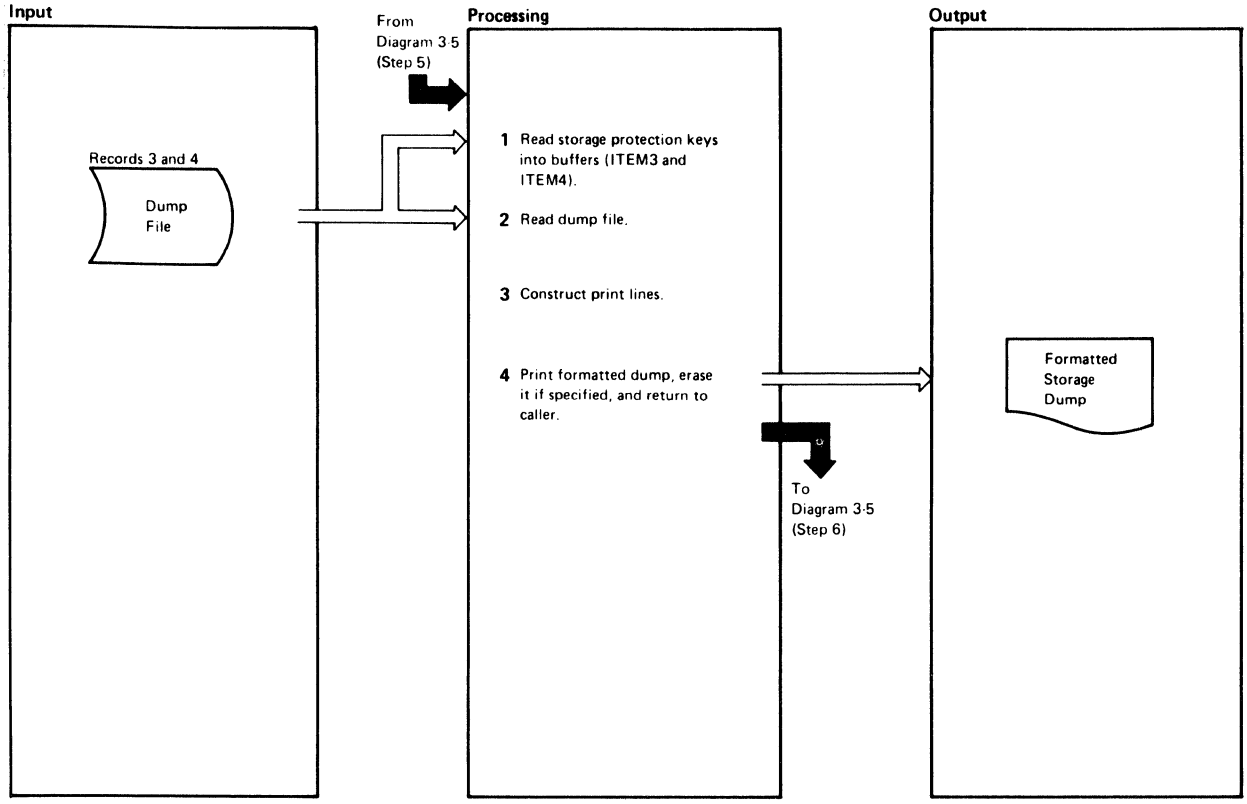
Diagram 3-10. Print Preliminary Information (DMMEDM)



Notes	Module	Label	Ref
1 Format and print the following real control blocks: • RCHBLOKs and IOBLOKs chained to them • RCUBLOKs and IOBLOKs chained to them. • RDEVBLOKs • Active IOBLOK • RSPLCTL and SFBLOK for unit record devices • CONTASK for termination • RECBLOKs for CP owned DASD devices Branch and link to the following routines for commonly used functions: GETPAGE To get a page of storage. TRANINIT To translate control blocks into printable form. BLKPRINT To print real control blocks. IOBPRINT To print IOBLOK. SFPRINT To print SFBLOK. IOERPRINT To print IOERBLOK.	DMMEDM	RCHFORM RCHPROC RCUINIT RDEVINIT IOPROC	
2 Print the printer, reader, and punch SFBLOK chains. *Note:* PRTSPL points to the punch spool and RDRSPL points to the reader spool.		SPFORM	
3 Unpack and print the CORTABLE.		CORTBL	

Notes	Module	Label	Ref
4 Format and print the following virtual control blocks: VMBLOKs ECBLOKs (if any) VCHBLOKs VCUBLOKs VDEVBLOKs Active IOBLOK (if any) VCONCTL (for console) VSPL CTL and SFBLOK (for unit record devices) Segment, Page and Swap Tables *Note:* These subroutines branch and link to subroutines to perform commonly used functions. **Subroutine** **Function** GETPAGE Get the page of storage containing the control block. TRANINIT Unpack control block for printing. BLKPRINT Print control block. SFPRINT Print SFBLOCK. IOERPROC Print IOERBLOCK. SEGPGTB Print segment, page, and swap tables.		VIRTUALM VMPRINT VCHINIT VCUINIT VDVINIT TSTSPOOL VMCK	

Diagram 3-11. Format and Print Control Blocks (DMMEDM)



Notes	Module	Label	Ref
1 DMMEDM reads record 3 and 4 from the dump file into the buffers called item3 and item4. These records contain the storage protection keys.	DMMEDM	HEXDUMP	
2 Read the remainder of the dump file, a page at a time, and place them in the print buffer.		READPAGE	
3 Construct the print line, placing the storage keys with the associated hexadecimal storage contents. (Printing of identical lines is suppressed.)		GETKEY	
4 Print the dump by branching and linking to the PNTPAGE routine. If ERASE is specified, erase it and issue message: DUMP PRBnnnnn PRINTED AND ERASED		GETKEY1	

Notes	Module	Label	Ref

Diagram 3-12. Print Storage (DMMEDM)

Program Organization

This section describes the program organization of Interactive Problem Control System (IPCS). The logic of modules DMMCPA, DMMDIR, DMMDSC, DMMEDM, DMMFED, DMMFEX, DMMGET, DMMGRC, DMMHEX, DMMIDM, DMMINI, DMMINT, DMMOIB, DMMLOC, DMMMAP, DMMMOD, DMMPRG, DMMPRM, DMMPRO, DMMREG, DMMRMV, DMMSCR, DMMSEA, DMMSTA, DMMSUM, DMMTRC, DMMTRN, DMMVMB, and DMMWRT.

DMMCPA — Extracts Information Pertinent to Individual Abend Conditions and Enters it in a Problem Report

Entry Point

DMMCPA

Entry Conditions

At entry, the shared constant area contains information previously gathered from the dump, and the PSA has been read into EXTPSA buffer.

Exit Conditions

The abend code and data related to that abend are in the problem report.

Routines Called

DMMIDM — Which finds the calling module and displacement.
DMMTRN — Which translates the data from hexadecimal to EBCDIC.
DMMGRC — Which reads in the requested dump file records.
DMMPRM — Which is the user prompting routine.
DMMRMV — Which moves the registers to the problem report.

Called By

DMMINI

Error Messages

DMMCPA805I

DMMDIR — Formats and Displays Hexadecimal Data on the Terminal Screen

Entry Points

DMMDIRLN — Which displays the HELP pages for DMMDSC.
DMMDIR — Which formats a screen from dump data.

Entry Conditions

R2: Points to the area to be displayed.
R7: The dump address to be displayed.

Exit Conditions

R15: Return code

0 Good
4 Print error
8 Unrecoverable error

Routines Called

DMMINT — Which translates from hexadecimal to EBCDIC.

Called By

DMMFEX, DMMLOC, DMMMOD, and DMMSCR

Error Messages

None

DMMDSC — Provides a Method of Examining the CMS Format CP Dumps Created by VMFDUMP

Entry Point

DUMPSCAN

Entry Conditions

From CMS when the DUMPSCAN command is issued.

Exit Conditions

R15: Return code

0 User 'HX', 'QUIT', or 'END'
8 Error processing the dump

Routines Called

DMMFEX — Which writes a full screen from the dump.
DMMFED — Which displays areas of the dump.
DMMLOC — Which locates data strings.
DMMSCR — Which performs the scroll function.
DMMREG — Which displays the registers.
DMMVMB — Which displays the VMBLOK summary.
DMMMOD — Which finds the modules and resolves the addresses.
DMMTRC — Which displays the trace table entries.

Called By

CMS via the DUMPSCAN command.

Error Messages

DMMDSC700I
DMMDSC701R
DMMDSC719I
DMMDSC720I
DMMDSC721I
DMMDSC722I
DMMDSC723I

DMMEDM — Edits and Prints a CP Dump

Entry Point

DMMEDM

Entry Conditions

R1: Address of option list
R13: SVC save area address
R14: Return address
R15: Entry point address

Exit Conditions

If an error is encountered reading the CP dump file (register 15 is nonzero), refer to the CMS RDBUF code meanings.

Routines Called

RDBUF — Via SVC to read in the dump file.
ERASE — Via SVC to delete the CP dump file from the P-disk.
CLOSIO — Via SVC to close out the printer.
PRINTR — Via SVC to print a line on the printer.
TYPLIN — Via SVC to write a message to the console.
DMMINI — To create a problem report.

Called By

CMS via the VMFDUMP command.

Error Messages

DMMEDM100S
DMMEDM200S
DMMEDM400S
DMMEDM850I
DMMEDM852I
DMMEDM853I
DMMEDM860I
DMMEDM861E
DMMEDM863E
DMMEDM864I

DMMFED — Displays 'nnn' Bytes from Address 'hexloc'

Entry Points

DMMFED -- Which formats the dump data.
DMMFEDLN – Which writes a line to the terminal.

Entry Conditions

Register 2 points to the parameter list with input truncated to 8-byte words.

Exit Conditions

R15: Return code

- 0 Good
- 4 Error in DMMGET accompanied by message DMMFEX702I or DMMFEX703I
- 8 Unrecoverable error

Routines Called

DMMGET — Which reads in an area of the dump.

Called By

DMMDSC, DMMFEX, DMMGET, DMMHEX, DMMOIB, DMMLOC, DMMMOD, DMMREG

Error Messages

DMMFED702I
DMMFED703I

DMMFEX — Displays X'130' Bytes of the Dump

Entry Point

DMMFEX

Entry Conditions

Register 2 points to the parameter list containing input truncated to 8-byte words.

Exit Conditions

R15: Return code

- 0 Good
- 4 Bad return from DMMGET or message DMMFEX704I is issued.

Routines Called

DMMDIR — Which formats and displays the data.

Called By

DMMDSC, DMMLOC

Error Messages

DMMFEX704I

DMMGET — Fetches Portions of the Dump into Storage

Entry Point

DMMGET

Entry Conditions

Register 2 contains the required dump address.

Exit Conditions

Register 2 contains the requested area's in storage address.

R15: Return code

0 Good

4 Warning message DMMGET708I is issued
R2 X'00' Page within dump not dumped
R2 X'FF' Page outside range of dump

8 Read error and message DMKGET100S is issued

Routines Called

DMMINT — Which translates the dump.

Called By

DMMFED, DMMFEX, DMMIOB, DMMLOC, DMMMOD, DMMREG, DMMSCR, DMMTRC, DMMVMB

Error Messages

DMMGET100S
DMMGET708I

DMMGRC — Reads Dump Record Containing Data at a Given Address and Passes Data Back to Caller

Entry Point

DMMGRC

Entry Conditions

The shared constant area contains an address at GRCPARM which is the requested data address.

Exit Conditions

Under normal conditions, register 1 points to the data read from the dump. Upon error return, control is passed to DMMPRM to prompt the user for information before quitting.

Routines Called

DMMPRM — Which handles errors if encountered (control not returned to DMMGRC).

Called By

DMMINI, DMMIDM, DMMCPA, DMMPRG

Error Messages

DMMGRC100S
DMMGRC809S

DMMHEX — Translates EBCDIC to Hexadecimal and Checks for Validity

Entry Point

DMMHEX

Entry Conditions

R3: Contains the count in bytes.
R4: Points to the leftmost byte of EBCDIC.

Exit Conditions

R5: Points to the leftmost byte of translated data.

R15: Return code

- 0 Good
- 4 Message DMMHEX714I is issued

Routines Called

DMMFED — Which displays the dump data line by line.

Called By

DMMDSC, DMMIOB, DMMLOC, DMMMOD

Error Messages

DMMHEX714I

DMMIDM — Determines the Failing or Calling Module Name and Displacement within the Module

Entry Point

DMMIDM

Entry Conditions

Register 1 points to a parameter list.

WORD 1 Failing address or base address of module.
WORD 2 Register 1 for the address of caller for non-program check condition.

Exit Conditions

The failing or calling module and displacement have been resolved and moved to the report. IF possible, an entry point name is also determined.

Routines Called

DMMGRC — Which reads the required record into the work buffer.
DMMTRN — Which translates the displacement from binary to a printable format.

Called By

DMMCPA, DMMPRG

Error Messages

None

DMMINI — Initializes for Data Extraction from the CMS File Containing the Dump

Entry Point

DMMINI

Entry Conditions

Register 1 contains the VMFDUMP parameter list.

Exit Conditions

Normal exit is to DMMEDM to process the CP spool file.

If the problem number file (SUMMFILE) retrieval results in an error, control is returned to DMMEDM and VMFDUMP processing is halted.

Routines Called

DMMPRG — Which extracts data for the CP program check.
DMMCPA — Which extracts data for the CP coded abend.
DMMMAP — Which compresses the load map.
DMMTRN — Which translates the data from binary to zoned.
DMMGRC — Which reads in the specified dump record.
DMMPRM — Which prompts the user for additional problem information.

Called By

DMMEDM

Error Messages

DMMINI100S
DMMINI400S
DMMINI800S
DMMINI803S

DMMINT — Translates the Binary Data to Printable Format

Entry Point

DMMINT

Entry Conditions

R3: Byte count
R4: Points to the input data string

Exit Conditions

R5: Points to the translated data

Routines Called

None

Called By

DMMDSC, DMMDIR, DMMFED, DMMLOC, DMMMOD, DMMREG, DMMTRC, and DMMVMB

Error Messages

None

DMMIOB — Displays the I/O Blocks

Entry Point

DMMIOB

Entry Conditions

Register 2 points to the parameter list with the input truncated to 8-byte words.

Exit Conditions

R15: Return code

0 Good

4 A bad return from DMMGET; message DMMIOB712I or DMMIOB713I was issued.

8 Unrecoverable error

Routines Called

DMMGET — Which fetches data into storage.
DMMHEX — Which converts EBCDIC to hexadecimal.

Called By

DMMDSC

Error Messages

DMMIOB712I
DMMIOB713I

DMMLOC — Locates 'string' 'from' 'to' 'increment'

Entry Point

DMMLOC

Entry Conditions

Register 2 points to a parameter list containing the command.

Exit Conditions

R15: Return code

0 String is found and data is displayed.
4 Message DMMLOC715I is issued; no data is displayed.
8 Error in DMSFREE or DMSFRET; no data is displayed.

Routines Called

DMMGET — Which fetches data into storage.
DMMINT — Which translates hexadecimal into EBCDIC.
DMMHEX — Which translates EBCDIC into hexadecimal.
DMMFEX — Which writes the found location to a screen.
DMMFED — Which writes the found location to a terminal.

Called By

DMMDSC

Error Messages

DMMLOC715I
DMMLOC716I
DMMLOC717I

DMMMAP — Appends Compressed and Sorted Load Map at End of Dump File

Entry Point

DMMMAP

Entry Conditions

Register 1 points to the parameter list. Word 1 of the parameter list points to the 12K output buffer.

Exit Conditions

R15: Return code

0 Normal completion (the load map information is appended to the dump).
8 Function not performed (error encountered).

Routines Called

None

Called By

DMMINI, DMMDSC

Error Messages

DMMMAP810S
DMMMAP200S
DMMMAP801I
DMMMAP802I
DMMMAP806R
DMMMAP807I
DMMMAP808I
DMMMAP810S

DMMMOD — Locates Modules and Entry Points in Load Map and Identifies Module Containing Given Address

Entry Point

DMMMOD

Entry Conditions

Register 2 points to the parameter list with input truncated to 8-byte words.

Exit Conditions

R15: Return code

0 Good

4 Message DMMMOD705E, DMMMOD706I, DMMMOD707I or DMMMOD718I is issued.

8 Unrecoverable error

Called By

DMMDSC

Error Messages

DMMMOD100I
DMMMOD705E
DMMMOD706I
DMMMOD707I
DMMMOD718I

DMMPRG — Handles the CP Program Check Processing

Entry Point

DMMPRG

Entry Conditions

The shared constant area contains information about the failure.

Exit Conditions

Exits to DMMPRM (the prompting subroutine) with the failing code in the text area.

Routines Called

DMMRMV — Which puts the registers in the output.
DMMGRC — Which gets the dump record containing the code.
DMMTRN — Which translates the failing code.
DMMPRM — Which prompts the user for any information concerning the problem.

Called By

DMMINI

Error Messages

None

DMMPRM — Prompts User for Supplementary Data Files and Textual Notes about Failure

Entry Point

DMMPRM

Entry Conditions

The common shared constant area contains information gathered by previous routines.

Exit Conditions

Exit to DMMEDM with the problem report created, the symptom summary file appended, and the summary record updated.

Routines Called

DMMWRT — Which writes the problem report to disk.
DMMSEA — Which performs the duplicate problem search.

Called By

DMMINI, DMMCPA, DMMPRG

Error Messages

DMMPRM200S
DMMPRM804I

DMMPRO — Creates a Problem Report through User Prompting

Entry Point

DMMPRO

Entry Conditions

Entry from CMS when PROB command is entered.

Exit Conditions

R15: Return code

- 0 Normal completion
- 4 The user entered 'HX' (halt execution)
- 8 Unrecoverable error

Routines Called

DMMWRT – Which writes the problem report to disk.
DMMSEA – Which looks for a duplicate of this problem.

Called By

By CMS when the PROB command is entered.

Error Messages

DMMPRO100S
DMMPRO200S

DMMREG – Displays the Registers

Entry Point

DMMREG

Entry Conditions

Register 2 points to the parameter list with input truncated to 8-byte words.

Exit Conditions

R15: Return code

- 0 Good
- 4 Warning
- 8 Unrecoverable error

Routines Called

DMMINT – Which translates hexadecimal to EBCDIC.
DMMFED – Which displays the dump data line-by-line.
DMMGET – Which fetches the dump pages into storage.

Called By

DMMDSC

Error Messages

DMMREG100S

DMMRMV — Places Registers in the Text Area of the Report

Entry Point

DMMRMV

Entry Conditions

Register 1 points to the savearea for one of the following save area sets:

BALR
FREE
General registers
savearea

Exit Conditions

The registers are in the text area of the report.

Routines Called

DMMTRN — Which translates the registers into a printable format.

Called By

DMMCPA, DMMPRG

Error Messages

None

DMMSCR — Scrolls the Display Up or Down from the Last Address

Entry Point

DMMSCR

Entry Conditions

Register 2 points to the parameter list containing the input truncated to 8-byte words.

Exit Conditions

R15: Return code

0 Good
4 Message DMMSCR709I is issued; bad return from DMMGET.
8 Unrecoverable error

Routines Called

DMMGET – Which fetches data into storage.
DMMDIR – Which formats and displays data.

Called By

DMMDSC

Error Messages

DMMSCR709I

DMMSEA – Locates any Problems which are Duplicates of a Newly Entered Problem

Entry Point

DMMSEA

Entry Conditions

Register 1 points to a parameter list as follows:

WD1 Pointer to the internal data area
WD2 Pointer to the keyword string (with length fields)
WD3 Pointer to the text area for this problem

Exit Conditions

R15: Return code
- 0 No duplicates found
- 4 Duplicates found
- 8 Unrecoverable error encountered

Routines Called

DMMSUM – Which posts duplicate status of the problem to the summary control record.

Called By

DMMPRM, DMMPRO

Error Messages

DMMSEA100S

DMMSTA — Displays the Status of a Given Problem or Group of Problems or all Problems

Entry Point

DMMSTA

Entry Conditions

Register 1 points to the passed parameters:

Module name	length 8
PARM1 (ALL or PNUM)	length 8
PARM2 (SRCH ARG1)	length 8
PARM3 (SRCH ARG2)	length 8

Exit Conditions

R15: Return code

- 0 Normal return, function performed
- 4 Problem number not found in symptom summary
- 8 Unrecoverable error encountered

Routines Called

None

Called By

CMS via the STAT command.

Error Messages

DMMSTA100S
DMMSTA200S
DMMSTA601S

DMMSUM — Updates or Finds Symptom Summary Control Record for a Given Problem and Passes it to Caller

Entry Point

SUMMARY

Entry Conditions

Register 1 contains the parameter pointer as follows:

8 characters	not used
8 characters	PRBxxxxx, where xxxxx is the problem number
8 characters	The function to be performed (UPcccccc) or FI where cccccc is EG, STAT, PTF, or FUNCT, and UP and FI stand for update and find
n characters	New data to be put in the appropriate field

Exit Conditions

R15: Return code

- 0 Normal successful completion
- 4 Requested problem not found
- 8 Unrecoverable error encountered

If a FIND was requested, register 1 points to the problem control record.

Routines Called

None

Called By

DMMSEA and PRB EXEC

Error Messages

DMMSUM100S
DMMSUM200S
DMMSUM501S
DMMSUM502S

DMMTRC — Displays 'nnn' Trace Entries

Entry Point

DMMTRC

Entry Conditions

Register 2 points to the parameter list.

Exit Conditions

Register 2 contains the last displayed scroll address.

R15: Return code

0 Good
4 Bad return from DMMGET, message DMMTRC710I is issued.

Routines Called

DMMGET — Which fetches the dump pages into storage.
DMMFED — Which displays the specified areas.

Called By

DMMDSC

Error Messages

DMMTRC710I

DMMTRN — Translates Binary Data into a Printable Format

Entry Point

DMMTRN

Entry Conditions

The common constant area TRNPARM and TRNPARM1 have the data length and data address respectively.

Exit Conditions

Register 1 points to the translated data.

Routines Called

DMMINI, DMMCPA, DMMPRG, DMMRMV, and DMMIDM

Error Messages

None

DMMVMB — Displays all VMBLOK Addresses, Userids, and Status

Entry Point

DMMVMB

Entry Conditions

None

Exit Conditions

R15: Return code

0 Good
4 Bad return from DMMGET, message DMMVMB711I issued.

Routines Called

DMMGET — Which fetches the dump into storage.
DMMINT — Which translates hexadecimal into EBCDIC.

Called By

DMMDSC

Error Messages

DMMVMB711I

DMMWRT — Creates a Problem Report on Disk and Adds this Problem to the Symptom Summary File

Entry Point

DMMWRT

Entry Conditions

Register 1 points to the parameter list as follows:

WD1 Points to the internal data (DSECT INTSECT)
WD2 Points to the keyword data (variable blocked format)
WD3 Points to the text description (halfword length prefix)
WD4 Points to the supplementary data (halfword length prefix)

Exit Conditions

R15: Return code

0 Normal, successful completion
8 Error occurred

Routines Called

None

Called By

DMMPRO (PROB command), and DMMPRM (during VMFDUMP processing)

Error Messages

DMMWRT200S

Directory

Figure 3-2 is an alphabetical list of some of the labels in the IPCS modules. The function performed at the point in the program indicated by each label is described and the associated method of operation diagram is referenced.

Label	Module	Diagram	Description
CPACALL	DMMCPA	3-8	Calls the user prompt routine.
CPACNTIN	DMMCPA	3-8	Scans the abend look up table.
CPALKUP	DMMCPA	3-8	Codes the abend look up table.
CHECKTWO	DMMDSC	3-1	Turns print on and off.
CLOSEPRT	DMMDSC	3-1	Issues the DIAGNOSE X'08' subcommand to close print.
ENTER	DMMDSC	3-1	Prompts for the dump name and file type.
FOUNDAMP	DMMDSC	3-1	Calls an entry from the &NAME table.
GETDUMP	DMMDSC	3-1	Prompts the user who has asked for HELP.
INCHECK	DMMDSC	3-1	Determines if the entry is HELP, QUIT, HX
MAPCHECK	DMMDSC	3-1	Ensures that the dump has no map.
NOTINTAB	DMMDSC	3-1	Adds the entry to the &NAME table.
PLISTSCN	DMMDSC	3-1	Examines the parameter to determine what
READIN	DMMDSC	3-1	Issues an RDTERM to accept subcommands.
RESUBCOM	DMMDSC	3-1	Reissues the previous subcommand.
SHOWPRSW	DMMDSC	3-1	Displays the print status.
SHOWTAB	DMMDSC	3-1	Displays a list of table entries.
SUBCOM	DMMDSC	3-1	Issues the print subcommand.
TOKEN	DMMDSC	3-1	Groups the input in an 8-byte parameter.
CHKOPT	DMMEDM	3-5	Checks the user options for accuracy.
CORTBL	DMMEDM	3-11	Edits and prints the storage table.
DMPEND1	DMMEDM	3-5	Issues the end-of-file message.
EDITDUMP	DMMEDM	3-5	Prints preliminary information.
ERASE	DMMEDM	3-5	Erases the CMS file containing the dump.
ERROP	DMMEDM	3-5	Issues message DMMEDM863I.
ERROR3	DMMEDM	3-5	Issues message DMMEDM861I.
GETKEY	DMMEDM	3-12	Constructs the print line showing the storage keys.
GETKEY1	DMMEDM	3-12	Prints a storage line.
HEXDUMP	DMMEDM	3-5	Prints all of storage.
IOBPROG	DMMEDM	3-11	Formats and prints the IOBLOK.
LOADMAP	DMMEDM	3-10	Ensures that the symbol table is requested.

Figure 3-2 (Part 1 of 4). The Interactive Problem Control System (IPCS) Label Directory

Label	Module	Diagram	Description
LOOP	DMMEDM	3-5	Creates a CMS file.
NODMP	DMMEDM	3-5	Issues message DMMEDM853I.
NXTWD	DMMEDM	3-5	Creates the item table from the bit map.
PRELIMII	DMMEDM	3-10	Changes the PSWs.
PRELIM4	DMMEDM	3-10	Prints the general and control registers.
PRELIM8	DMMEDM	3-10	Prints the floating point registers.
PREREC	DMMEDM	3-10	Prints the heading.
RCHFORM	DMMEDM	3-5	Prints the real control blocks.
RCHPROC	DMMEDM	3-11	Prints RCHBLOKs.
RCUINIT	DMMEDM	3-11	Prints RCUBLOKs.
RDEVINIT	DMMEDM	3-11	Prints the RDEVBLOKs.
RDUMP	DMMEDM	3-5	Writes the CMS file containing the CP dump.
READPAGE	DMMEDM	3-12	Reads the dump file storage pages.
REREAD	DMMEDM	3-5	Reads the operator response.
RETN	DMMEDM	3-5	Saves the file if ERASE is not specified.
SETEDM	DMMEDM	3-5	Opens the dump file.
SFFORM	DMMEDM	3-11	Prints the SFBLOK chains for unit record I/O.
TSTSPOOL	DMMEDM	3-11	Prints VSPLCTL and SFBLOKs.
VCHINIT	DMMEDM	3-11	Formats and prints VCHBLOKs.
VCUINIT	DMMEDM	3-11	Formats and prints VCUBLOKs.
VINIT	DMMEDM	3-11	Formats and prints VDEVBLOKs.
VIRTUALM	DMMEDM	3-11	Prints virtual control blocks.
VMCK	DMMEDM	3-11	Prints segment, page, and swap tables.
VMPRINT	DMMEDM	3-11	Formats and prints VMBLOKs.
CTRAN	DMMFED	3-1	Converts the count to hexadecimal.
DIRECT	DMMFED	3-1	Formats the dump data.
SCRNFULL	DMMFED	3-1	Writes the dump data.
TRANADD	DMMFED	3-1	Converts the address to hexadecimal.
DIRECT	DMMFEX	3-1	Calls DMMDIR to display the dump data.
INDIR	DMMFEX	3-1	Checks for indirect requests.
NOROUND	DMMFEX	3-1	Calls DMMGET to get the dump data.
TRANADD	DMMFEX	3-1	Converts the address to hexadecimal.
MODREAL	DMMIDM	3-7	Determines the address location.

Figure 3-2 (Part 2 of 4). The Interactive Problem Control System (IPCS) Label Directory

Label	Module	Diagram	Description
EXTLEAV	DMMINI	3-9	Processes the operator initiated dump.
EXTLPWT	DMMINI	3-9	Examines wait bit in PSW.
EXTPSCHK	DMMINI	3-9	Processes operator initiated dump.
EXTPSWCK	DMMINI	3-5	Finds the failure type.
EXTREAD	DMMINI	3-5	Initializes the buffers for DUMPINREC.
EXTREND	DMMINI	3-5	Calls the map compression routine.
EXTSVCHK	DMMINI	3-5	Calls the coded abend routine.
EXTTRTAB	DMMINI	3-9	Locates the last four trace entries.
COMPRCUB	DMMIOB	3-1	Gets addresses for the real control
VIO	DMMIOB	3-1	Gets the addresses for the virtual
DMMLOC	DMMLOC	3-1	Fetchs the 'from' page.
EXECUTOR	DMMLOC	3-1	Compares the string to the dump.
GOGOFEX	DMMLOC	3-1	Displays the equal compare.
MAPERROR	DMMMAP	3-6	Issues message DMMMAP808I.
MAPNAME	DMMMAP	3-6	Issues message DMMMAP806R.
READ	DMMMAP	3-6	Reads in 'NUC MAP A'.
READERR	DMMMAP	3-6	Issues message DMMMAP807I.
WRTOUT	DMMMAP	3-6	Adds the compressed nucleus map to the dump.
XCK	DMMMAP	3-6	Checks the nucleus load map for validity.
DMMMOD	DMMMOD	3-1	Reads the load map.
MAPRED	DMMMOD	3-1	Scans the load map.
PAGEMOD	DMMMOD	3-1	Checks for a pageable module.
QREQUEST	DMMMOD	3-1	Scans the load map.
READ	DMMMOD	3-1	Reads the load map.
PRGMORCD	DMMPRG	3-7	Calls the user prompt routine.
EXIT	DMMPRM	3-5	Returns to the VMFDUMP print routine.
NORMEXIT	DMMPRM	3-5	Calls the write and search routines.
EXIST	DMMPRO	3-3	Prompts to determine new or old problem.
GETFAIL	DMMPRO	3-3	Prompts for specific type of problem.
GETSDATA	DMMPRO	3-3	Gets supporting data file names.
MAINLINE	DMMPRO	3-3	Gathers general problem data.
OLDADD	DMMPRO	3-3	Appends information to the problem report.

Figure 3-2 (Part 3 of 4). The Interactive Problem Control System (IPCS) Label Directory

Label	Module	Diagram	Description
OLDPROB	DMMPRO	3-3	Gets the number of the old problem.
SRCHRTN	DMMPRO	3-3	Calls DMMSEA for a duplicate search.
TEXTENTR	DMMPRO	3-3	Prompts for free form text information.
DMMREG	DMMREG	3-1	Checks for AP or UP dump.
DIRECTIT	DMMSCR	3-1	Formats and displays.
DMMSCR	DMMSCR	3-1	Calculates new display address.
ENDRTN	DMMSEA	3-3	Updates symptom summary with duplicate entry.
PUTOUT	DMMSEA	3-3	Notifies user of duplicate problem.
START	DMMSEA	3-3	Searches for a duplicate problem.
SPNUM	DMMSTA	3-4	Prompts for the problem number.
STALLOC	DMMSTA	3-4	Creates the STATALL LOCAL file.
START	DMMSTA	3-4	Checks the first operand for validity.
STATRDY	DMMSTA	3-4	Displays the status of a given problem.
STATSRCH	DMMSTA	3-4	Searches for the type of status record.
REPORT1	DMMSUM	3-2	Appends status change to problem report.
START	DMMSUM	3-2	Determines the type of request.
DMMTRC	DMMTRC	3-1	Finds the trace table.
BALGET	DMMVMB	3-1	Chains through the VMBLOKs.
DMMVMB	DMMVMB	3-1	Gets the system VMBLOK pointer.
MOVEL	DMMVMB	3-1	Prints the VMBLOK list.
CNTRLOUT	DMMWRT	3-3	Adds PROB control record to the
INTOUT	DMMWRT	3-3	Writes the first three report records.
KEYOUT	DMMWRT	3-3	Adds keyword data to the report.
SUPPOUT	DMMWRT	3-3	Adds supplementary file names to report.
TEXTOUT	DMMWRT	3-3	Adds free form text to report.
-APAR	PRB EXEC	3-2	Posts the PARM number.
-CLOSE	PRB EXEC	3-2	Closes the problem.
-DUPOF	PRB EXEC	3-2	Posts the problem as a duplicate.
-IBM	PRB EXEC	3-2	Posts the report to IBM.
-MORE	PRB EXEC	3-2	Indicates that more information is needed.
-PTFIS	PRB EXEC	3-2	Posts the PTF number.
-PTFON	PRB EXEC	3-2	Applies the posted PTF.
-RETRY	PRB EXEC	3-2	Checks the operand for validity.
-SEV	PRB EXEC	3-2	Changes the severity.
-USER	PRB EXEC	3-2	Posts the problem as the user's responsibility.

Figure 3-2 (Part 4 of 4). The Interactive Problem Control System (IPCS) Label Directory

Data Areas

This section describes the data areas used by the Interactive Problem Control System (IPCS). The data areas are:

SHARECON — VMFDUMP Shared Constant Area

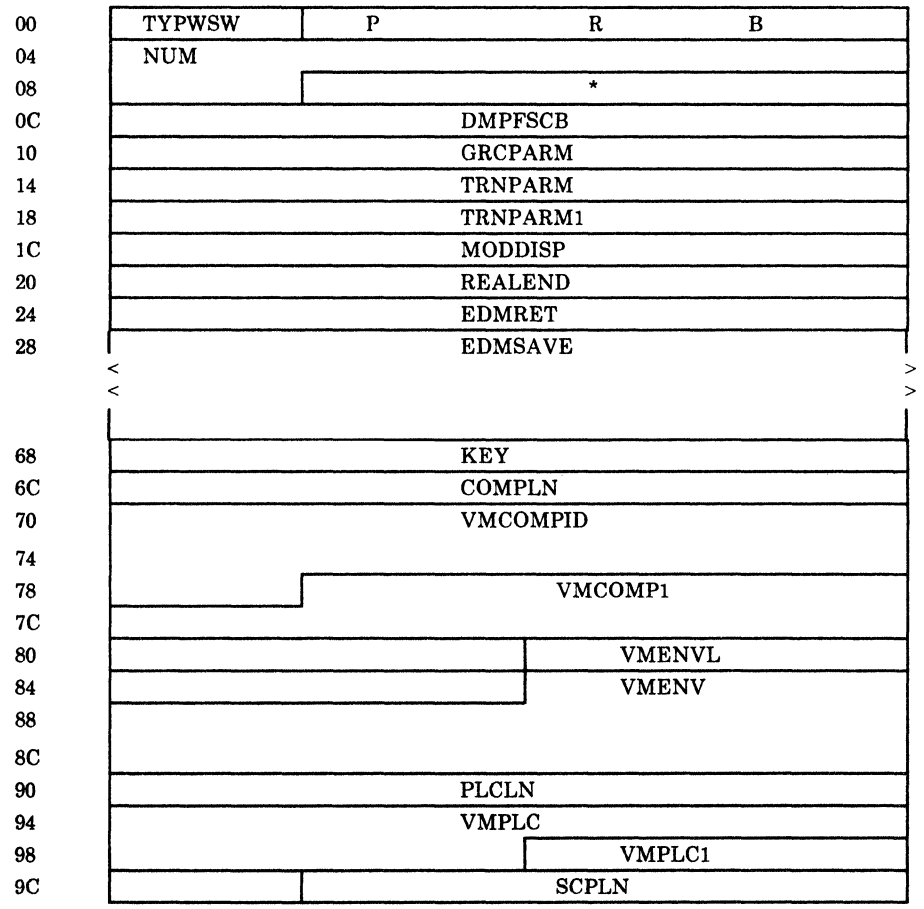


Figure 3-3 (Part 1 of 4). VMFDUMP Shared Constant Area

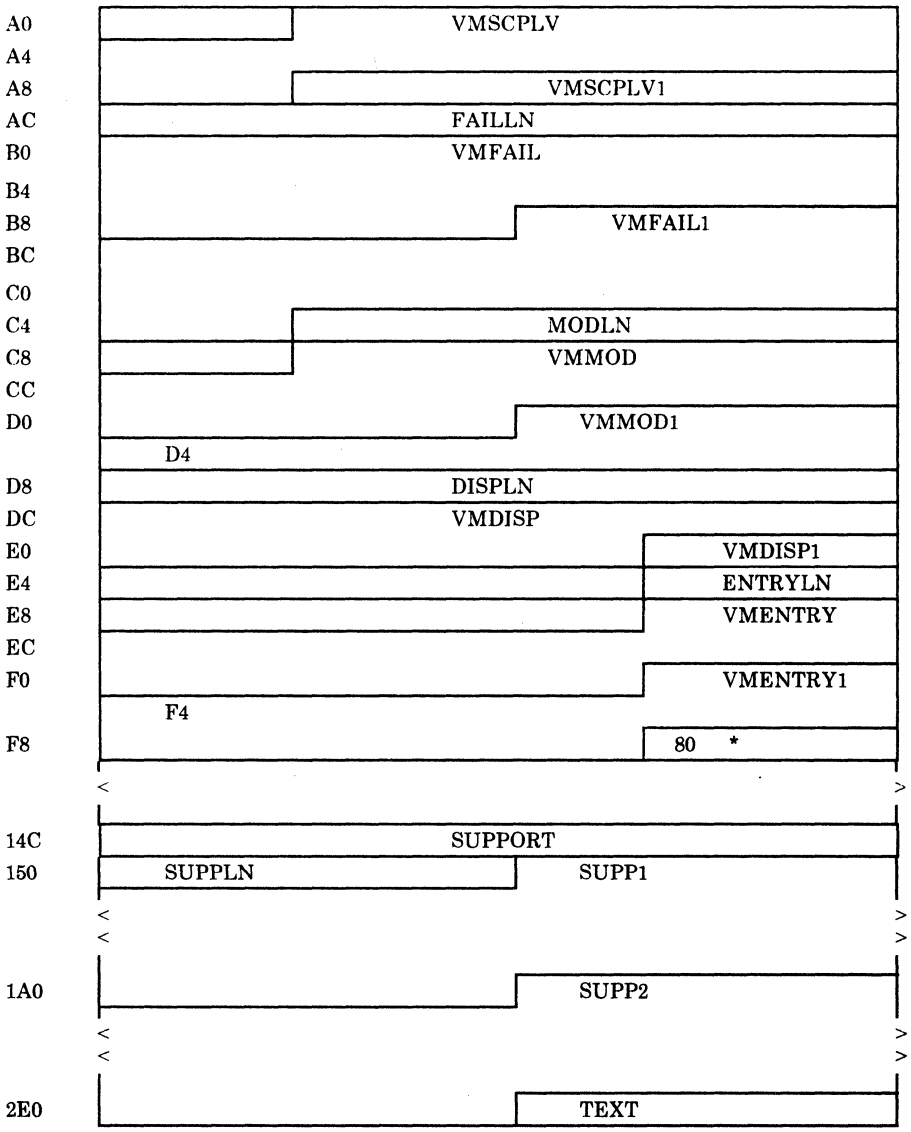
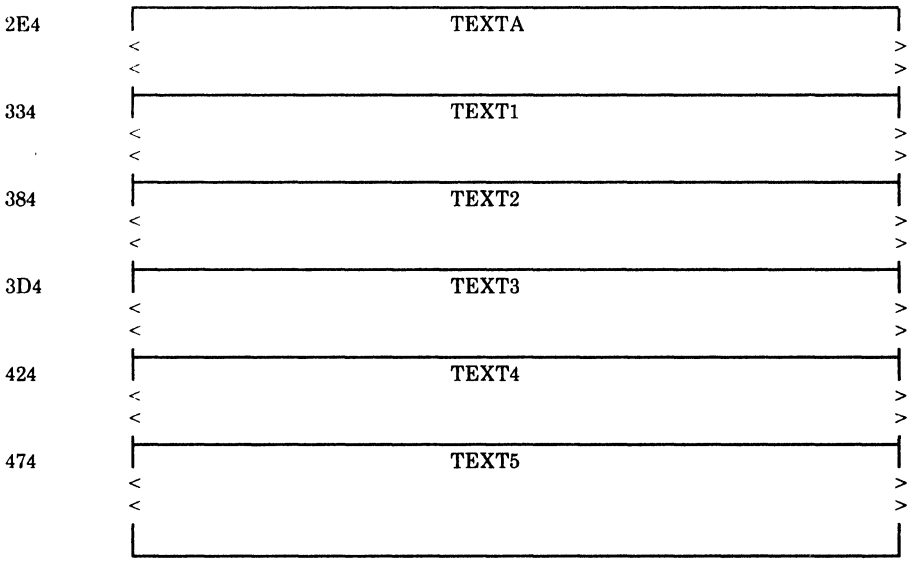


Figure 3-3 (Part 2 of 4). VMFDUMP Shared Constant Area



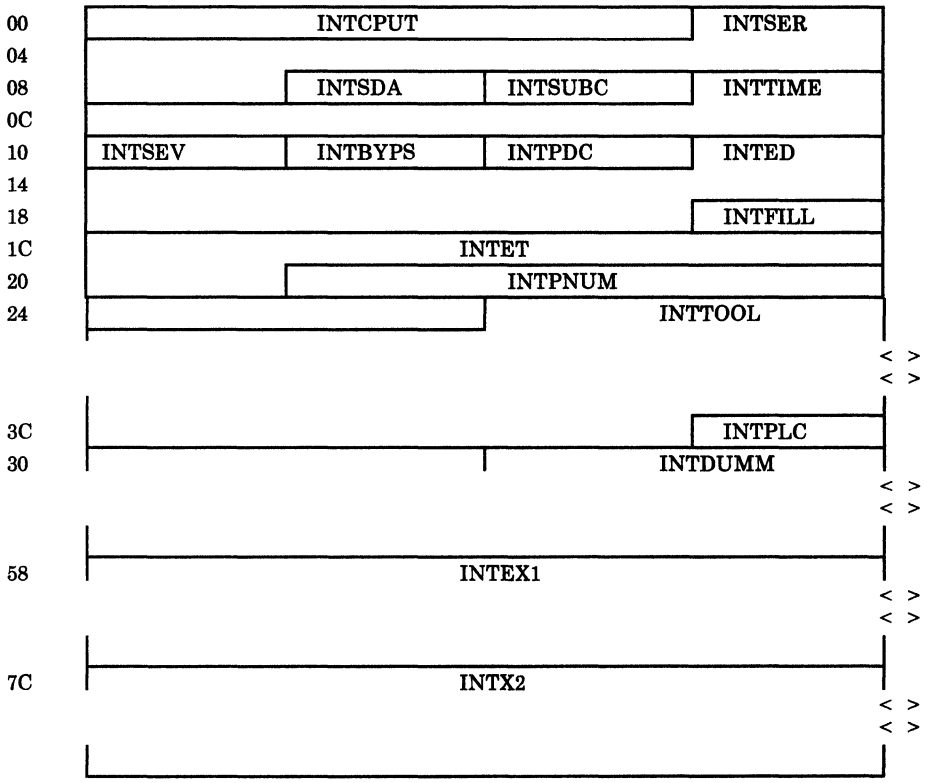
Displacement Hex	Dec	Field Name		Description	
00	00	TYPESW	DC	X'00'	Dump type switch
		WAITSW	EQU	X'01'	Wait
		LOOPSW	EQU	X'02'	Loop or performance
		PRGCKSW	EQU	X'04'	Program check
		CPABSW	EQU	X'08'	CP coded abend
		PROCERR	EQU	X'80'	Data extraction process error
01	01	DUMPNUM	DS	0CL8	Unique problem identification assigned
01	01		DC	C'PRB'	Problem number prefix
04	04	NUM	DC	C'00000'	Problem number
0C	12	DMPFSCB	DC	F'0'	Address of dump read FSCB
10	16	GRCPARM	DC	F'0'	GETREC parameter list address
14	20	TRNPARM	DC	F'0'	Translate routine PARM2 (data length)
18	24	TRNPARM1	DC	F'0'	Translate routine PARM1 (address)
1C	28	MODDISP	DS	F	Displacement of failure in module
20	32	REALEND	DS	F	Highest address of fixed storage
24	36	EDMRET	DS	F	Return address in DMMEDM
28	40	EDMSAVE	DS	16F	Save area for DMMEDM
113 **** THE PROBLEM REPORT KEYWORD AREA FOLLOWS ****					
68	104	KEY	DC	AL2(VMKEYS-KEY),X'0000'	Set initial length
6C	108	COMPLN	DC	X'00160000'	Length of component id keyword
70	112	VMCOMPID	DC	C'VMCOMPID='	Component id keyword
79	121	VCOMP1	DC	C'5749DMK00'	VM/370 component id
82	130	VMENVL	DC	X'000E0000'	Length of environment key
86	134	VMENV	DC	C'VMENVIR=CP'	Environment keyword
90	144	PLCLN	DC	X'000D0000'	Length of PLC keyword
94	148	VM PLC	DC	C'VMPLC='	PLC keyword
9A	154	VM PLC1	DS	CL3	PLC number
9D	157	SCPLN	DC	X'000F0000'	Length of SCP keyword area
A1	161	VM SCPLV	DC	C'VMSCPLV='	SCP keyword
A9	169	VM SCPLV1	DS	CL3	SCP number
AC	172	FAILLN	DC	X'00190000'	Length of failure keyword area

Figure 3-3 (Part 3 of 4). VMFDUMP Shared Constant Area

Displacement Hex	Dec	Field Name		Description	
B0	176	VMFAIL	DC	C'VMFAILURE='	Failure keyword
		VMFAILLP	EQU	*+4	End of VMFAIL if loop or performance failure
		VMFAILOT	EQU	*+5	End of VMFAIL if other failure
BA	186	VMFAIL1	DS	CL11	Failure type
		VMKEYS	EQU	*	Length of base keys
		**** OTHER KEYWORDS WHICH MAY OR MAY NOT BE USED FOLLOW****			
C5	197	MODLN	DC	X'00130000'	Length of module name
C9	201	VMMOD	DC	C'VMMODULE='	Module name keyword
D2	210	VMMOD1	DS	CL6	Failing module name
D8	216	DISPLN	DC	X'000F0000'	Displacement key area length
DC	220	VMDISP	DC	C'VMDISP='	Displacement keyword
E3	227	VMDISP1	DS	CL4	Displacement
E7	231	ENTRYLN	DC	X'00140000'	Entry point key area length
EB	235	VMENTRY	DC	C'VMENTRY='	Entry point keyword
F3	243	VMENTRY1	DS	CL8	Entry point address
FB	251		DC	80C' '	Padded for additional key
14C	332	SUPPORT	DS	F	Supporting data area
150	336	SUPPLN	DC	X'0050'	Supplementary data area length initially set
152	338	SUPP1	DC	80C' '	Supporting data reserved fora the dump fileid
1A2	418	SUPP2	DC	320C' '	User supporting data area
		**** THE PROBLEM REPORT TEXT AREA FOLLOWS: ****			
2E2	738	TEXT	DC	X'01E0'	Text area length initially set to 48
		**** THE FIRST 480 BYTES RESERVED FOR THE EXTRACTION PROGRAM ****			
2E4	740	TEXTA	DC	80C' '	Data extraction text line a
334	820	TEXT1	DC	80C' '	Data extraction text line 1
384	900	TEXT2	DC	80C' '	Data extraction text line 2
3D4	980	TEXT3	DC	80C' '	Data extraction text line 3
424	1060	TEXT4	DC	80C' '	Data extraction text line 4
474	1140	TEXT5	DC	80C' '	Data extraction text line 5
		**** USER TEXT AREA UP 15 LINES OF 80 BYTE ENTRIES ****			

Figure 3-3 (Part 4 of 4). VMFDUMP Shared Constant Area

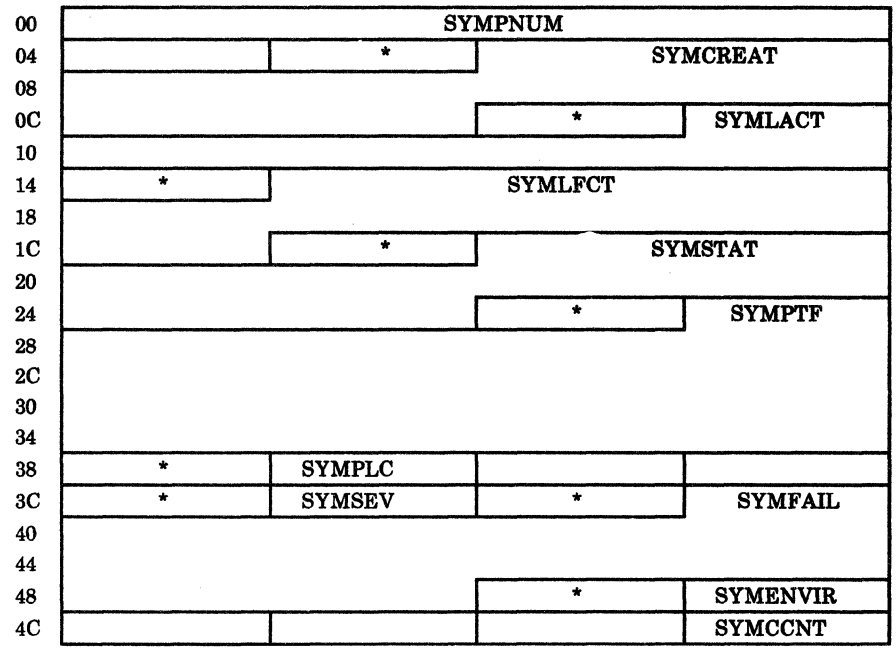
INTSECT – VMFDUMP and PROB Internal Data Area



Displacement Hex	Dec	Field Name		Description
00	00	INTSECT	DSECT	
00	00	INTCPUT	DS CL3	CPU type
03	03	INTSER	DS CL6	CPU serial
09	09	INTSDA	DS CL1	Support data available switch
0A	10	INTSUBC	DS CL1	Submitter's code
0B	11	INTTIME	DS CL5	Total time expended
10	16	INTSEV	DS CL1	Problem severity
11	17	INTBYPS	DS CL1	Bypass required switch
12	18	INTPDC	DS CL1	Filler
13	19	INTERRT	DS 0CL14	Error date and time
13	19	INTED	DS CL8	Date of error
1B	27	INTFILL	DS CL1	Filler
1C	28	INTET	DS CL5	Time of error
21	33	INTPNUM	DS CL5	System-assigned problem number
26	38	INTTOOL	DS CL25	Tool usage codes
3F	63	INTPLC	DS CL3	PLC level of system
42	66	INTDUMM	DS CL22	Filler
58	88	INTX1	DS CL36	VMFAILURE (internal use only)
7C	124	INTX2	DS CL36	VMENVIR (internal use only)

Figure 3-4. VMFDUMP and PROB Internal Data Area

SYMSECT — Symptom Summary Control Record Format



Displacement Hex	Dec	Field Name			Description
00	00	SYMSECT	DSECT		
00	00	SYMPNUM	DS	CL5	Problem number
05	05		DS	CL1	Filler
06	06	SYMCREAT	DS	CL8	Creation date for this problem
0E	14		DS	CL1	Filler
0F	15	SYMLACT	DS	CL5	Date of last activity
14	20		DS	CL1	Filler
15	25	SYMLFCT	DS	CL8	Last activity performed
1D	29		DS	CL1	Filler
1E	30	SYMSTAT	DS	CL8	Current status of this problem
26	38		DS	CL1	Filler
27	39	SYMPTF	DS	0CL17	Filename and filetype of PTF for this problem or PNUM or duplicate problem
27	39	SYMPTFFN	DS	CL8	PTF filename
2F	47	SYMPTFDV	DS	CL1	Divider between filename and filetype
30	48	SYMPTFFT	DS	CL8	PTF filetype

Figure 3-5 (Part 1 of 2). Symptom Summary Control Record Format

Displacement Hex	Dec	Field Name			Description
38	56		ORG	SYMPTF	
27	39	SYMAPAR1	DS	CL4	Place for APAR
2B	43	SYMAPARX	DS	CL1	Blank divider
2C	44	SYMAPAR2	DS	CL8	APAR number
34	52		ORG	SYMPTF	
27	39	SYMCLOSE	DS	CL17	Closing code if not resolved
38	56		ORG	SYMPTF	
27	39	SYMDUP	DS	0CL17	Field to flag problem as duplicate
27	39	SYMDUP1	DS	CL7	'DUP OF'
2E	46	SYMDUP2	DS	CL5	Problem number of duplicate
33	51	SYMDUPX	DS	CL5	Filler
38	56		DS	CL1	Filler
39	57	SYMPLC	DS	CL3	PLC level of system
3C	60		DS	CL1	Filler
3D	61	SYMSEV	DS	CL1	Severity of this problem
3E	62		DS	CL1	Filler
3F	63	SYMFAIL	DS	CL11	Value of keyword VMFAILURE
4A	74		DS	CL1	Filler
4B	75	SYMENVIR	DS	CL4	Value of keyword VMENVIR
4F	79	SYMCCNT	DS	CL1	Number of keyword symptom records following this header

Figure 3-5 (Part 2 of 2). Symptom Summary Control Record Format

The following CP and CMS data areas are used by IPCS:

CMS

NUCON	CMS low core constant area
FSCB	File system control block

CP

VMBLOK	Virtual machine block
PSA	Prefix storage area
RDEVBLOK	Real device block
RCUBLOK	Real control unit block
RCHBLOK	Real channel block
IOBLOK	I/O control block
BSCBLOK	Binary synchronous control block
VDEVBLOK	Virtual device block
VCUBLOK	Virtual control unit block
VCHBLOK	Virtual channel block
DMPINREC	Dump file information record
DMPKYREC	Dump file key record
DMBTBREC	Dump file symbol table record

These data areas are described in the *VM/SP HPO Data Areas and Control Block Logic – CP.*

Diagnostic Aids

Figure 3-6 is an alphabetical list of all the messages issued by IPCS. The nearest label and the associated method of operation diagram are identified.

Message Code	Label	Diagram	Message Text
DMMCPA805I	CPAEND	3-8	xxxxxx ABEND CODE NOT RECOGNIZED BY DATA EXTRACTION
DMMDSC700I	GETDUMP	3-1	TYPE HELP OR ENTER
DMMDSC701R	ENTER	3-1	ENTER DUMP NUMBER AND MODE
DMMDSC719I	STATERR	3-1	ERROR IN FSSTATE
DMMDSC720I	TWOMAPS	3-1	LOAD MAP ALREADY PRESENT
DMMDSC721I	FULLMSG	3-1	&NAME TABLE IS FULL
DMMDSC722I	ILLEGAMP	3-1	INVALID ENTRY INTO THE &NAME TABLE
DMMDSC723I	NOWRITE	3-1	THE DUMP IS NOT ON THE A-DISK
DMMEDM100S	PRBQUITR	3-5	ERROR 'nnn' READING FILE 'SUMMARY RECORD A'
DMMEDM200S	PRBWRTER	3-5	ERROR 'nnn' WRITING FILE 'SUMMARY RECORD A'
DMMEDM400S	PRBQUITC	3-5	ERROR 'nnn' CLOSING FILE 'SUMMARY RECORD A'
DMMEDM850I	LOOP	3-5	UNABLE TO READ DUMP FROM READER
DMMEDM851I			TEN DUMP FILES ALREADY EXIST
DMMEDM852I	ERRWRT	3-5	FATAL I/O ERROR WRITING DUMP
DMMEDM853I	NODMP	3-5	NO DUMP FILES EXIST
DMMEDM860I	QUIT	3-5	FATAL I/O ERROR READING DUMP
DMMEDM861E	ERROR3	3-5	DUMP FILE 'filename' NOT FOUND
DMMEDM863E	ERRFND	3-5	INVALID PARAMETER - 'parameter'
DMMFED702I	CBADIN	3-1	NON-HEX CHARACTER IN COUNT - RETRY
DMMFED703I	BADIN	3-1	NON-HEX CHARACTER IN ADDRESS - RETRY
DMMFEX704I	BADIN	3-1	NON-HEX CHARACTER IN ADDRESS - RETRY
DMMGET100S	RDERR		ERROR 'nnn' READING FILE 'fileid'
DMMGET708I	MSGITEND		PAGE 'page' NOT FOUND IN THE DUMP
DMMGRC100S	RDERROR		ERROR 'nnn' READING FILE 'PRBnnnnn DUMP A1'
DMMGRC809S	MSGITEND		REQUESTED ADDRESS NOT IN DUMP
DMMHEX714I	BADIN	3-1	NON-HEX CHARACTER IN INPUT - RETRY
DMMINI100S	EXTERR	3-5	ERROR 'nnn' READING FILE 'PRBnnnnn DUMP A1'
DMMINI400S	EXTERRC	3-5	ERROR 'nnn' CLOSING FILE 'PRBnnnnn DUMP A1'
DMMINI800S	WRTMSG	3-5	DATA EXTRACTION FAILURE
DMMINI803S	ERR202	3-5	ERROR 'nnn' ATTEMPTING LOADMOD FOR VMFDUMP2
DMMIOB712I	WRITERR	3-1	DEVICE 'cuu' NOT FOUND
DMMIOB713I	NOUSER	3-1	USER 'userid' VMBLOK NOT FOUND

Figure 3-6 (Part 1 of 2). Interactive Problem Control System Messages

Message Code	Label	Diagram	Message Text
DMMLOC715I	NONHEXST	3-1	NON-HEX CHARACTER IN STRING
DMMLOC716I	MOREMSG	3-1	STRING 'string' NOT FOUND
DMMLOC716I	NOTOFF	3-1	STRING 'string' NOT FOUND BEFORE END OF DUMP
DMMLOC717I	NOPARM	3-1	INVALID FORM OF LOCATE COMMAND
DMMMAP200S	WRTERR	3-6	ERROR 'nnn' WRITING FILE 'PRBnnnnn DUMP A1'
DMMMAP801I	MAPERROR	3-6	FILE 'NUC MAP' IS NOT VALID FOR THIS DUMP
DMMMAP802I	CONTMSG	3-6	PROCEEDING..
DMMMAP806R	MAPNAME	3-6	ENTER fn ft fm OF THE NUCLEUS LOAD MAP
DMMMAP807I	STATERR	3-6	UNABLE TO LOCATE 'fileid'
DMMMAP808I	MAPERROR	3-6	NUCLEUS MAP INVALID 'fileid'
DMKMAP810S	READERR	3-6	ERROR 'nnn' READING FILE 'fileid'
DMMMOD100I	RDERR	3-1	ERROR 'nnn' READING FILE 'fileid'
DMMMOD705E	EOTAB	3-1	ERROR IN ITEM TABLE
DMMMOD706I	MODNF	3-1	'entry name' NOT FOUND IN THE LOAD MAP
DMMMOD707I	GETREAL	3-1	'module' 'page' PAGE NOT VALID
DMMMOD718I	LMERR	3-1	THIS DUMP HAS NO LOAD MAP - SEE MAP SUBCOMMAND
DMMPRM200S	PRBWRTER	3-5	ERROR 'nnn' WRITING FILE 'SUMMARY RECORD A1'
DMMPRM804I	START	3-5	ERROR IN DATA EXTRACTION
DMMPRO100S	SUMERRR2	3-3	ERROR 'nnn' READING FILE 'fileid'
DMMPRO200S	SUMERRW	3-3	ERROR 'nnn' WRITING FILE 'fileid'
DMMREG100I	RDERR	3-1	ERROR 'nnn' READING FILE 'fileid'
DMMSCR709I	EYECATCH	3-1	NO VALID SCROLL ADDRESS
DMMSEA100S	RDERR	3-1	ERROR 'nnn' READING FILE 'SYMPTOM SUMMARY A1'
DMMSTA100S	RDERR4	3-4	ERROR 'nnn' READING FILE 'fileid'
DMMSTA200S	WRTERR	3-4	ERROR 'nnn' WRITING FILE 'fileid'
DMMSTA601S	CK2CONT	3-4	OPERAND NOT RECOGNIZED, STATALL ASSUMED
DMMSUM100S	RDERR	3-2	ERROR 'nnn' READING FILE 'SYMPTOM SUMMARY A1'
DMMSUM200S	ERRWRT	3-2	ERROR 'nnn' WRITING FILE 'fileid'
DMMSUM501S	PARMERR	3-2	INVALID PARM 'parm' PASSED TO SUMMARY UPDATE PROGRAM
DMMSUM502S	RETCOD4	3-2	PROBLEM 'PRBnnnnn' NOT FOUND IN SYMPTOM SUMMARY
DMMTRC710I	CBADIN	3-1	NON-NUMERIC COUNT CHARACTER - RETRY
DMMVMB711I	NBA	3-1	LOOP IN VMBLOK CHAIN
DMMWRT200S	FSWRITE	3-3	ERROR 'nnn' WRITING FILE 'fileid'

Figure 3-6 (Part 2 of 2). Interactive Problem Control System Messages

Chapter 4. The Format/Allocate Service Program

Introduction

The Format/Allocate service routine is a standalone program which:

- Formats all or part of a DASD device
- Allocates DASD space
- Creates volume labels for IBM 2314, 2319, 3330, 3340, 3350 series, 2305 series, 3375, 3380, and FB-512 direct access storage devices.

Operands entered from the IPL device and/or a 1052 console control the execution of the Format program.

With the inclusion of FB-512 devices, the format/allocate service program now supports two distinct types of DASD devices. It is important to understand the differences in these types.

The main difference is one of data format and addressing. One type, count-key-data, is referenced by a cylinder, head, and record number. A given record has two components; a count field and a data field. The count field contains the DASD address (cchhr) and length of the corresponding data. Formatting for CP's use means that these count and data fields are initialized to 4096-byte records (format writes 4096-byte records).

The other type, FB-512 devices, are addressed by a block number. The data is thought of as a linear address space of n blocks, numbered 0 through n-1. Each block is 512 bytes of data. Therefore, a CP page consists of eight consecutive blocks. Because the data is not stamped with a self-identifying label (such as cchhr in the count field of count-key-data devices), and the length of each block is fixed, the concept of formatting is quite different. Count-key-data space is formatted and allocated in units of cylinders. That means that the user "talks" to format/allocate by referring to specific cylinder numbers. FB-512 disk space is formatted and allocated in units of pages.

The distinction between count-key-data and FB-512 operation is detailed in the following pages.

If you install the speed matching buffer feature (Feature #6550) with the 3380, the extended count-key-data channel programs are used.

Format Operation

Count-Key-Data

The Format program writes 4096-byte (one page) records on all the specified cylinders. The records just written are then read to verify the disk surface. Any records not passing the read-after-write check are counted. When the format operation is complete, a summary of the addresses of the unusable pages is written on the console.

The first three records of cylinder 0 contain special system data including the volume label. If the format operation includes cylinder 0 any existing volume label is read first and if an OS Format 4 label is present, the information in the label concerning alternate track assignments is carried forward to the new label. Then the new volume label is written on the DASD device.

If cylinder 0 is not to be formatted, label checking is performed.

If unrecoverable DASD errors occur during the formatting operation, the format function is canceled, the message

DMKFMT735E FATAL DASD I/O ERROR

is issued, and the next control statement is read.

FB-512

The Format program writes zeros in the specified pages. The write is done with a read-back check to verify the disk surface. The format operation stops if any block fails the read-after-write check. The error message contains the block number in error.

The first two pages (pages 0 and 1) contain special system data including the volume label. If the format operation includes page 0, a volume label is written. If page 0 is not to be formatted, label checking is performed.

If fatal DASD errors occur during the formatting operation, the format function is canceled, the message

DMKFMT735E FATAL DASD I/O ERROR

is issued, and the next control statement is read.

Label-Only Operation

Count-Key-Data

In a label-only operation, a new volume is written on cylinder 0, track 0, record 3 of the DASD device. No label checking is done before the new label is written. The device must already be formatted before a label operation can be performed.

FB-512

In a label-only operation, a new volume label is written in the volume label block (block 1). A label-only operation can be done any time (the volume need not be formatted first).

Allocation Operation

In an allocation operation, disk space is assigned on the specified device in units of one cylinder for count-key-data or one page for FB-512. This disk space may be used as:

- Temporary space (TEMP)
- Permanent space (PERM)
- Directory space (DRCT)
- Temporary user space (TDSK)
- Paging space (PAGE)
- CP Dump space (DUMP)
- Override file space (OVRD)

The input parameters provide the information needed to update the allocation table. When the END allocation statement is processed:

- The allocation table is written. For count-key-data this is the byte allocation map on cylinder 0, track 0, record 4 of the DASD device. For FB-512 this is the allocation extent map in blocks 3 and 4.

- The results of the allocation operation are displayed at the console.

The DASD device must already be formatted before an allocation operation can be performed.

Executing the Format Program

The sequence for executing the Format program is:

1. Ready the DASD device.

2. Ready the reader. The reader must contain the Format/Allocate program and may also contain control cards for the program.

3. IPL the reader.

4. If a console is not located at either address 009 or 01F, signal attention from the console so the Format program can establish the address of the console.

5. The program title is printed.

6. When there are no control cards in the reader, the program requests control statements by sending prompting messages to the console.

7. When control cards are in the reader, they are processed. The prompter messages are displayed with the response field updated from the control statements already entered through the card reader. The program requests additional input, which can be entered via the reader or console.

8. The program issues messages indicating the start or end of an operation.

9. An operation in progress may be canceled by signaling attention from the console. Execution resumes with the next operation.

10. The Format/Allocate program cancels an operation if a an unrecoverable DASD I/O error occurs. A message indicating the cause of the error is displayed.

Method of Operation

This section describes the execution of the disk format program and shows the processing associated with:

- Formatting DASD space
- Allocating DASD space
- Writing a volume label.

Figure 4-1 shows the relationship of the diagrams.

Diagram 4-1 describes the major functions of the Format/Allocate program.

Diagram 4-2 describes the format function of the Format/Allocate program for count-key-data.

Diagram 4-3 describes the allocate function of the Format/Allocate program for count-key-data.

Diagram 4-4 describes the format function for FB-512.

Diagram 4-5 describes the allocate function for FB-512.

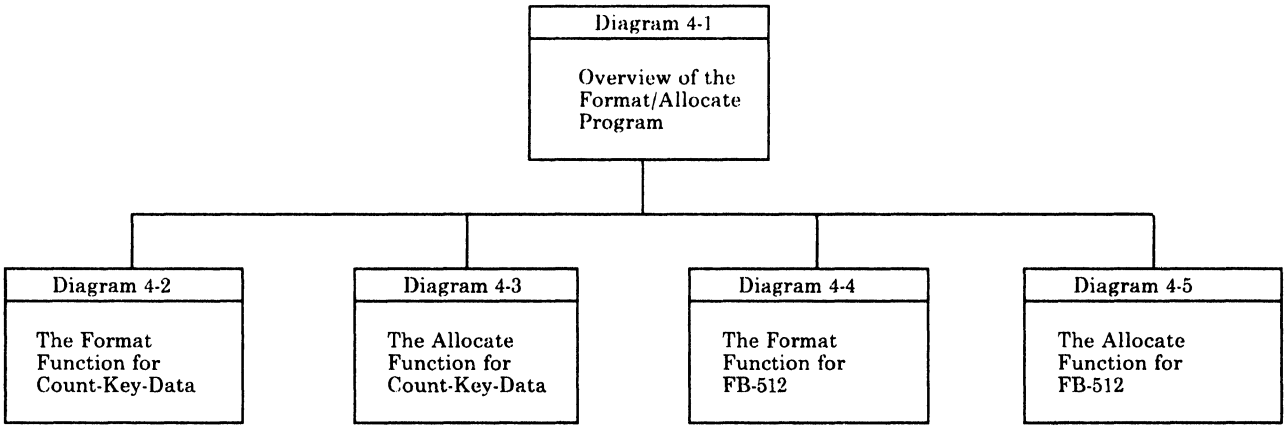
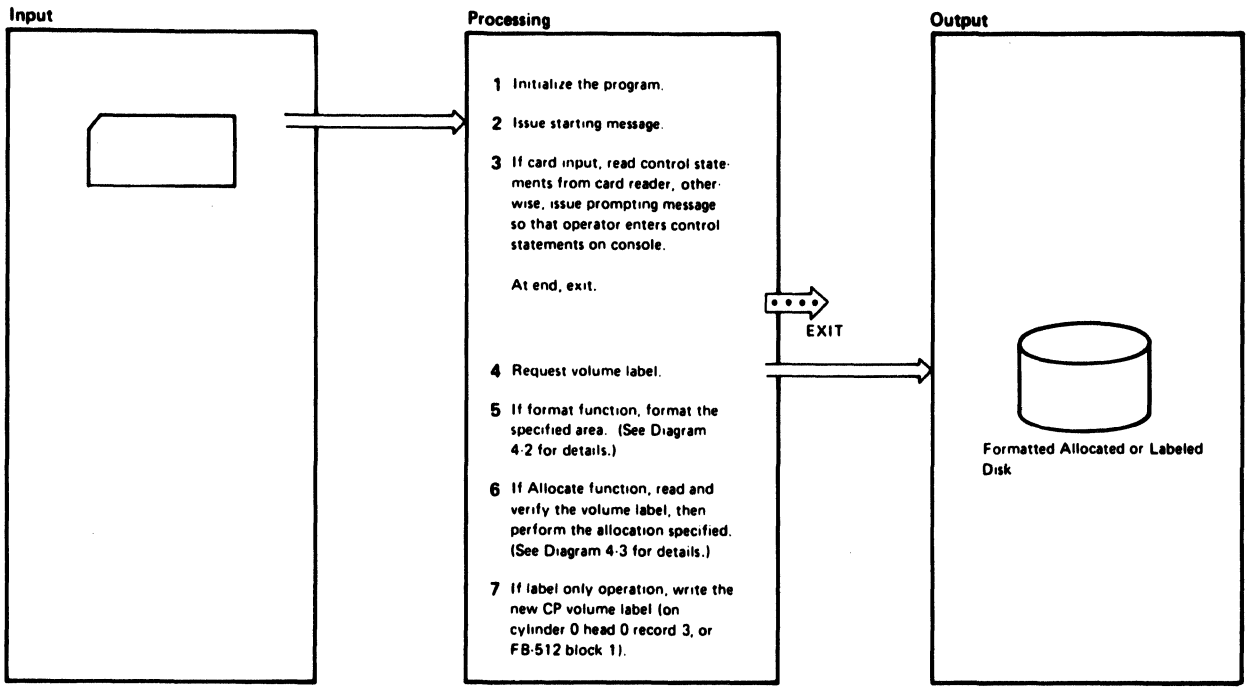


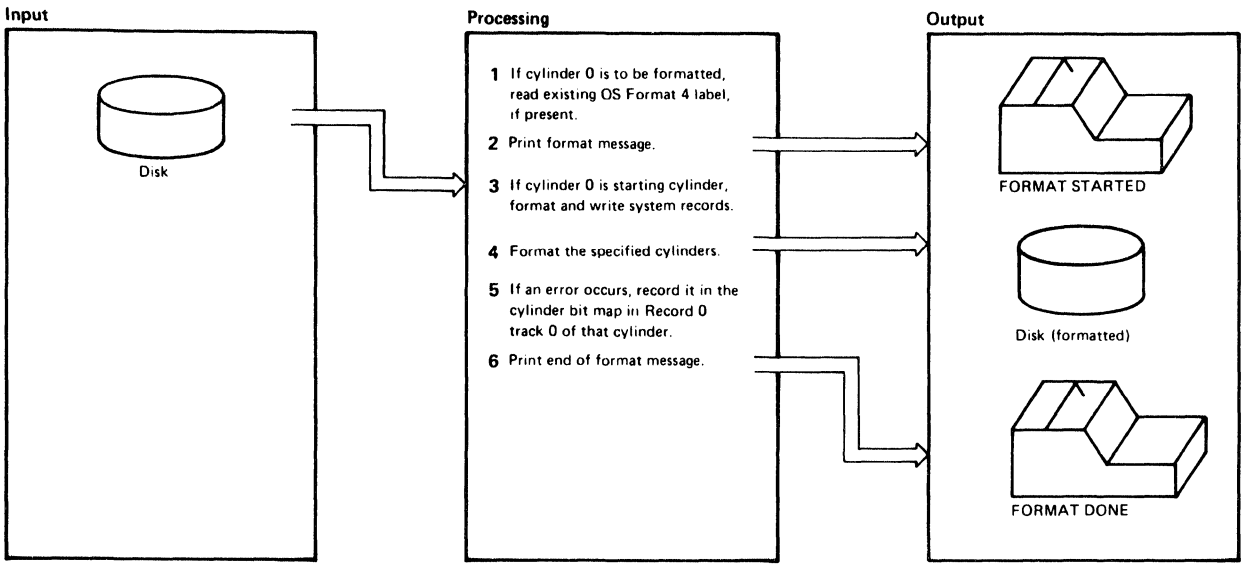
Figure 4-1. Key to the Format/Allocate Program Method of Operation Diagrams



Notes	Module	Label	Ref
1 DMKFMT sets up registers 15, 11, 13, 8, and 12 as base registers gets the IPL device address from the I/O old PSW, and stores it in IPLDEV. Next, DMKFMT locates the consoles by testing 009 and 01F. If neither of these devices is available, it enters the wait state until an attention interruption is received from the console.	DMKFMT	DMKFMT	
2 The program title VM/370 FORMAT/ALLOCATE PROGRAM is displayed at the console.	DMKFMT	STMSG	
3 If the switch (CDSW2) contains 'FF', the reader enters the wait state until an I/O interrupt occurs. The CONSINT routine reads the control statements and the VALIDATE routine checks that they are valid.	DMKFMT	GETCARD CONSINT VALIDATE	
The prompter messages are issued. If the control statements are entered through the card reader, the prompter messages include the response that was already specified in cards.		SELECT	
The message ENTER FORMAT OR ALLOCATE prompts the operator. If the operator correctly enters FORMAT (F) or ALLOCATE (A), one of the following messages FORMAT FUNCTION SELECTED ALLOCATE FUNCTION SELECTED appears on the console. Otherwise, the prompter message is reissued. Then, the message ENTER DEVICE ADDRESS (cuu): prompts the entering of the device address.		DEVICEAD DEVICEAD	

Notes	Module	Label	Ref
If the device address entered is valid, the device type is requested. ENTER DEVICE TYPE: For count-key data, the high cylinder address, highest record, and device type are initialized depending on the device type entered. For FB-512 devices, a "read device characteristics" CCW is performed and the highest block number is determined. From this, the highest page number is calculated.		DEVTYPE	
If the device address entered is not available, the error message DMKFMT730E DEV xxx NOT OPERATIONAL is issued and the request for a device is repeated.			
4 The message ENTER DEVICE LABEL: is displayed.	DMKFMT	LAB	
5 If the function being performed by the Format/Allocate program is the format operation, then, if cylinder 0 or page 0 is to be formatted, DMKFMT branches to FMT, otherwise, it branches to REGFORM1.	DMKFMT	LAB	
6 The volume label is read and verified by the LBLREC or FBALABRD CCW string, then DMKFMT branches to the ALLOCATE routine.	DMKFMT	LAB	
7 The CP volume label is written by the LABWRITE or FBAELIOW CCW string. Processing continues by reading the next control statement (see Step 3).	DMKFMT	LABONLY	

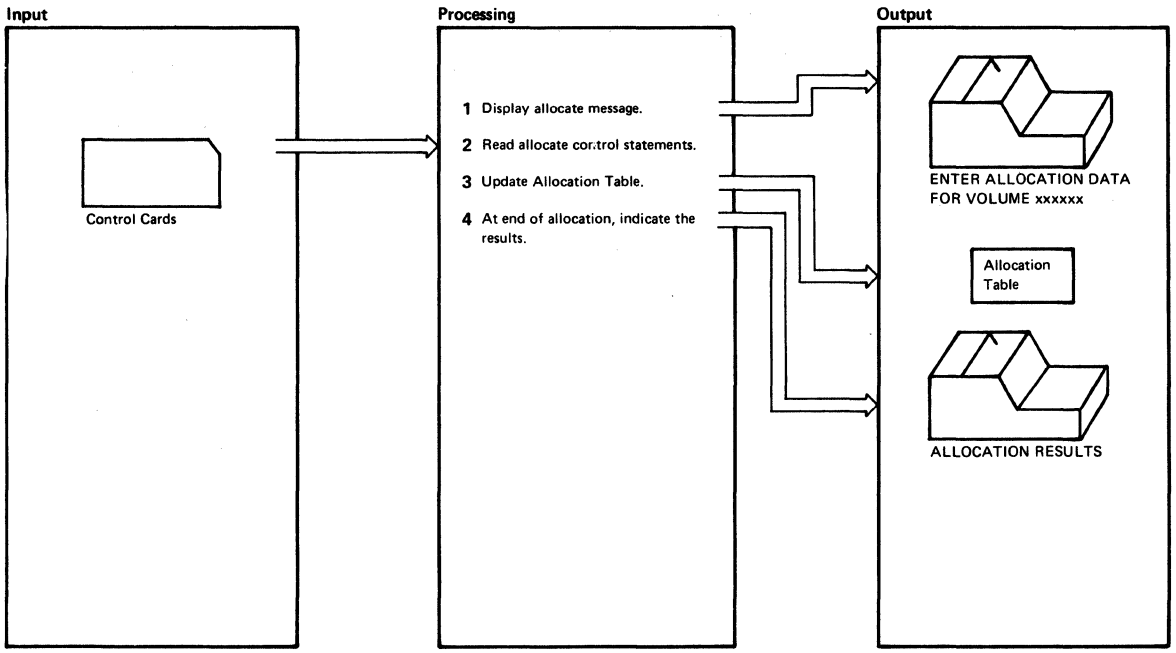
Diagram 4-1. Overview of the Format/Allocate Program



Notes	Module	Label	Ref
1 If cylinder 0 is to be formatted, any existing OS Format 4 label is read to preserve (for IBCDASDI) the CCHH address of the next unassigned alternate track and also the count of the remaining unassigned alternates. This data will be put in the new OS Format 4 label on track 0.	DMKFMT	FMT	
2 DMKFMT branches and links to the message writing (WMSG) routine to display FORMAT STARTED Then it updates the I/O new PSW so that the IOINT routine executes when an I/O interrupt occurs.	DMKFMT	REGFORM1	
3 If cylinder 0 is the starting cylinder, the FORMAT program formats cylinder 0 by setting up the CCWs appropriate to the device type and then branching to the STIO routine to perform the I/O operation. Once cylinder 0 is formatted, system records are written on it. Then branch to the CHECK0 routine is set to NOP so that CHECK0 is executed only once. The records written on cylinder 0 are Record – Description 0 – Page bit map 1 – IPL record 2 – Checkpoint record 3 – Vol1 label 4 – Allocation bit map 5 – Format 4 label 6 – Format 5 label F3 – Page size filler F4 – Filler record for 2314/2319	DMKFMT	STORE CHECK0	

Notes	Module	Label	Ref
4 The appropriate device type CCWs are set up by the Format program. Page size records are written and verified by the STIO routine. Control returns to the RESUMP routine if no error occurs. The RESUMP routine updates the record numbers and the STIO routine again writes and verifies the record. This loop continues until the last cylinder specified is completely formatted.	DMKFMT	STORE STIO RESUMP	
5 If an error occurs in the STIO routine, control is transferred to the IOINT routine. The error is retried up to 9 times before the message DMKFMT736E IO ERROR xxx CCHHR= SENSE= is displayed. The Page bit map is updated to indicate a bad surface. The errors that cause the Format function to terminate are: seek error error in writing or reading the home address error writing or reading record 0 error setting file mask error in reading count-key-data The message DMKFMT735E FATAL DASD I/O ERROR is displayed and control returns to the GETCARD routine.	DMKFMT	IOINT READER06	
6 DMKFMT displays the message FORMAT DONE to indicate that the specified cylinders are formatted, and then summarizes the errors with the message xxxPAGE RECORDS FLAGGED	DMKFMT	CLEANUP	

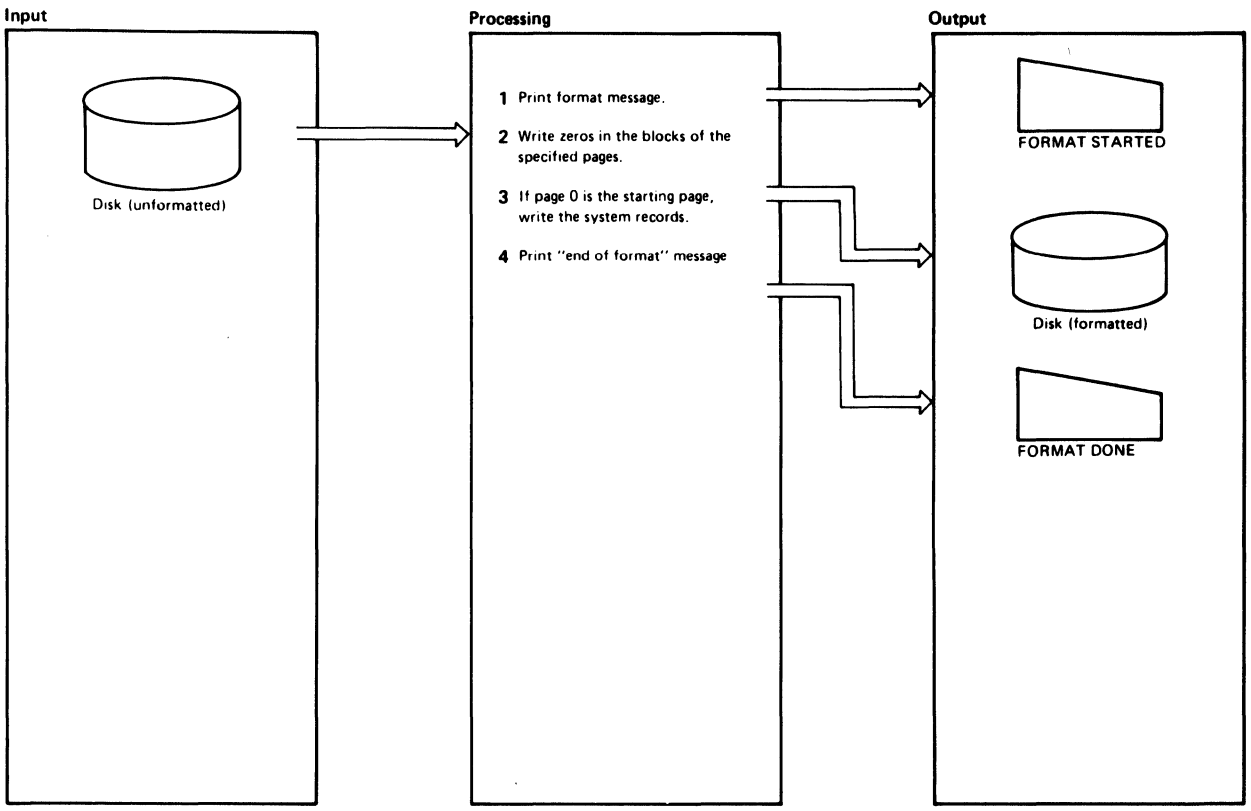
Diagram 4-2. The Format Function for Count-Key-Data



Notes	Module	Label	Ref
1 The messages ENTER ALLOCATION DATA FOR VOLUME xxxxxx type cyl cyl . . . . . . are displayed.	DMKFMT	ALLOCATE	
2 If the Allocate control statements are entered via a card reader, the switch (CDSW2) contains X'FF' Control is transferred to the GETCARD routine which reads the cards. The CONSINT and VALIDATE routines verify the control statements and allocate processing resumes at the label REREAD. There is a branch and link to the RMSG routine to read from the console. The console read is not performed in this case because CDSW2 is X'FF'.	DMKFMT	GETCARD CONSINT VALIDATE REREAD	
If the allocate control statements are entered via the console, the switch (CDSW2) contains X'00'. The control statements are read from the console by branching and linking to the RMSG routine.		REREAD RMSG	
3 The address of the cylinder byte map is loaded into register 9. The total number of cylinders specified is loaded into register 8. The cylinder byte map is updated for each of the specified cylinders according to the type indicated in the control statement.	DMKFMT	AOKALL INDIC	

Notes	Module	Label	Ref
Control Statement / Indication in Cylinder Byte Map TEMP X'00' PERM X'01' TDSK X'02' DRCT X'04' PAGE X'08' DUMP X'10' OVRD X'14' The map is printed after the END statement is processed.			
4 The message ALLOCATION RESULTS followed by the type corresponding to the allocated cylinders is displayed. Finally, the message DEVICE xxx VOLUME xxxxxx ALLOCATION ENDED is displayed	DMKFMT	FINI	

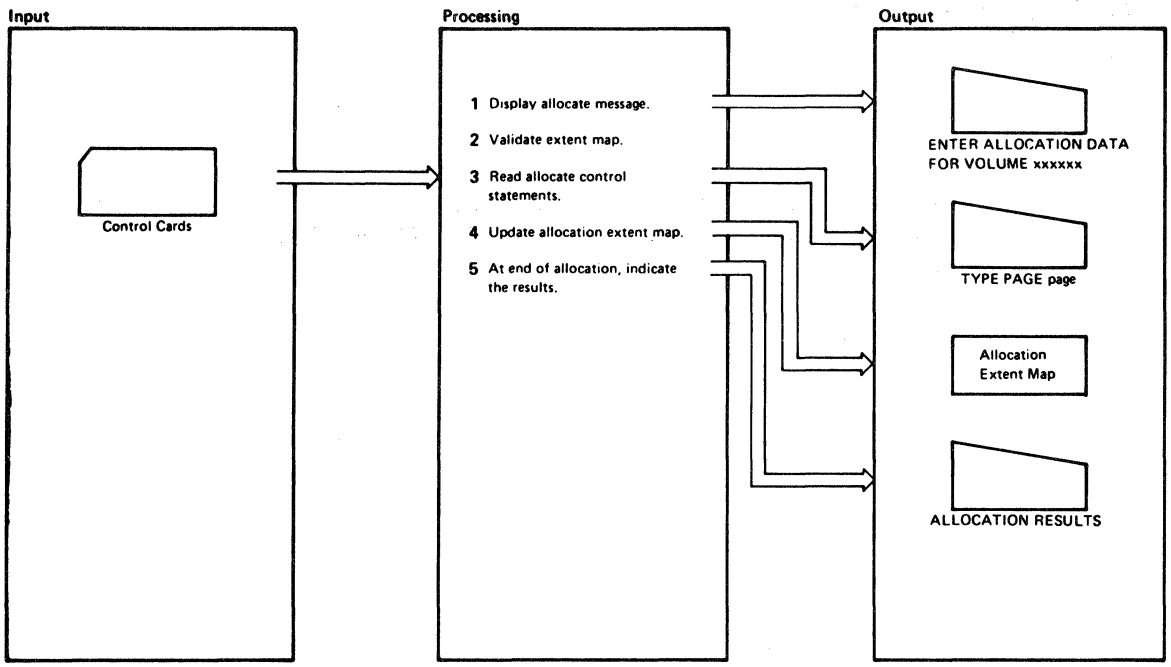
Diagram 4-3. The Allocate Function for Count-Key-Data



Notes	Module	Label	Ref
1 DMKFMT branches and links to the message writing (WMSG) routine to display FORMAT STARTED Then it updates the I/O PSW so that the IOINT routine executes when an I/O interrupt occurs.	DMKFMT DMKFMT	REGFORM1 FORMFBA	
2 Define extent data is initialized to map the entire volume. The FORMCCW string is used to write zeros in the blocks of the specified pages. Each call to the STIO routine writes a track's worth of blocks. The RESUMFBA routine increases to the next block number and the STIO routine again writes and verifies the blocks. This loop continues until the last block of the last page has been written.	DMKFMT	FORMFBA NEXTCAG FORMIO RESUMFBA	
3 If page 0 was specified as the starting page, write the system data in the first 16 blocks. The blocks written are: Block – Description 0 – IPL record 1 – Vol1 label 2 – VTOC 3-4 – Allocation map 5-12 – DMKCKP program 13-15 – Reserved – all zeros	DMFFMT	FORMEND	

Notes	Module	Label	Ref
4 DMKFMT displays the message FORMAT DONE to indicate that the specified cylinders are formatted, and then summarizes the errors with the message xxxPAGE RECORDS FLAGGED	DMKFMT	CLEANUP	

Diagram 4-4. The Format Function for FB-512



Notes	Module	Label	Ref
1 Display the message ENTER ALLOCATION DATA FOR VOLUME xxxxxx	DMKFMT	ALLOCATE	
2 Validate the extent map. If it is empty, go to Step 3. If there is data already in the map, verify that the entries are numerically ascending and that a X'FF' marks the end of the map. If the validity test fails, set a flag (BADFLAG) and go to Step 3.	DMKFMT	VALOOP	
3 Display the message TYPE PAGE page If the allocate control statements are entered via a card reader, the switch (CDSW2) contains X'FF'. Control is transferred to the GETCARD routine, which reads the cards. The CONSINT and VALIDATE routines verify the control statements and allocate processing resumes at the label REREAD. There is a branch and link to the RMSG routine to read from the console. The console read is not performed in this case because CDSW2 is X'FF'.	DMKFMT	INITMAP CONSINT VALIDATE REREAD	
If the allocate control statements are entered via the console, the switch (CDSW2) contains X'00'. The control statements are read from the console by branching and linking to the RMSG routine.		REREAD RMSG	

Notes	Module	Label	Ref
4 Each input is checked for validity. A dummy entry is created for the extent map and is used to scan the map looking for the correct insertion point. If the new entry overlays existing entries in any manner, the extent map is reconstructed, to fit in the new entry. The 'type' byte of the 12-byte extent map is set based on the input control statement: Control Statement / Type Byte TEMP X'00' PERM X'01' TDSK X'02' DRCT X'04' PAGE X'08' DUMP X'10' OVRD X'14' The results of the allocation are printed after the END statement is processed.	DMKFMT	ALOCFBA	
5 The message ALLOCATION RESULTS is displayed following by the type and corresponding start and end page numbers. Then the message DEVICE xxx VOLUME xxxxxx ALLOCATION ENDED is displayed.	DMKFMT	PRINTALL	

Diagram 4-5. The Allocate Function for FB-512

Program Organization

DMKFMT

A standalone program that formats, allocates, and labels all (or part) of 2314, 2319, 3330, 3340, 3350 series, 3375, 3380, FB-512 series, and 2305 series direct access storage devices.

Entry Point

DMKFMT

Routines Called

None

Register Usage

R0-7:	Scratch
R8:	5th base register
R9-10:	Scratch
R11:	3rd base register
R12:	2nd base register
R13:	4th base register
R14:	Scratch
R14:	Linkage register
R15:	1st base register

Directory

Figure 4-2 is an alphabetical list of the major labels in the Format/Allocate program. The associated method of operation diagram and a brief description of the function performed at the point in the program indicated by each label are included in the list.

Label	Diagram	Description
ALLOCATE	4-3	Performs the allocate function of the Format program (count-key-data).
ALOCFBA	4-5	Performs the allocate function of the format program (FB-512).
ALTTRACK		Performs alternate track recovery for 3340/3344.
AOKALL	4-3	Locates the cylinder byte map.
CHECK0	4-2	Writes system records on cylinder 0.
CLEANUP	4-2	Summarizes the errors encountered while formatting the disk.
CONSINT	4-1 4-3	Processes console interrupts.
DEVICEAD	4-1	Displays the prompter message requesting the device address.
DEVTYPE	4-1	Displays the prompter message requesting the device type.
DMKFMT	4-1	Initializes the Format program.
ERRECOV		Performs DASD error recovery.
FATAL		Displays the termination message and reads the next control statement.
FINI	4-3	Displays the cylinders just allocated with the type of allocation.
FMT	4-2	Initializes cylinder 0 for formatting by first reading any existing OS Format 4 label.
FORMAL		Displays the starting cylinder or label message.
FORMFBA	4-4	The main FB-512 formatting routine.
GETCARD	4-1 4-3	The main control routine. It reads control statements from the reader or transfers control to the SELECT routine to issue prompter messages.
GRAPHID		Handles input and output operations for display terminals.
INDIC	4-3	Updates the cylinder byte map to reflect the type of allocation for each cylinder.
IOINT	4-2	Handles I/O interrupts and retries errors.
LAB	4-1	Displays the prompter message requesting the device label.
LABELIOR		Reads and verifies the volume label.
LABONLY	4-1	Rewrites the volume label (record 3) and nothing else.
MCRTN		Processes machine checks.
NEXT		Displays end of cylinder message.
PRINTALL		Displays the allocation table on the terminal.

Figure 4-2 (Part 1 of 2). The Format/Allocate Program Label Directory

Label	Diagram	Description
READER06	4-2	Updates the page bit map to indicate a bad surface.
REGFORM1	4-2	Initializes the format function when cylinder 0 is not included.
REREAD	4-3	Reads control statements from the console for the allocate function.
RESUMFBA	4-4	Updates the block number during the format operation (FB-512).
RESUMP	4-2	Updates the record number during the format operation (count-data-key).
RMSG	4-3	Reads from the typewriter terminals.
SELECT	4-1	Prompts the operator to enter the appropriate control statement.
SENSIT		Gets sense information.
SENSIT2		Displays the sense information.
STIO	4-2	Writes and verifies page size records during format operation.
STMSG	4-1	Displays the program title.
STORE	4-2	Sets up CCW string to format cylinder 0.
VALIDATE	4-1 4-3	Checks control statements entered through a card reader for accuracy.
WMSG		Displays messages on the terminal.
XBIN		Converts hexadecimal numbers to binary.

Figure 4-2 (Part 2 of 2). The Format/Allocate Program Label Directory

Data Areas

This section contains descriptions of the DASD record formats and the layout of these DASD records for:

- 2305 Models 1 and 2
- 2314/2319 devices
- 3330 series
- 3340 series
- 3350 series*
- FB-512 series
- 3380

Record 0

Record 0 is the standard 8-byte data area, set to 0 except for cylinder 0. On cylinder 0, pages are flagged as follows:

For 2305, 2314/2319, 3330, 3340, 3380, and 3350 in compatibility mode:

E0	00	00	00	00	00	00	00

For 3350 in native mode only:

F0	00	00	00	00	00	00	00

Figure 4-3. Record 0 Format

Record 1

24 bytes - track 0 cylinder 0

IPL record - puts system into wait state if storage device is loaded (via IPL function).

```
00 02 00 00 00 00 00 00 03 00 00 00 20 00 00 00 00 00 00 00 00 00 00 00
```

Figure 4-4. Record 1 Format

* When a 3350 DASD is formatted for the 3880 Model 21 Storage Subsystem, it can be used for that subsystem only, and only for paging and swapping. It must be reformatted as a normal 3350 before it can be used as such.

Record 2

4096 bytes - track 0 cylinder 0

Checkpoint record - this is the Checkpoint program load at IPL time to retrieve and save control information for a warm start.

Figure 4-5. Record 2 Format

Record 3 consists of a 4-byte key and an 80-byte record.

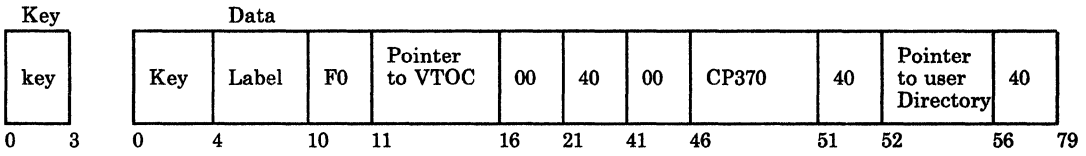


Figure 4-6. Record 3 Format

Record 4

1024 bytes - track 0 cylinder 0

Record 4 is an allocation byte map used to identify cylinder usage. Each byte corresponds to one cylinder; the value of the byte indicates the type of usage for the cylinder.

Value	Usage
00	temporary
01	permanent
02	T-disk
04	directory
08	paging
10	CP dump
14	OVRD (override)

Example

```
00 00 01 00 04 02 00 . . . . FF 00 . . . . . . 00
```

Cylinder beyond the last cylinder that can be allocated. This point varies depending on the device.

Figure 4-7. Record 4 Format

Record 5

44 bytes key - track 0 cylinder 0
96 bytes data area

Format 4 DSCB type label - used to be compatible with other systems.

04---04	FORMAT 4 LABEL
44-byte Key	96-byte Data Area

Figure 4-8. Record 5 Format

Record 6

44 bytes key - track 0 cylinder 0
96 bytes data area

Format 5 DSCB type label - used to be compatible with other systems.

05 05 05 05 00	FORMAT 5 LABEL
44 Byte Key	96 Byte Data Area

Figure 4-9. Record 6 Format

Record F3

4096 bytes - 1 page, track 0 or track 1 (2314/2319 devices only).

F3 Record is reserved for IBM use, and is referred to as a filler record.

Figure 4-10. Record F3

Record F4

1624 bytes - track 1 (2314/2319 devices only).

Record F4 is used only on 2314 and 2319 to align Record 4 on the track.

Figure 4-11. Record F4

Record 4

824 bytes - cylinder 0, track 1 (2314/2319 devices only).
The first segment of Record 4 is used for paging.

Figure 4-12. Record 4

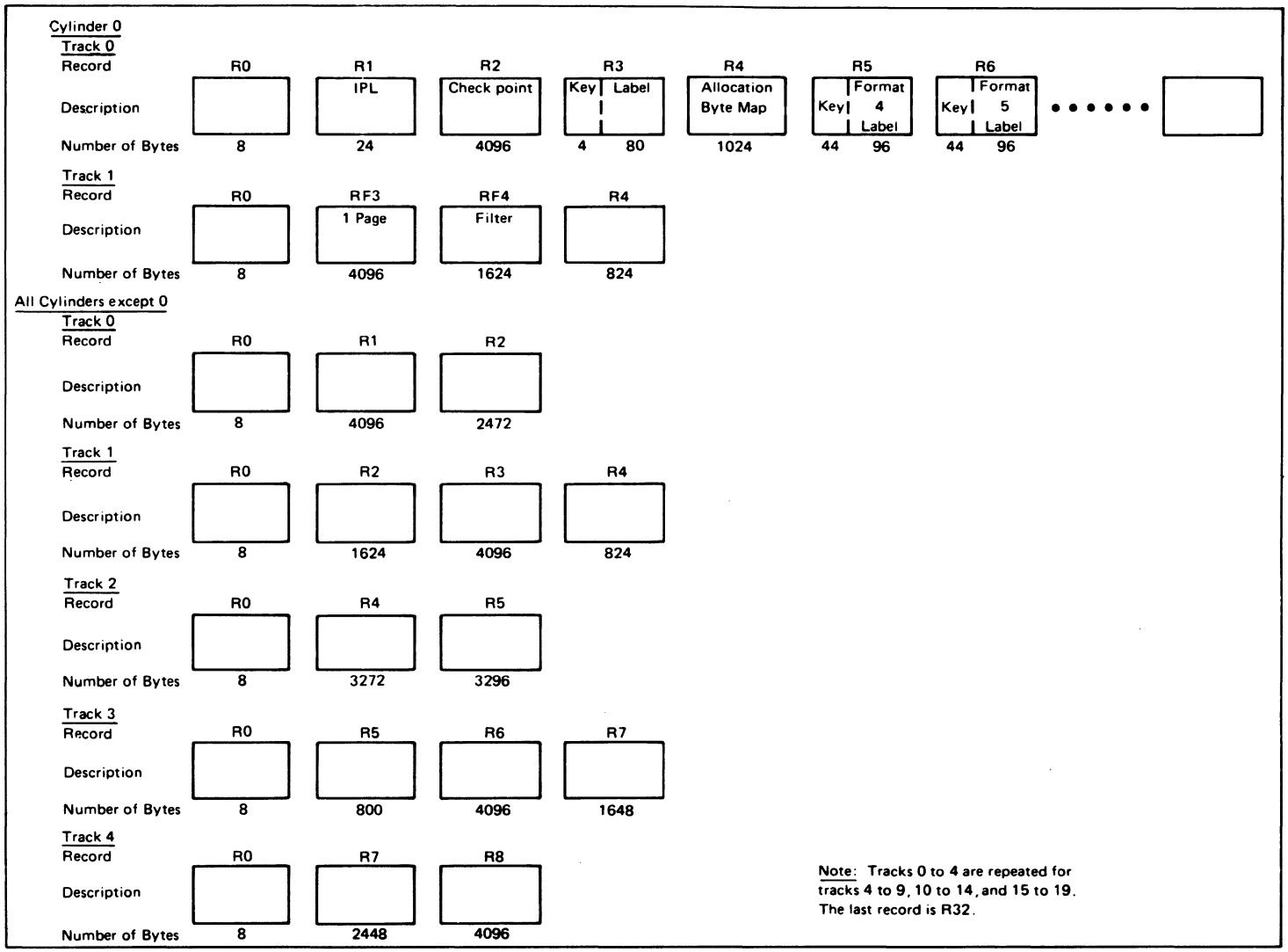


Figure 4-13. 2314/2319 Record Layout

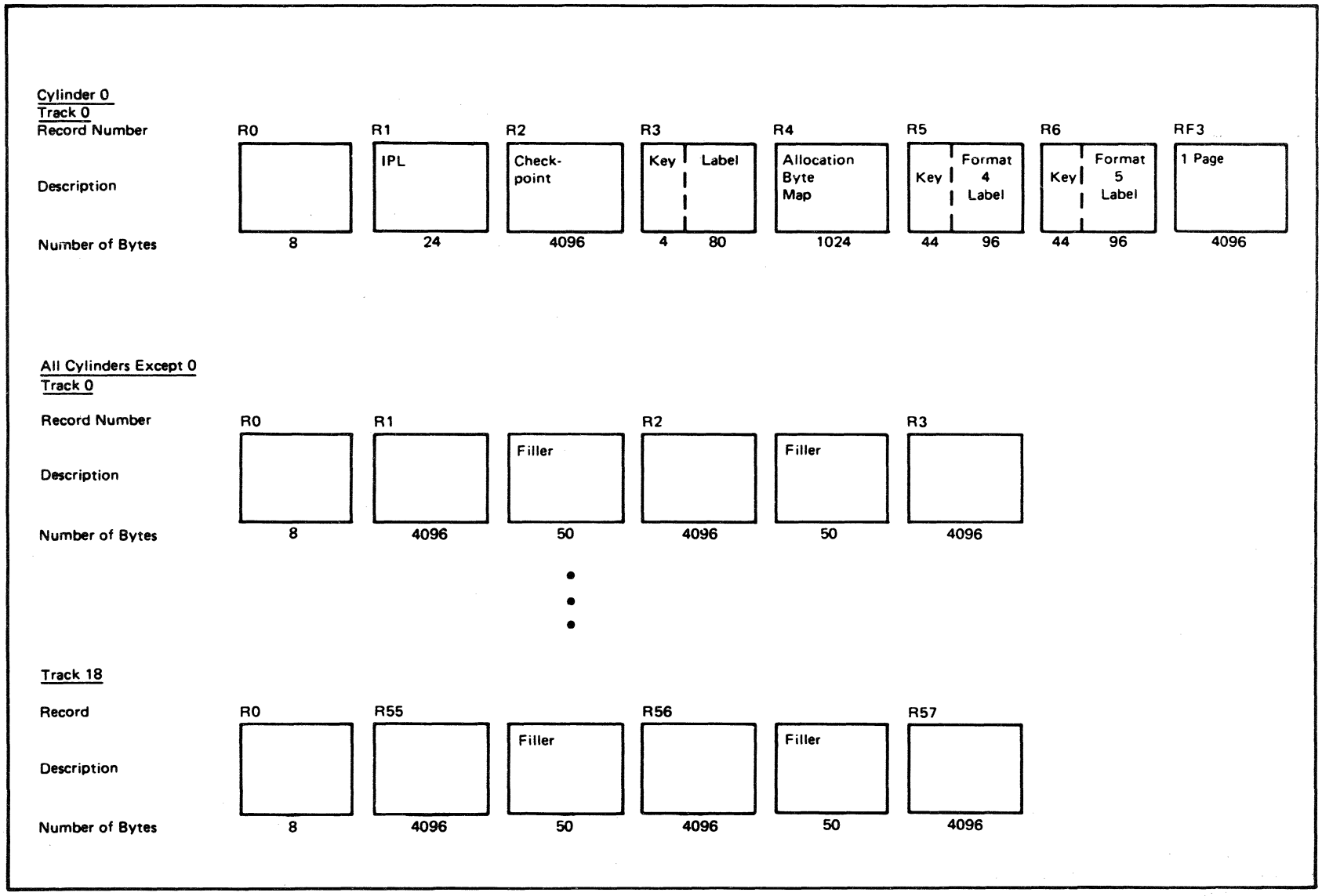


Figure 4-14. 3330 Series Record Layout

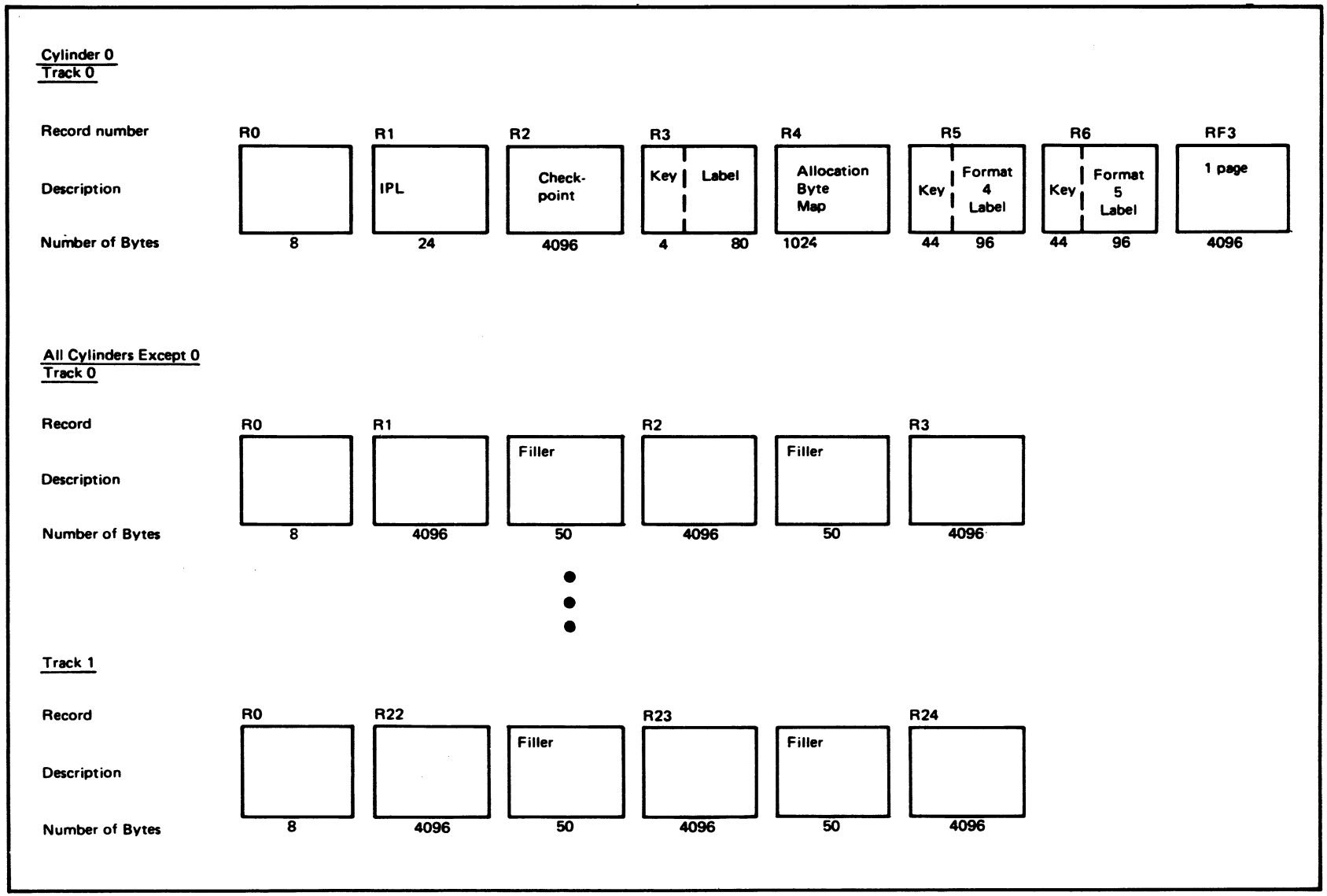


Figure 4-15. 2305 Models 1 and 2 Record Layout

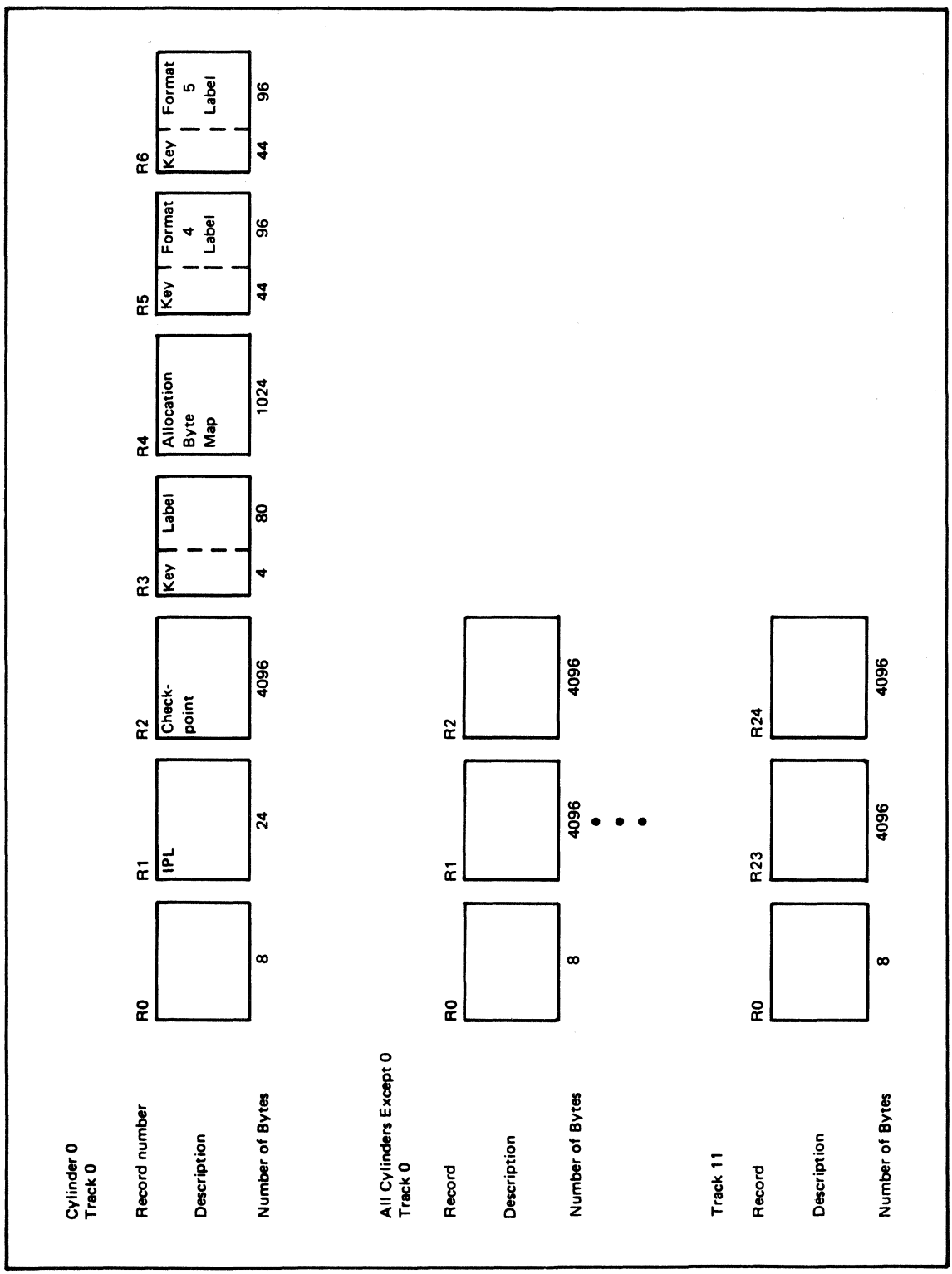
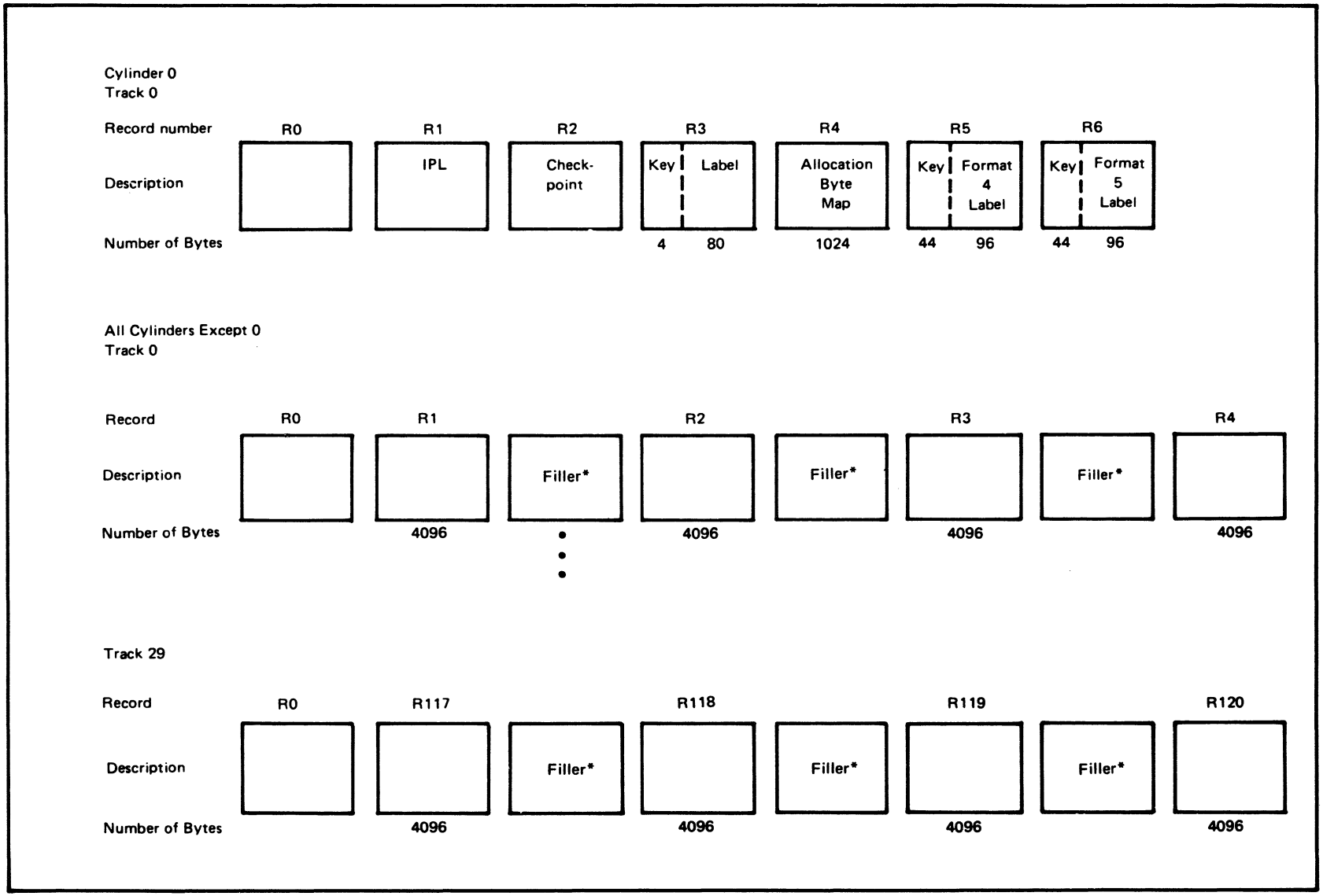


Figure 4-16. 3340 Record Layout



*Not included when formatted for a 3880 Model 21 Storage Subsystem.

Figure 4-17. 3350 Record Layout

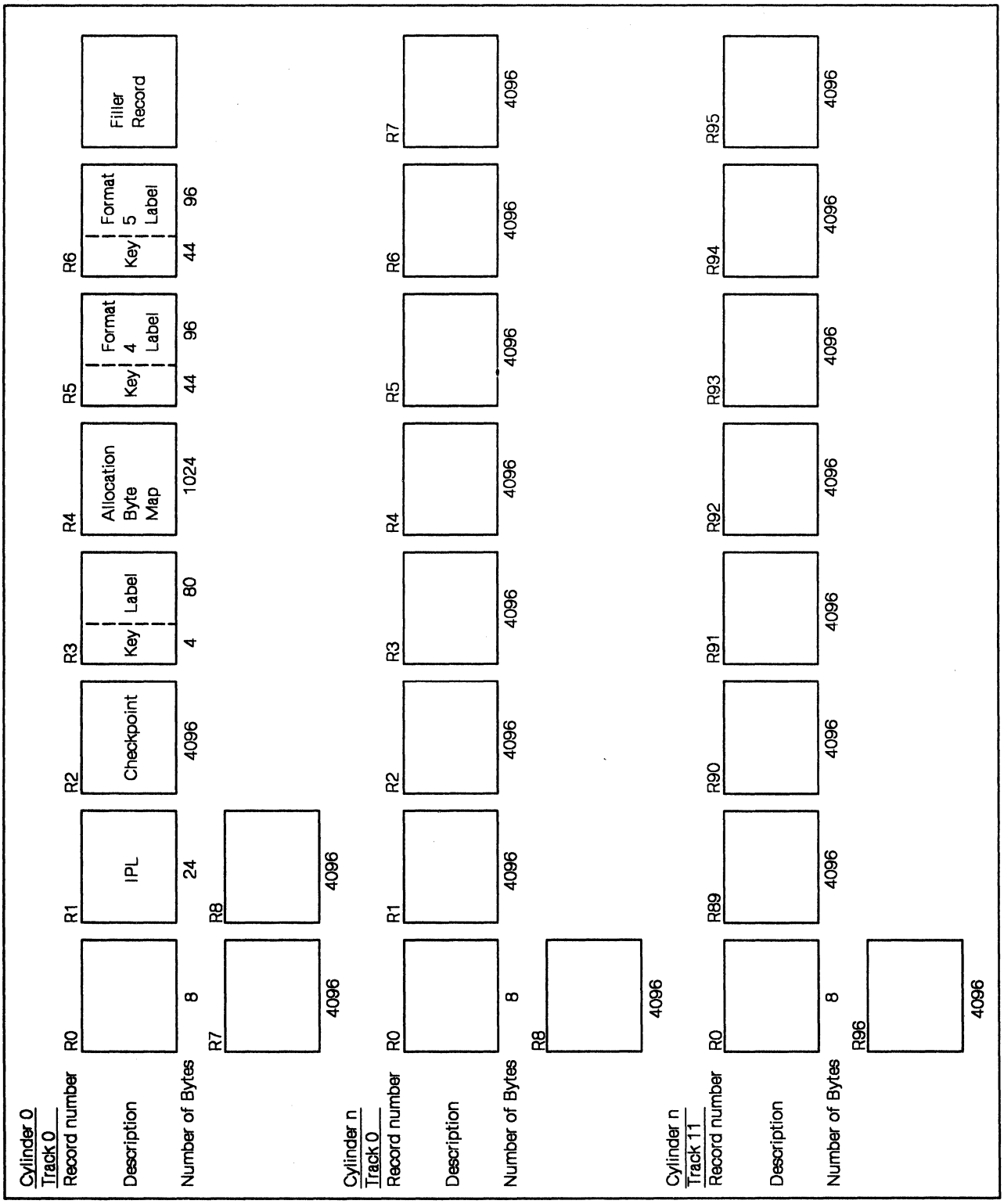


Figure 4-18. 3375 Record Layout

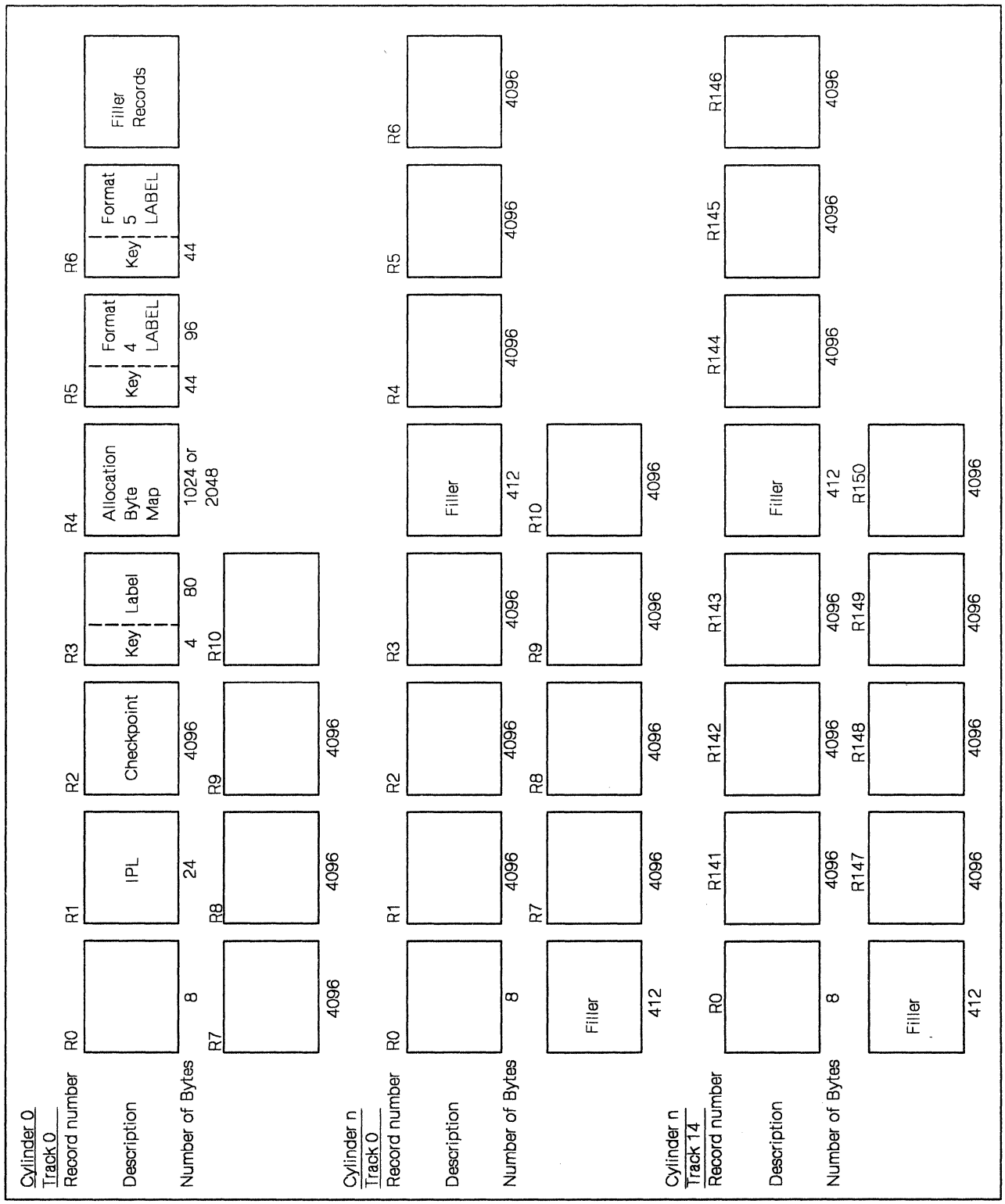


Figure 4-19. 3380 Record Layout

FB-512 Data Layout and Content (each block is 512 bytes)

	IPL	VOL1	VTOC	Allocation Extent Map	DMKCKP	Zeros
Block	0	1	2	3-4	5-12	13-15

Blocks 16 to the end of the volume contain CP pages.

Block 0

24 bytes used - remainder of block contains zeros.

IPL record - puts system into wait state if device is loaded (via IPL function).

```
00 02 00 00 00 00 00 00 03 00 00 00 20 00 00 00 00 00 00 00 00 00 00 00
```

Figure 4-20. Block 0 Format

Block 1

80 byte volume label - remainder of block contains zeros.

```
VOL1 | label | F0 | VTOC Pointer | 00 | CI Size | BLK/CI | LAB/CI | 4040
0      4       10   11             16   21        25       29       33

0000 | CP370 | 40 | Pointer to Directory | 404040
41     46      51   52                     56          79
```

The VTOC pointer (bytes 11-16) contains	00	00	00	00	02
The CI size (bytes 21-25) contains		00	00	02	00
The blocks per CI (bytes 25-29) contain		00	00	00	01
The labels per CI (bytes 29-33) contain		00	00	00	03

Figure 4-21. Block 1 Format

Block 2

The Volume Table of Contents - contains a Format-4 and Format-5 DSCB.

0404	F4	0000	01	00000000	VTOC Start	VTOC End	00000000000000	0505	F5	00000000
0	44	45	59	61	107	111	115	139	140	184	186	228

The remainder of the block contains zeros

The VTOC start address (bytes 107-110) contains 00000002

The VTOC end address (bytes 111-114) contains 00000003

Figure 4-22. Block 2 Format

Block 3-4

Allocation Extent Map - used to identify the usage of pages on the device. The 1024-byte map is made up of 12-byte entries. Each entry represents an extent of pages, and the usage of that extent.

Type	Unused	Start	End	Type		Start	End	....
0	4	8	12	16	20	24	1023

The Type field can be:

00 - TEMP
01 - PERM
02 - TDSK
04 - DRCT
08 - PAGE
10 - DUMP
14 - OVRD
FF - Marks the end of the table

Start and end are page numbers denoting the range. The entries are sorted from low to high.

Figure 4-23. Block 3-4 Format

Block 5-12

4096 bytes used.

Checkpoint record - this is the first page of the checkpoint program (DMKCKP) that gets control at IPL to retrieve and save control information for a warm start.

Figure 4-24. Block 5-12 Format

Block 13-15

Reserved for system use - contains zeros.

Figure 4-25. Block 13-15 Format

Diagnostic Aids

Figure 4-26 is a list of the messages issued by the Format/Allocate program. The label of the message, the label of the routine issuing the message and the associated method of operation diagram are included in the list.

Message Code	Label of Message	Issuing Routine	Diagram	Message Text
DMKFMT536I	DRCTFAIL	CSWMSS		raddr devname REPORTS DISABLED INTERFACE; FAULT CODE = cccc; NOTIFY CE
DMKFMT730E	WR1	DEVTYPE	4-1	DEV raddr NOT OPERATIONAL OR NOT READY
DMKFMT732E	MCMSG	MCRTN		MACHINE CHECK
DMKFMT733E	WRONG	LABELBAD		VOLID READ IS volid1 NOT volid2
DMKFMT734E	TYPERR	VALIDATE		TYPE OR {CYL|PAGE} INVALID
DMKFMT735E	FATLMSG	FATAL	4-2	FATAL DASD I/O ERROR. CSW = csw
DMKFMT736E	IOERR	DEVICEAD	4-2	I/O ERROR raddr {CCHRHR=cchr|BLOCK=nnnnnn}
		SENSIT2		SENSE = sense
DMKFMT737E	BAD	BADINPUT		INVALID OPERAND
DMKFMT738A	IPLERROR	DEVICEAD		DEV raddr INTERVENTION REQUIRED
DMKFMT739E	MSGATRK	ALTTRACK		FLAGGED PRIMARY TRACK HAS NO ALTERNATE ASSIGNED; IO ERROR FOLLOWS
DMKFMT740E	MSG35MB	DEVTYPE	4-1	PACK MOUNTED IS 3340-35, NOT 3340-70. MOUNT ANOTHER OR RESPECIFY
DMKFMT741E	WRDEV1			DEVICE raddr IS zzzz NOT xxxx-nn AS SPECIFIED. RESPECIFY OR NOTIFY SYSTEM SUPPORT.
DMKFMT742E	MSG742			ALLOCATION FUNCTION IS NOT ALLOWED- FORMAT OF VOLUME IS A PREREQUISITE
DMKFMT756E	PCMSG	PRCHK		PROGRAM CHECK PSW = psw
	TITLE	STMSG	4-1	VM/370 FORMAT/ALLOCATE PROGRAM RELEASE n
	FORA	SELECT	4-1	ENTER FORMAT OR ALLOCATE:
	FMTMSG	SELECT	4-1	FORMAT FUNCTION SELECTED
	ALLOCMSG	SELECT	4-1	ALLOCATE FUNCTION SELECTED
	ADDRESS	DEVICEAD	4-1	ENTER DEVICE ADDRESS (CCU):
	TYPMSG	DEVTYPE	4-1	ENTER DEVICE TYPE:
	DATAMSG	ALLOCATE	4-3	ENTER ALLOCATION DATA FOR VOLUME xxxxxx

Figure 4-26 (Part 1 of 2). The Format/Allocate Program Messages

Message Code	Label of Message	Issuing Routine	Diagram	Message Text
	ALMSG	ALLOCATE	4-3	TYPE CYL CYL
	ALMSG1	ALLOCATE	4-3	....
	ALLEND	FINI	4-3	DEVICE xxx VOLUME xxxxxx ALLOCATION ENDED
	STCYL	FORMALL		ENTER START CYLINDER (xxx) OR "LABEL":
	ENDCYL	NEXT		ENTER END CYLINDER (xxx):
	PROGFOR	REGFORM	4-2	FORMAT STARTED
	RDLAB	LAB	4-1	ENTER DEVICE LABEL:
	ENDFOR	CLEANUP	4-2	FORMAT DONE
	PAGE	CLEANUP	4-2	xxx PAGE RECORDS FLAGGED
	RESULTS	FINI	4-3	ALLOCATION RESULTS
	MAP	PRINTALL		TEMP 000 000
	LABELCHK	LABONLY		LABEL IS NOW xxxxxx
	STPAGE	STRTPAG	4-1	ENTER START NUMBER OR "LABEL":
	ENPAGE	ENDPAG	4-1	ENTER END PAGE NUMBER:
	ALPMSG	INITMAP	4-4	TYPE PAGE page
	MAPFULL	COMPRESS	4-4	NUMBER OF EXTENTS EXCEEDS MAXIMUM - RESPECIFY
	PAGEXC	ALOCFBA	4-5	HIGHEST ALLOCATABLE PAGE IS xxxxxx - RESPECIFY
	PAG2LO	ALOCFBA	4-5	LOWEST ALLOCATABLE PAGE IS PAGE 2 - RESPECIFY
	PAGERR	ALOCFBA	4-5	PAGE NUMBER INVALID - RESPECIFY
	FBAMAP	FBAPRALL	4-5	TYPE xxxxxx xxxxxx

Figure 4-26 (Part 2 of 2). The Format/Allocate Program Messages

Chapter 5. DMKDIR—The Directory Program

Introduction

The DMKDIR program builds the directory on a volume previously formatted by the Format/Allocate program, using space that was previously allocated for use as directory space.

Under the control of CMS or standalone, the new directory is dynamically swapped and placed in use provided the directory has been created without errors, on a volume in the system-owned list, and provided the user class is A, B, or C.

The new directory can be built so that it does not overlay an existing directory. To do this, allocate enough space for two directories or allocate space for a new directory each time the directory is created.

The directory program can be run standalone or under the control of CMS. The CMS DIRECT command invokes the directory program under CMS.

Method of Operation

This section describes the operation of the Directory program. Figure 5-1 shows the relationship of the Method of Operation diagrams.

Diagram 5-1 describes the major functions of the Directory program.

Diagrams 5-2, 5-3, and 5-4 describe the control statement processing and the resulting action.

Diagram 5-5 shows the functions performed before the program terminates.

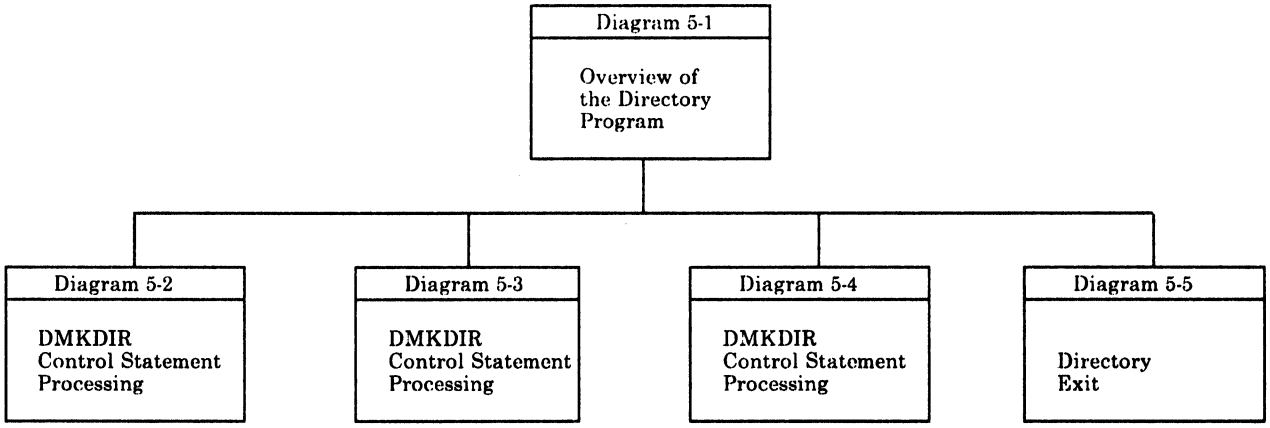
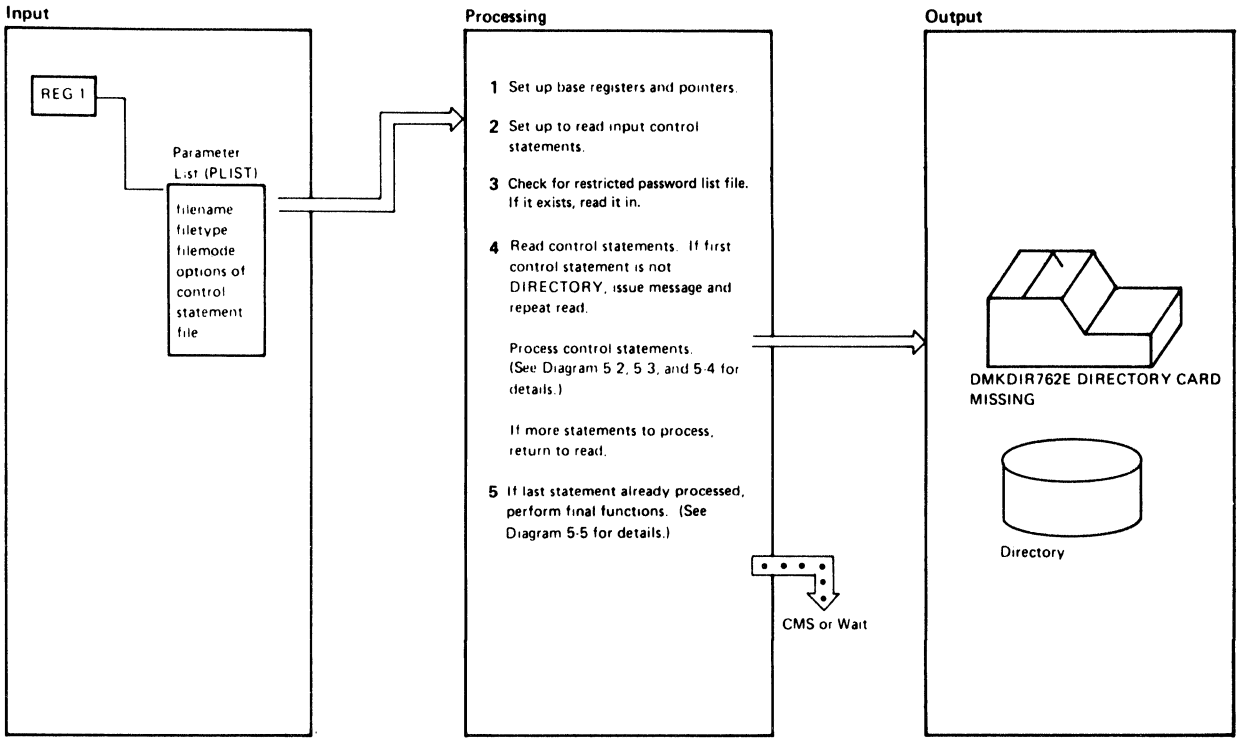


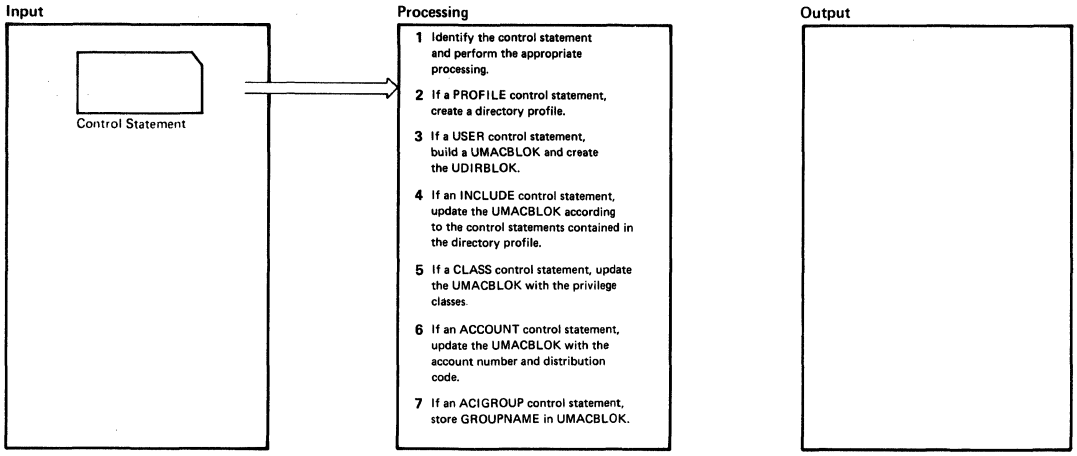
Figure 5-1. Key to the Directory Program Method of Operation Diagrams



Notes	Module	Label	Ref
1 DMKDIR sets up registers 12, 13, and 9 as base registers and sets up pointers to the first UDEVBLOK and the allocation record buffer.	DMKDIR	DMKDIRCT	
2 If running standalone, the header line is printed: USER DIRECTORY CREATION PROGRAM VM/SP RELEASE 2 ENTER CARD READER DEVICE ADDRESS AND OPTIONS The program then reads a response from the console. A read is issued to the card reader indicated (if any). If the operator enters a null line in response to the message, the IPL device is used as the input card reader. If the EDIT option is specified, DIRFLAG is set to X'20'. If running under CMS, set the P-list containing the filename, filetype and filemode of the file containing the directory control cards. If EDIT is specified, the DIRFLAG is set to X'20'. The STATE macro is issued to see if the control statement file exists. If the file is not found, the messages, DMKDIR763E INVALID FILE NAME OR FILE NOT FOUND EOJ DIRECTORY NOT UPDATED	DMKDIR	MSGRET MSG02A DEFAUL13 STOREADD CMS 1 EDITTEST STATE	

Notes	Module	Label	Ref
are displayed and control returns to CMS.		TERM	
3 The STATE macro is issued to see if the restricted password list file exists. If so, the FSREAD macro is issued to read the file into a buffer. If it does not exist, message DMK750W is issued, and processing continues.		STATE	
4 Control statements are read via SVC 202 when the Directory program is run under the control of CMS. When the Directory program runs standalone, the read function is performed either by the GRAPHID routine (if the console is a display device) or by the STARTIO routine in all other cases. The READ routine scans the control statement and branches to the appropriate processing routine. After processing each control statement and executing the associated routine, control returns to READ to process the next control statement.	DMKDIR	READ GRAPHID STARTIO	
5 When the last statement is read and processed, the READ routine branches and links to the EXIT routine.		EXIT	

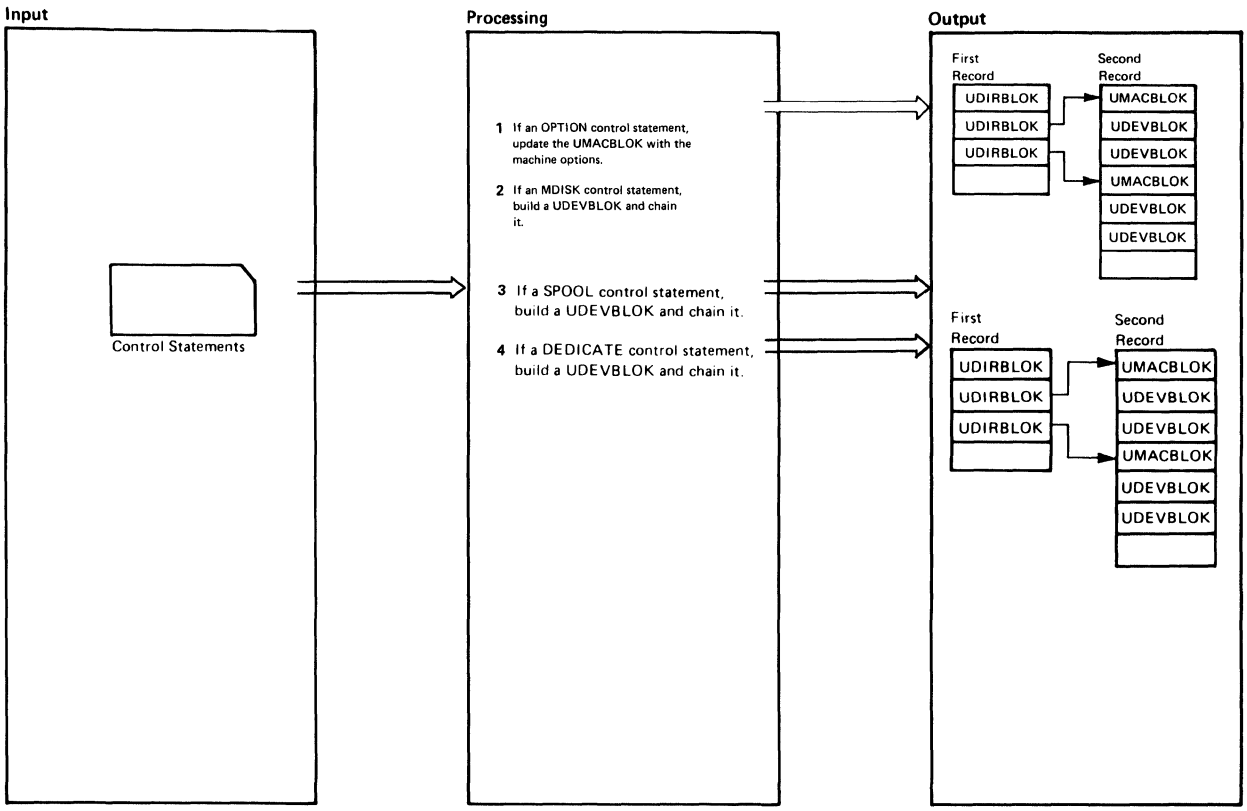
Diagram 5-1. Overview of the Directory Program



Notes	Module	Label	Ref
1 The READ routine branches and links to the SCANNAME routine with register 4 pointing to TABLE 1. TABLE 1 is searched for a keyword matching the control statement name and control is passed to the routine indicated in the corresponding ADCON.	DMKDIR	READ SCANNAME SCAN1	
2 If the PROFILE control statement does not precede all USER control statements, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name appears on the console preceeded by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. The SCANPROF routines creates a directory profile to include common control statements that can be referenced by each user's directory via the INCLUDE control statement.	DMKDIR	SCANPROF ERROR52	
3 If the USER control statement does not proceed a device control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name appears on the console followed by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. The last UDIRBLOK and UMACBLOK are masked off. Update the pointers to the buffers and write out the buffers that are full. The SCANUSER routine locates a UDIRBLOK and initializes it. Then the UMACBLOK is located and initialized.	DMKDIR	SCANUSER ERROR52	
4 If the INCLUDE control statement does not follow a USER control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name appears on the console preceeded by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. The SCANINCL routine updates the UMACBLOK by referencing the control statements contained in the directory profile.	DMKDIR	SCANINCL ERROR52	

Notes	Module	Label	Ref
5 If the CLASS control statement does not preceed a device control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name appears on the console followed by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. The CLASSMAP routine creates a mask in UMACBLOK (UMACCLVL) to indicate the privilege classes allowed for this virtual machine.	DMKDIR	SCANCLAS ERROR52 CLASSMAP	
6 If the ACCOUNT control card does not preceed a device control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name appears on the console followed by the statement that was out of sequence Directory processing is terminated after scanning the remaining statements for syntax. The SCANACCO routine updates the account number (UMACACCT) and distribution code (UMACDIST) fields of the UMACBLOK.	DMKDIR	SCANACCO ERROR52	
7 If the INCLUDE control statement does not preceed a device control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name appears on the console preceeded by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. The SCANACIG routine creates a mask in UDIRBLOK (UDIRGRPN) to identify the user as a member of the GROUPNAME.	DMKDIR	SCANACIG ERROR52	

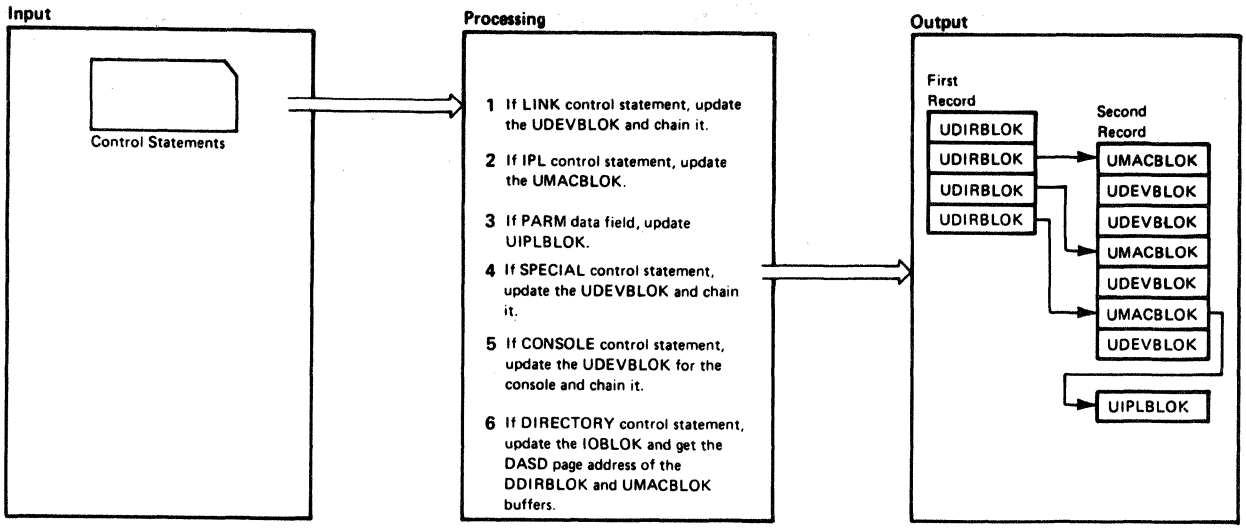
Diagram 5-2. DMKDIR Control Statement Processing - Part I



Notes	Module	Label	Ref
1 If the OPTION control statement does preceed a device control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name appears on the console followed by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. The SCANOPTI routine sets fields in the UMACBLOK to indicate the machine options.	DMKDIR	SCANOPT1 ERROR52	
2 The SCANMDIS routine branches and links to the SCANNAME routine with register 4 pointing to TABLE4. TABLE4 is scanned by device type to get the corresponding device class. The SCANMDIS routine then updates the device type (UDEVTYPE) and class (UDEVTYPC) fields in the UDEVBLOK. The UDEVSTAT field is updated to indicate a T-disk or long block, if either is present, and the number of cylinders is updated. For all disks other than T-disk, the volume serial number, mode, and password field of the UDEVBLOK are initialized. The mode is updated (except for a T-disk). Label Value Comments UDEVR 00 R link-mode UDEVRR 04 RR link-mode UDEVW 08 W link-mode UDEVWR 12 WR link-mode UDEVM 16 M link-mode UDEVMR 20 MR link-mode UDEVMW 24 MW link-mode The SCANMDIS routine then branches to the CHAINDEV routine to chain the UDEVBLOK to the UMACBLOK.	DMKDIR	SCANMDIS SCANNAME SCANMDIS CHAINDEV	

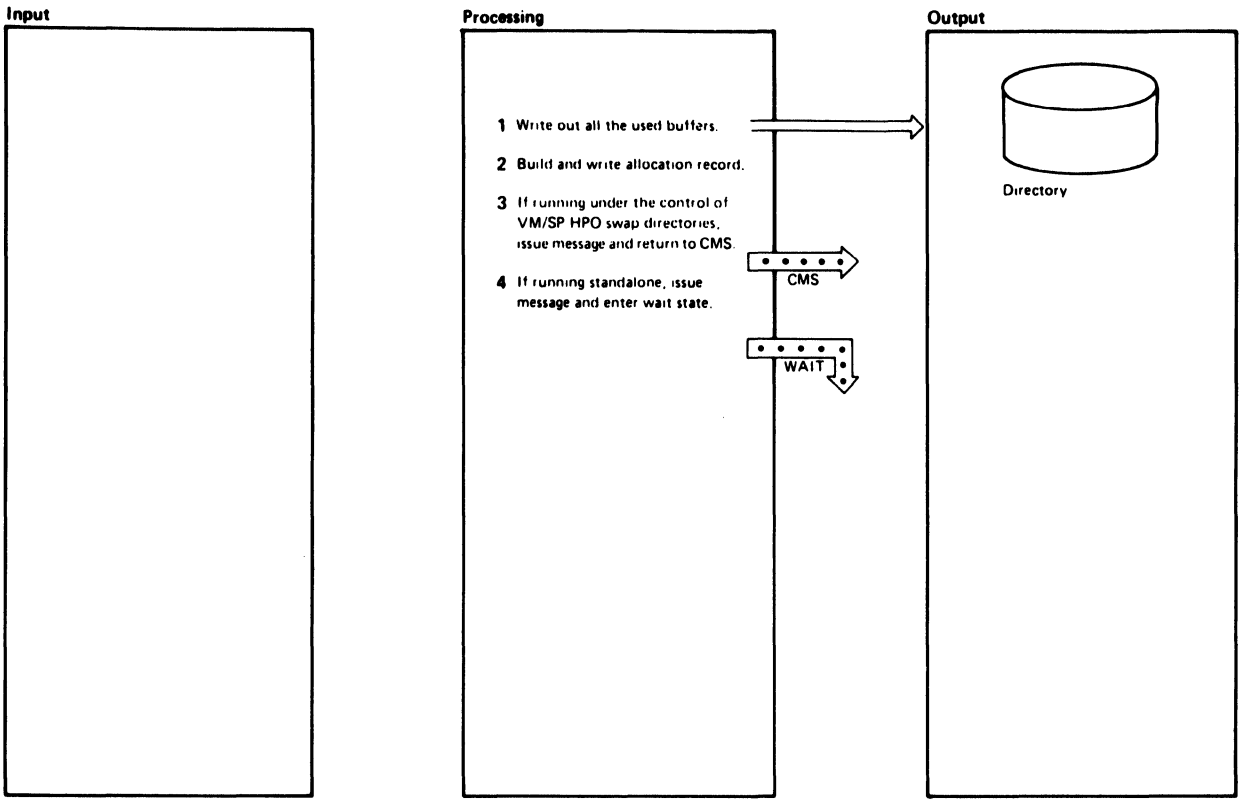
Notes	Module	Label	Ref
3 The SCANSPOO routine builds a UDEVBLOK. The UDEVSTAT field is set to X'08' to indicate a spool device. The virtual device address is stored in the UDEVADD field and the spool class is stored in the UDEVCLAS field. The SCANSPOO routine branches and links to the SCANNAME routine with register 4 pointing to TABLE5. For all device types except the 2540, the spool class is picked up directly from TABLE5. For a 2540 device, the device class is determined in the SCAN2540 routine. The default class is A, except for readers (readers default to class *). The SCANSPOO routine then branches to the CHAINDEV routine to chain the UDEVBLOK to the UMACBLOK.	DMKDIR	SCANSPOO CHAINDEV	
4 The SCANDEDI routine builds a UDEVBLOK. The UDEVSTAT field is set to X'80' to indicate a dedicated device. The virtual device address is stored in UDEVADD field. And, either the volume serial number (UDEVVSER) or user link to disk (UDEVLINK) fields are updated. The SCANDEDI routine then branches to the CHAINDEV routine to chain the UDEVBLOK to the UMACBLOK.	DMKDIR	SCANDEDI CHAINDEV	

Diagram 5-3. DMKDIR Control Statement Processing - Part II



Notes	Module	Label	Ref
1 The SCANLINK routine builds a UDEVBLOK. The UDEVSTAT field is set to X'10' to indicate the device is to be linked at logon time. The virtual device address (UDEVADD) and link device address (UDEVLINK) are updated. The mode (UDEVMODE) is also updated.	DMKDIR	SCANLINK	
Label Value Comments UDEVR 00 R link-mode UDEVRR 04 RR link-mode UDEVW 08 W link-mode UDEVWR 12 WR link-mode UDEVM 16 M link-mode UDEVMR 20 MR link-mode UDEVMW 24 MW link-mode			
2 If the IPL control statement does not follow a USER, ACCOUNT, or OPTION control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING USER user appears on the console followed by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. If there is no PARM data field, the name of the system to be loaded (via IPL) at logon time is placed in the UMACIPL field of the UMACBLOK.	DMKDIR	SCANIPL ERROR52	
3 If there is a PARM data field, fill in the UIPLBLOK with the name of the system to be loaded, the length of the data and the data. Update the UMACIPL field of the UMACBLOK to point to the UIPLBLOK. Set a code of X'00' in the UMACIPLX field to indicate that there is PARM data.			
4 The SCANSPEC routine builds a UDEVBLOK for a special device. The virtual device address is stored in UDEVADD, the device type is stored in UDEVTYPE, and the class is stored in UDEVTYPC. The SPANSPEC routine branches and links to the SCANNAME routine with register 4 pointing to TABLE2. The device type and class is picked up directly from TABLE2 for a 3270 and pseudo-timer.	DMKDIR	SCANSPEC SCANNAME	
The SCANNAME routine branches (via an ADCON) to the SCANCTCA, SCAN2701, SCAN2702, and SCAN2703 routines to determine the device type and class of channel-to-channel adapter, or 2701, 2702, and 2703 special device.		SCANCTCA SCAN2701 SCAN2702 SCAN2703	
5 The SCANCONS routine builds a UDEVBLOK for the console. The virtual device address is stored in UDEVADD, the device type is stored in UDEVTYPE, and the class is stored in UDEVTYPC. The default class is T. The SCANCONS routine branches and links to the SCANNAME routine with register 4 pointing to TABLE3. The device type and class are picked up directly from TABLE3.	DMKDIR	SCANCONS	
6 If the DIRECTORY control statement is not the first control statement, DMKDIR752E STATEMENT SEQUENCE ERROR FOLLOWING USER user appears on the console followed by the statement that was out of sequence. Directory processing is terminated after scanning the remaining statements for syntax. The SCANDIRE routine sets up to update the IOBLOK. The output device address is stored in DASDADD, and the serial number is stored in DASDVSER. The DIRFLAG is set to a hexadecimal value that indicates the device type of the output unit. Then, the SCANDIRE routine gets the pointer to the first page of the directory and machine buffer areas.	DMKDIR ERROR52	SCANDIRE	

Diagram 5-4. DMKDIR Control Statement Processing - Part III



Notes	Module	Label	Ref
1 All of the user directory, user machine, and user device buffers that were used are written. The buffers are written out by loading the DASD address into register 2, loading the buffer address into register 1, and then branching and linking to the WRITE routine.	DMKDIR	EXIT	
2 The allocate table is built. A table setting of X'04' indicates an unallocated cylinder and X'0C' indicates an allocated cylinder. The VOL1 and allocation records are written.	DMKDIR	SCANALLO	
3 First the return PSW is set up and Registers 1 and 2 are set to the volume serial number. The user directories are swapped via a DIAGNOSE call to DMKUDRDS. The DIAGNOSE will program check if the user is not class A, B, or C. The directories are not swapped if the volume is not found in the OWNDLIST or if an I/O error occurs under CP. The message EOJ DIRECTORY UPDATED appears on the console and control returns to CMS. If no errors occur, and if the active system directory was updated, the directories are swapped. The message EOJ DIRECTORY UPDATED AND ON LINE appears on the console and control returns to CMS.	DMKDIR	MOVEPSW LOOP 11	

Notes	Module	Label	Ref
4 If not running under VM/SP HPO, the message EOJ DIRECTORY UPDATED appears on the console and the wait state is entered by loading the SVCNEW PSW.	DMKDIR	BARE	

Diagram 5-5. Directory Exit

Program Organization

This section includes a program description of the DMKDIR module.

DMKDIR

Creates the directory on a system owned volume.

Entry Points

DMKDIRCT is the entry point when the directory program is executed standalone and DMKDIRED is the entry point when the directory program is executed under the control of CMS.

Routines Called

None

Attributes

Not serially reusable.

Exit Conditions

If executed under the control of CMS, register 15 contains a return code at exit.

Return Code	Meaning
1	Invalid filename or file not found.
2	Error loading the directory.
3	Invalid option from CMS.
4	Directory not swapped, user class not A, B, or C.
5	Directory not swapped, system (old) directory locked.
6	Directory not swapped; the directory the system is using is not the directory just updated.
1xx	Error in CMS RDBUF routine.
2xx	Error in CMS TYPLIN routine.

where xx is the CMS routine return code.

Register Usage

R0: Work register.
R1: Pointer to input field.
Pointer to IOB.
Pointer to output buffer.
Work register.
R2: Input count from SCANCARD.
DASD address.
Work register.
R3: Work register.
R4: Work register.
R5: Branch and link return address.
Pointer to the next UDEVBLOK.
Work register.
R6: RDIRBUF, pointer to the UDIRBLOK buffer.
R7: RMACBUF, pointer to the UMACBLOK buffer.
R8: RDEVBUF, pointer to the UMDEVBLOK buffer.
R9: Base register 3.
R10: RMAC, pointer to UMACBLOK.
R11: RDEV, pointer to UDEVBLOK.
R12: Base register 1.
R13: Base register 2.
R14: Return address.
R15: RDIR, pointer to UDIRBLOK.

External References

DMKURDS is called via a DIAGNOSE instruction to write the new directory on DASD.

Directory

Figure 5-2 is an alphabetic list of the major labels of the Directory program. The associated method of operation diagram is referenced and a brief description of the function performed at the point in the program corresponding to each label is included.

Label	Diagram	Description
BARE	5-5	Directory program exit when not running under the control of VM/SP HPO.
BILDUDIR		Builds UDIRBLOK.
BILDUMAC		Builds UMACBLOK.
BINCONV		Converts decimal numbers to binary.
CHAINDEV	5-2, 5-3	Chains UDEVBLOK to UMACBLOK.
CLASSMAP	5-2	Builds a mask in UMACBLOK indicating privilege classes allowed for this virtual machine.
CMS1	5-1	Sets up the parameter list identifying the file containing the control statements when running under CMS.
CMS3		Reads CMS control cards via SVC 202.
COMMERRP		Prints queued error messages.
COMPARE		Compares keywords and sets condition codes.
DECCONV		Converts decimal numbers to hexadecimal.
DEFAUL13	5-1	Defaults to the IPL device for control statement input device when running standalone.
DMKDIRCT	5-1	Sets up base registers and initializes pointers.
EDITTEST	5-1	Sets DIRFLAG to X'20' to indicate edit, if EDIT is specified when the Directory program is run under VM/SP HPO.
EOF		Simulates a USER card.
ERROR51		Error processing for invalid operand.
ERROR52	5-2 5-4	Issues message when a control statement is out of sequence.
ERROR58		Issues message DMKDIR758E.
ERROR62		Issues message DMKDIR762E.
EXIT	5-1 5-5	End-of-job processing for Directory Program.
GETALT		Makes switch from first to second device address specified.
GETCYLNO		Fills in cylinder relocation for minidisks.
GETPAGE		Assigns a DASD page address.
GRAPHID	5-1	Reads the input control statements from a display terminal when the directory program is not running under CMS.
HEXCONV		Converts hexadecimal numbers to binary.
LONG		Turns on long block indicator for minidisks.
LOOP11	5-5	Calls DMKUDRDS via the DIAGNOSE instruction to swap directories when running under VM/SP HPO.
MOVECPT		Sets up current control statement pointer.
MOVEDISP		Updates UMACBLOK.
MOVEPSW	5-5	Sets up return PSW before issuing DIAGNOSE to call DMKUDRDS.
MSGRET	5-1	When running standalone, a header line is printed.
MSG02A	5-1	Requests input device when running standalone.
MSGWRITE		Writes messages to the terminal.
NOTUSED		Updates UMACBLOK pointer.
POINTDEV		Updates UDEVBLOK pointer.

Figure 5-2 (Part 1 of 2). The Directory Program Label Directory

Label	Diagram	Description
READ	5-1 5-2	Reads control statements and branches to appropriate processing routine.
REREAD		Sets up pointer to control statement read buffer.
RET1		Scans control statements.
SCANACCO	5-2	ACCOUNT statement processing routine.
SCANACIG		ACIGROUP statement processing routine.
SCANALLO	5-5	Builds allocation record.
SCANCARD		Scans the control statement for the next operand.
SCANCLAS	5-2	Process the CLASS control statement.
SCANCONS	5-4	CONSOLE statement processing routine.
SCANCTCA	5-4	Updates the UDEVBLOK and chains the control unit to the UDEVBLOK for channel-to-channel adapters.
SCANDEDI	5-3	DEDICATE statement processing routine.
SCANDIRE	5-4	DIRECTORY statement processing routine.
SCANINCL		INCLUDE statement processing routine.
SCANIPL	5-4	IPL statement processing routine.
SCANLINK	5-4	LINK statement processing routine.
SCANMDIS	5-2	MDISK statement processing routine.
SCANNAME	5-2 5-4	Scans the name table until a match is found. Register 4 points to the name table. If the name field is a constant, it is put in the UDEVBLOK. If the name field is an address, control is passed to that address.
SCANOPTI	5-2	OPTION statement processing routine.
SCANPROF		PROFILE statement processing routine.
SCANSPEC	5-4	SPECIAL statement processing routine.
SCANSPOO	5-3	SPOOL statement processing routine.
SCANUSER	5-2	USER statement processing routine.
SCAN1	5-2	Points register 4 to TABLE1, then branches and links to SCANNAME routine to determine the appropriate control statement processing routine.
SCAN2311		Updates the UDEVBLOK for 2311 disks.
SCAN2540		Updates the UDEVBLOK for 2540 devices.
SCAN2701	5-4	Updates the UDEVBLOK for 2701 devices.
SCAN2702	5-4	Updates the UDEVBLOK for 2702 devices.
SCAN2703	5-4	Updates the UDEVBLOK for 2703 devices.
STARTIO	5-1	Reads the input control statements if the directory program is not running under CMS.
STATE	5-1	Checks that control statement file exists.
STOREADD	5-1	Sets the DIRFLAG to X'20' to indicate edit, if EDIT is specified when the Directory program is run standalone.
TERM	5-1	At end of processing, returns control to CMS if running under VM/SP HPO.
TESTBUFF		Tests to see if UDEVBLOK was used.
TESTUDEV		Gets DASD address of UMACBLOK.
UPDATE		Points to next UDEVBLOK.
UPDATECT		Updates device count in UMACBLOK.
WRITE		Writes the directory on DASD.
XERR754E		Keeps track if first address on DIRECT statement not operational.
XERR755E		Keeps track of I/O errors.
XERR761E		Keeps track of volid.

Figure 5-2 (Part 2 of 2). The Directory Program Label Directory

Data Areas

The directory exists on disk as 4K (page size) records. The VOL1 label (cylinder 0 track 0 record 3), on the volume containing the directory, points to the directory. The directory starts with the first available record.

The first UDIRBLOK is a dummy UDIRBLOK. Its UDIRDISP field points to the last UDIRBLOK in that record. The UDIRDASD field points to the next UDIR record, or, if it is the last record, it contains zeros. The second UDIRBLOK in the first record points to the UMACBLOK for that user, located in the second record. In turn, the UMACBLOK points to the first UDEVBLOK for that user. It is the second block in the second record. The last UDEVBLOK for this user has a pointer of all zeros.

The directory entry for the second user consists of a UDIRBLOK in the first record and associated UMACBLOK, and UDEVBLOKs in the second record. When a record becomes full, the chain continues into the next available record.

When the directory is created, all UDIRBLOKs are grouped 169 blocks per record. The UMACBLOK and UDEVBLOKs are sequentially chained into a separate record. If the record becomes full before the end of the chain, the chain overflows into the next available record. The formula to find the number of records is:

$$\frac{NU}{169} + ((NU \times 3.66) + (NM \times 2.33) + (ND \times 1.33)) = NR$$

where:

NU is the number of user records.

NM is the number of MDISK cards describing a virtual disk (not T-Disk).

ND is the total number of MDISK (describing T-Disk space), SPOOL, LINK, SPECIAL, CONSOLE, and DEDICATE cards.

NR is the total number of records used.

For count-key-data DASD, to find the number of cylinders, divide the total number of records by 32 for 2314/2319 devices, by 57 for 3330 series devices, or by 24 for 3340 and 2305 series devices, by 96 for the 3375 and by 150 for the 3380. For FB-512 DASD, the total number of pages needed equals NR. To ensure that a new directory will not overlap an existing directory, allow space for two directories or allocate a new directory each time the directory is created.

The following data areas are used by the directory program:

- The UDEVBLOK (user device block), built in the UDEVBLOK or UMACBLOK buffer.
- The UDIRBLOK (user directory block), built in the DIRBLOK buffer.
- The UMACBLOK (user machine block), built in the UMACBLOK buffer.

These data areas, as well as a figure showing the user directory format and the relationship of the above blocks, are described in the *VM/SP HPO Data Areas and Control Block Logic – CP*.

Note: If you install the speed matching buffer feature (Feature #6550) with the 3380, the extended count-key-data channel programs are used.

Diagnostic Aids

Figure 5-3 lists the messages issued by the Directory program. The label of the message and the associated method of operation diagram are included in the list.

Message Code	Label	Diagram	Message Text
DMKDIR536I	ERR536		raddr devname REPORTS DISABLED INTERFACE; FAULT CODE = cccc; NOTIFY CE
DMKDIR751E	ERROR51A		INVALID OPERAND - operand
DMKDIR752E	ERROR52	5-2, 5-4	STATEMENT SEQUENCE ERROR FOLLOWING PROFILE/USER name
DMKDIR753E	ERROR53		OPERAND MISSING
DMKDIR754E	ERROR54A STARTIO READ WRITE		DEV raddr NOT OPERATIONAL
DMKDIR755E	ERROR55A WRITE		I/O ERROR raddr CSW csw SENSE sense
DMKDIR756E	ERROR56A		PROGRAM CHECK PSW = psw
DMKDIR757E	ERROR57		MACHINE CHECK
DMKDIR758E	ERROR58 CHAINDEV		DUPLICATE UNIT DEFINITION
DMKDIR760E	ERROR60 GETPAGE		NOT ENOUGH SPACE ALLOCATED FOR {DIRECTORY|OVERRIDES}
DMKDIR761E	ERROR61A SCANDIRE		VOLID READ IS volid1 NOT volid2 (ON raddr)
DMKDIR762E	ERROR62 READ		{DESTINATION|DIRECTORY} MISSING
DMKDIR763E	ERROR63 STATE	5-1	INVALID FILENAME OR FILE NOT FOUND
DMKDIR764E	ERROR64		ERROR IN routine
	MSG04	5-1	EOJ DIRECTORY NOT UPDATED
	MSG01	5-5	EOJ DIRECTORY UPDATED
	MSG03	5-5	EOJ DIRECTORY UPDATED AND ON LINE
	MSG02	5-1	USER DIRECTORY CREATION PROGRAM VM/SP HPO RELEASE 5
	MSG02A	5-1	ENTER CARD READER DEVICE ADDRESS AND OPTIONS
DMKDIR765E	ERROR65		INVALID CLASS DEFINITION
DMKDIR766E	ERROR66		DUPLICATE CLASS DEFINITION
DMKDIR767W	ERROR67		PASSWORD CHANGED TO NOLOG FOR userid
DMKDIR768E	ERRSNS		FOR userid- MOVE vaddr TO A vcutype VCU
DMKDIR771E	ERROR71		RESTRICTED PASSWORD AND NOLOG INVALID FOR userid

Figure 5-3. The Directory Program Messages

Chapter 6. The DASD Dump Restore Program

Introduction

The DASD Dump Restore program executes under the control of CMS via the DDR command. It performs five functions for direct access storage devices (both count-key-data and FB-512). The five functions are:

- Dump[1]
- Restore
- Copy
- Print
- Type

DDR can store data on tape in a compact format. DDR does this by compressing strings of duplicate data into a smaller amount of space and reducing the amount of space necessary to represent the characters in the data. This uses less tape space than the standard format. The compact format is an option (COMPACT) specified by the user on the OUTPUT control statement for the dump function. The COMPACT option is ignored on the OUTPUT control statement for the restore, print, and type functions and on the INPUT control statement. If it is used on the OUTPUT control statement for the copy function, a system message is issued saying the COMPACT option is ignored. It is valid only for tape output. Tapes created by DDR which are in the compact format may be used as input to the restore, copy, print, and type functions.

Note: Tapes created by DDR which are in the compact format cannot be used as input to earlier levels of DDR.

---

1 The FTR operand is only valid with the DUMP function statement. It requests the use of the full track read feature for those devices supporting the feature (3310, 3330, 3340, 3350, 3370, 3375, and 3380).

DUMP

The dump function saves data from a direct access volume on magnetic tape. The output tape may be put into compact format. For the count-key-data (non-full track read and non-compact format), the data is saved cylinder by cylinder. The format of the tape is:

- Record 1, volume header record – data describing the volume.
- Record 2, track header record – a list of count fields to restore the track and the number of data records written on tape. After the last count field the record contains key and data records to fill the 4K buffer.
- Record 3, track data records – key and data records packed into 4K blocks with the last block truncated.
- Record 4, either the end-of-volume or end-of-job trailer label. The end-of-volume label contains the same information as the next volume header record except that the ID field contains EOV. The end-of-job trailer label contains the same information as record 1 except that the cylinder number field contains the disk address of the last record on tape and the ID field contains EOJ.

For FB-512 devices, (in either compact or noncompact format), the data is saved in groups of FB-512 blocks. Any number of blocks can be dumped. The format of the tape is:

- Record 1, volume header record – data describing the volume.
- Record 2, track header record – data describing the group of FB-512 data blocks that follow, as well as the number of tape records required to hold these blocks. After the control data, the record contains FB-512 data to fill the 4K buffer.
- Record 3, FB-512 data records – contains the FB-512 blocks dumped from the FB-512 volume.
- Record 4, either the end-of-volume or end-of-job trailer label. The end-of-volume label contains the same information as the next volume header record except that the ID field contains EOV. The end-of-job trailer is the same as the EOV label except that the ID contains EOJ and the block-number field contains the number of the last block on the tape.

For count-key-data (full track read format or compact format), the data is saved cylinder by cylinder as follows:

- Record 1, volume header record – data describing the volume.
- Record 2, track header record – length of track, density of the tape, and number of count fields in the track followed by track contents.

- Record 3, track data records – count-key-data records in 8K blocks for 800 bpi or 1600 bpi tapes, 12K blocks for 1600 bpi tapes, or 49K for 6250 bpi tapes. The last block being a short block.

- Record 4, either the end-of-volume or end-of-job trailer label. The end-of-volume label contains the same information as the next volume header record except that the ID field contains EOV. The end-of-job trailer label contains the same information as record 1 except that the cylinder number field contains the disk address of the last record on tape and designates the allocation of cylinders and the ID field contains EOJ.

RESTORE

The restore function transfers data from a tape created by the DDR dump function to a DASD device. The data may be restored only to a device of the same type as the device from which it was dumped.

A tape in compact format may be used as input. DDR checks to see if the input is in compact format, and expands the data, if needed.

COPY

The copy function copies data from one device to another device of the same type (DDR does not copy from count-key-data to FB-512 or from FB-512 to count-key-data). For disk-to-disk operations, data may be reordered on a cylinder or block basis. If copying from tape-to-tape, the input tape must have been created by the DDR dump function.

A tape in compact format may be used as input. For a tape-to-tape copy, the output tape will be in the same format (compact or standard) as the input tape. The COMPACT option on the OUTPUT control statement is not valid for the COPY function. If it is specified, a message stating, 'COMPACT OPTION IS IGNORED' is displayed.

PRINT

The print function prints both hexadecimal and EBCDIC representations of selected records of a DASD or Tape Volume device on a printer. The word "record" here means a particular block when referring to FB-512 DASD, and a particular page when referring to CKD DASD. A tape in compact format may be used as input.

TYPE

The type function displays at the terminal both hexadecimal and EBCDIC representations of selected records of a DASD or Tape Volume device. The word "record" here means a particular block when referring to FB-512 DASD, and a particular page when referring to CKD DASD. A tape in compact format may be used as input.

Method of Operation

The method of operation diagrams describe the major functions of the DDR (DASD Dump Restore) program. The relationship of the method of operation diagrams is described in Figure 6-1.

The five functions of DDR apply equally to FB-512 data as they do to count-key-data devices. The method of operation for each is the same at a given level of description. The main difference is the unit of DASD data that DDR can handle for FB-512 devices. This unit is an FB-512 block. This means that DDR can copy, dump, restore, print, or type any number of blocks in increments as small as one block. For count-key-data, the unit of copy, dump, or restore is one cylinder; for print or type it is one record. This difference leads to a different control statement format, as well as different control statement format, as well as different internal processing. This distinction is noted in the following diagrams. This distinction is noted in the following diagrams, as appropriate.

Diagram 6-1 describes the major functions of the DDR program.

Diagram 6-2 shows the control statement processing for the DDR program.

Diagram 6-3 describes the Dump function.

Diagram 6-4 describes the Dump function with streaming.

Diagram 6-5 describes the Restore function.

Diagram 6-6 describes the Restore function with streaming.

Diagram 6-7 describes the Copy function.

Diagram 6-8 describes the Print function.

Diagram 6-9 describes the Type function.

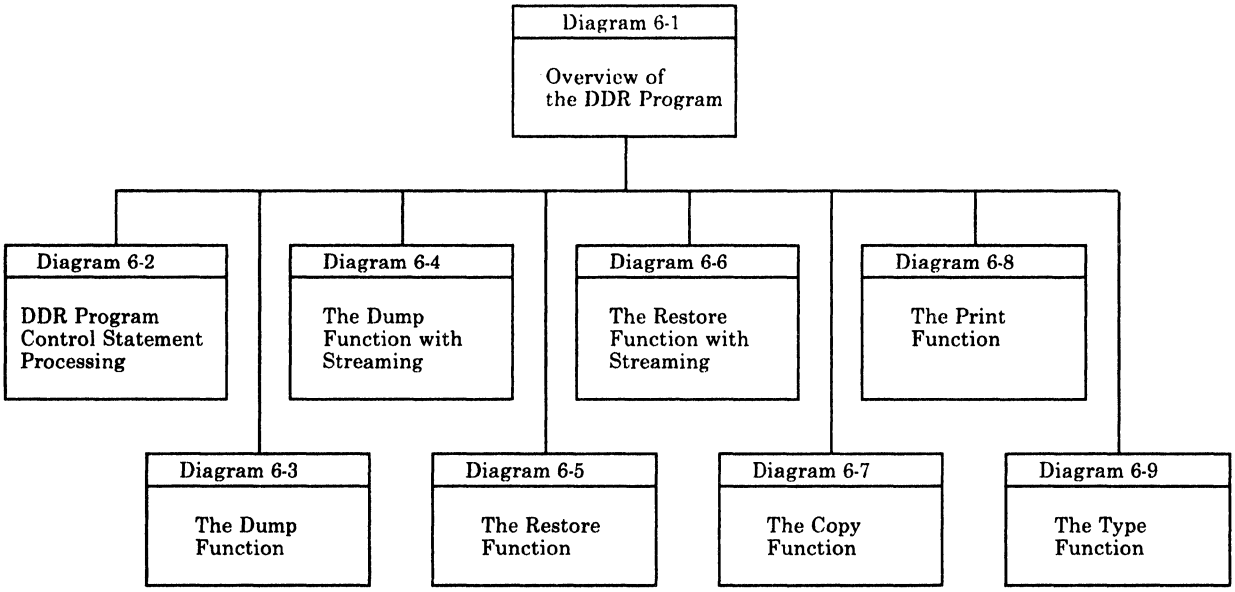
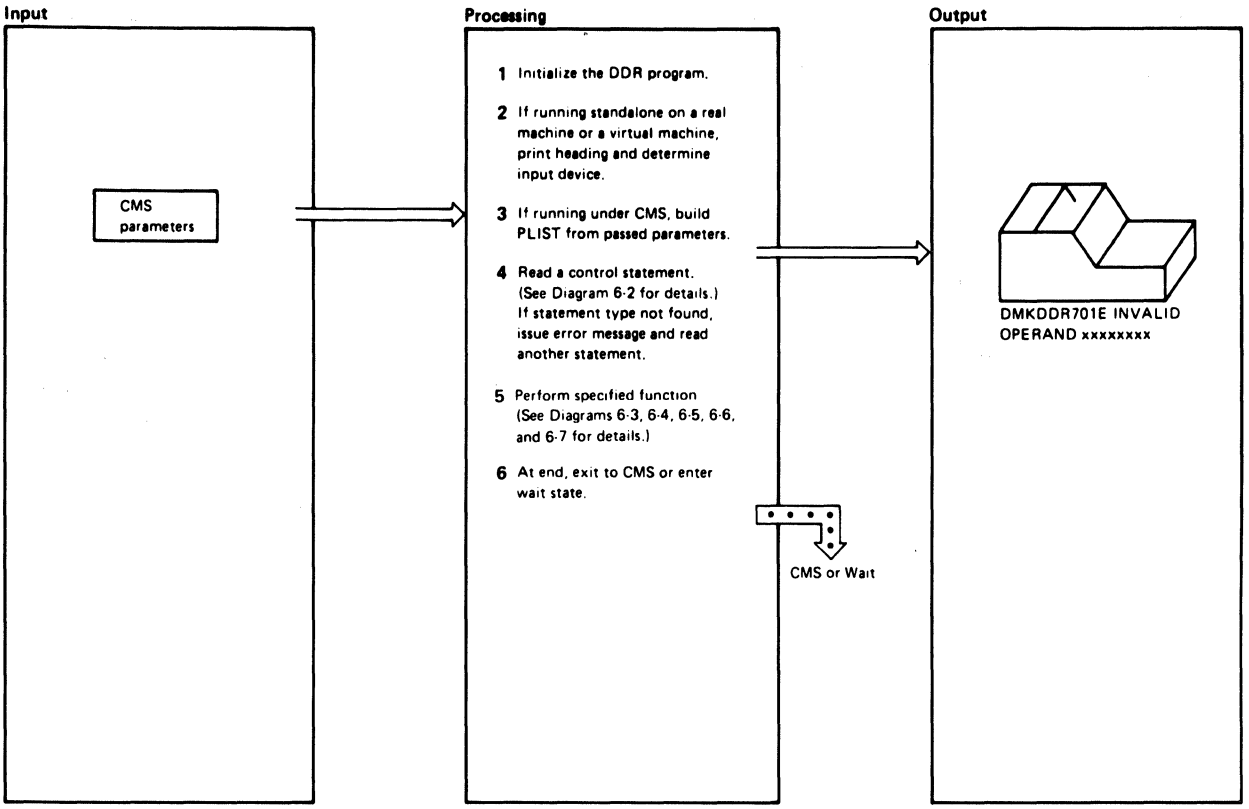


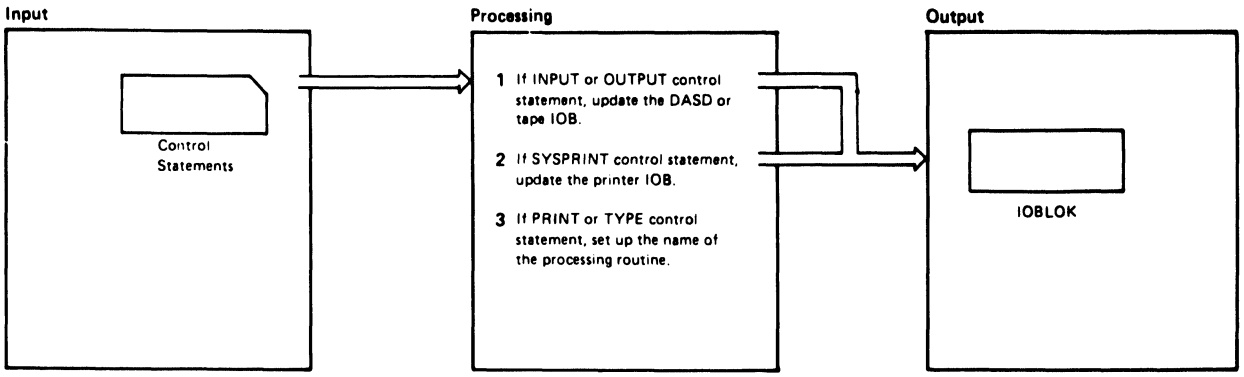
Figure 6-1. Key to the DASD Dump Restore Program Method of Operation Diagrams



Notes	Module	Label	Ref
1 The DDR program is initialized and the base registers (9, 10, 11, 12, and 13) are set up. Register 8 is initialized to the data buffer address.	DMKDDR	DMKDDREP	
2 The heading VM/370 DASD DUMP/RESTORE PROGRAM RELEASE n is displayed. If no input device is specified, the IPL device is used as the input device.	DMKDDR	NEWADD	
3 DMKDDR builds a PLIST if parameters are passed from CMS to the DDR program.	DMKDDR	CMS1	
4 DMKDDR reads the control statement. The routine needed to initialize the DDR function is found by branching and linking to the SCANNAME routine and searching the name table.	DMKDDR	GTCARD	
5 The designated function is performed. At its end, control returns to the GTCARD routine to read the next control statement and perform the next function.	DMKDDR		
6 When the last control statement is read and processed the GTCARD routine branches to the EXIT routine.	DMKDDR	EXIT	

Notes	Module	Label	Ref
The end of job statement (MSG001) is displayed. If running under CP the SYSPRINT device is closed and control returns to the CMS command environment.		CMS8	
If running standalone, the wait state is entered.		TESTCMS	

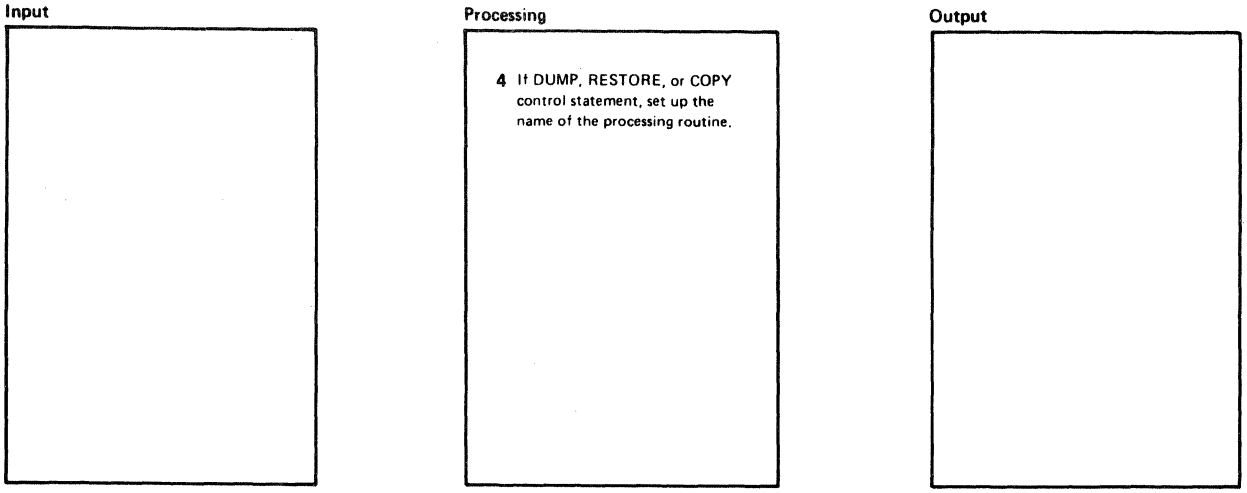
Diagram 6-1. Overview of the DDR Program



Notes	Module	Label	Ref
1 The address of the IOB is loaded into register 15. DMKDDR gets the unit address. The unit address (IOBUADD) and alternate tape address (IOBATAPE) fields of the IOB are filled in. DMKDRR reads the device type from the control statement and then branches and links to the SCAN routine. The SCAN routine searches a table of valid devices and picks up the device class and type. The class (IOBCLASS) and type (IOBTYPE) fields are updated. The codes for the various device classes are contained in the DEVTYPES COPY file. If a DASD serial number is specified, the volume serial number (IOBVSER) field is updated. If tape options are specified, the IOB is updated. Field — Options IOBSKIP — number of times file to be forward spaced. IOBMODE — tape mode. X'C3' indicates 9 track 1600 BPI X'CB' indicates 9 track 800 BPI X'DB' indicates 18 track 38K BPI X'D3' indicates 9 track 6250 BPI IOBDISP — disposition of tape. X'07' indicates rewind X'0F' indicates rewind and unload X'03' indicates tape is not to be repositioned Either of the following error messages may be displayed while processing INPUT or OUTPUT control statements. DMKDDR701E INVALID OPERAND – xxxxxxxx DMKDDR703E OPERAND MISSING	DMKDDR	SCANINPU SCANOUTP	

Notes	Module	Label	Ref
If either of these errors occurs, the control statement is ignored and control returns to the GTCARD routine to read the next control statement. 2 The address of the printer IOB is loaded into register 15. The printer unit address is placed in the IOBUADD fild of the IOB. If an error occurs, either message DMKDDR701E INVALID OPERAND – xxxxxxxx DMKDDR703E OPERAND MISSING is displayed. The statement in error is ignored, and control returns to the GTCARD routine to read the next control statement. 3 The translate table is set up. If TYPE is specified, the LOWERCAS table is used. If PRINT is specified, the UPPERCAS table is used. The routine name is set up: PRINT or TYPE. The start address (default is track 0 record 0 or block 0 for FB-512), and the stop address (default is last track and last record or the last block for FB-512) are set up. If TYPE is specified, the console skips one line. If PRINT is specified, the printer skips to channel 1. If there is an error in the control statement, either error message DMKDDR701E INVALID OPERAND – xxxxxxxx DMKDDR703E OPERAND MISSING is displayed. The control statement is ignored, and the next control card is read by the GTCARD routine.	DMKDDR DMKDDR	SCANSYP SCANPRIN SCANTYPE	

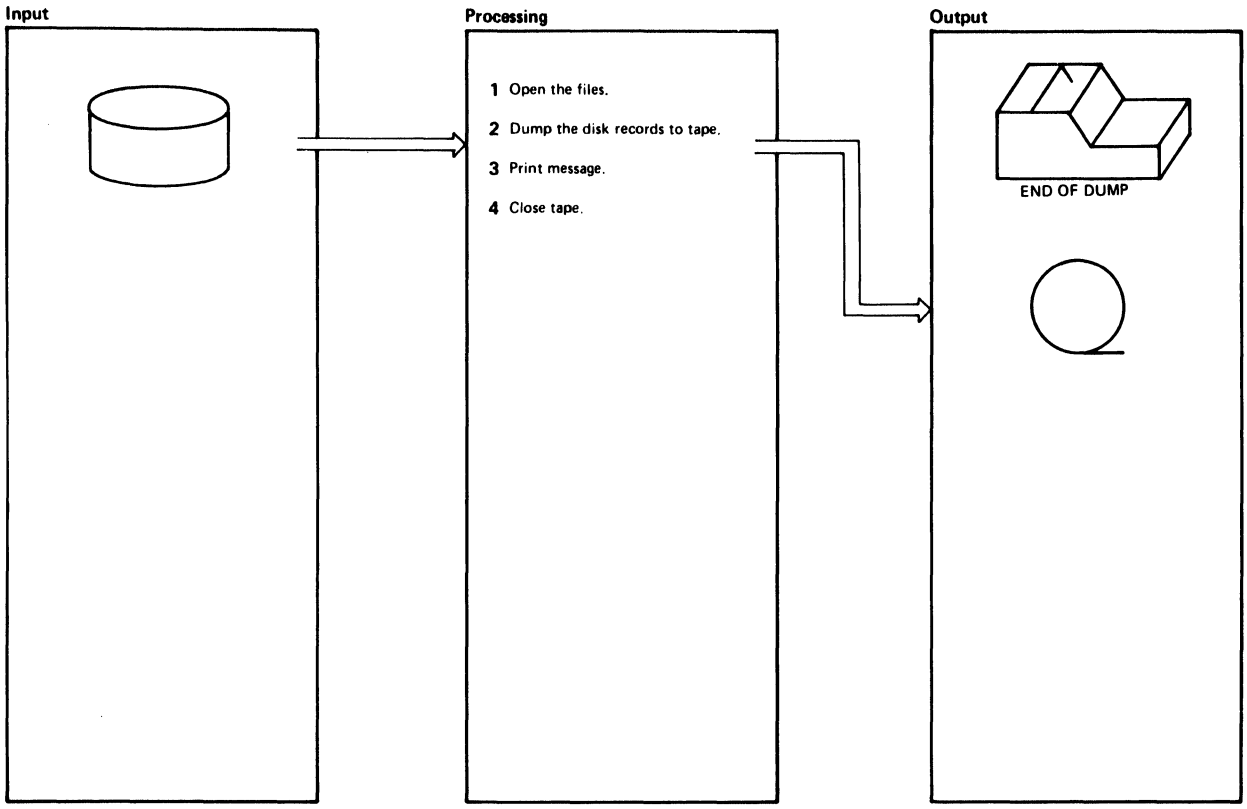
Diagram 6-2. DDR Program Control Statement Processing (Part 1 of 2)



Notes	Module	Label	Ref
4 If DUMP control statement, set the processing routine name to DUMP.	DMKDDR	SCANDUMP	
If RESTORE control statement, set the processing routine name to RESTORE.		SCANREST	
If COPY control statement, set the processing routine name to COPY.		SCANCOPY	
For the dump function, the input must be a DASD and output a tape.			
For the restore function, the input must be a tape, the output a DASD.			
For a copy function, the input and output devices must be the same class and type. If the input device contains more cylinders or blocks than the output device the following message is issued: DMKDDR725R ORIGINAL INPUT DEVICE WAS(IS) LARGER THAN OUTPUT DEVICE			
The operator must determine if the copy function is to continue.			
For the dump function, the COMPACT option is valid. For the restore function, the COMPACT option is ignored. For the copy function, if the COMPACT option is specified, the following message is issued and processing continues. DMKDDR731I COMPACT OPTION IS IGNORED FOR COPY OPERATIONS		DDR731	

Notes	Module	Label	Ref

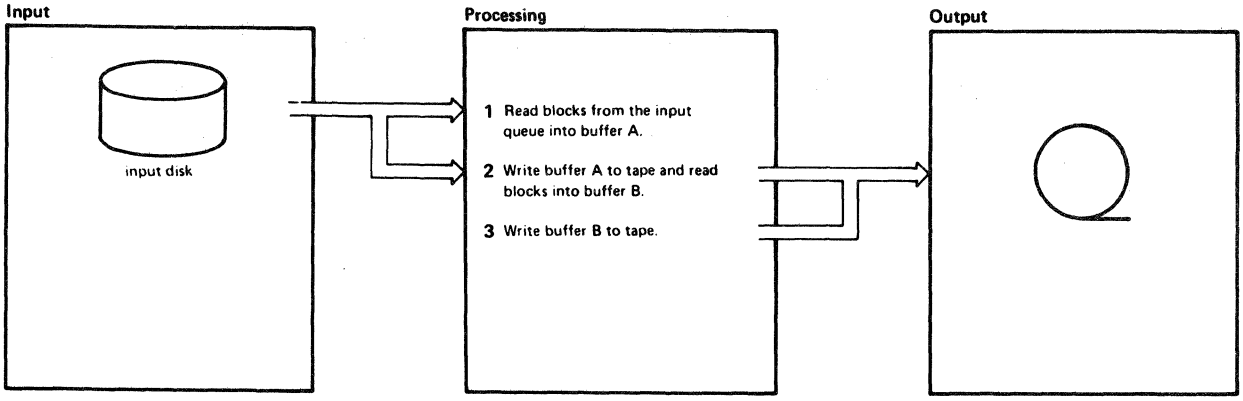
Diagram 6-2. DDR Program Control Statement Processing (Part 2 of 2)



Notes	Module	Label	Ref
1 The input disk is opened by branching and linking to the OPENDASD routine. The extent table is updated to define the extents to be dumped. Each statement updates the extent table until a null line, an INPUT statement, or OUTPUT statement is read	DMKDDR	OPENIN GETEXT	
The output tape is opened, the proper number (if any) of records is skipped and the volume header record (VHR) is written.		OPENOUT	
2 Prints the headings indicating the function being performed and the date and time of the dump.	DMKDDR	PRINTH MSG004	
The read, write, and update cycle continues until the indicated disk extents are dumped to tape. Starting at the first disk extent (CYLSTART or BLKSTART), the disk records are read. The record is written on tape and the pointers are updated to the next disk record. If the COMPACT option is specified, a branch is made to the encoding routines and data is written in compact format. The dump cycle continues until the last disk extent CYLSTOP or BLKSTOP, is dumped to tape.		BUILDTHR CCMP300 DMKDNC DMKDNT DMKDDR	TESTOUT VPDTADD

Notes	Module	Label	Ref
3 The message END OF DUMP indicates that the dump function has successfully terminated.	DMKDDR	CLOSEJOB	
If the COMPACT option was specified, the following messages are displayed: BYTES IN____ BYTES OUT____ TRACKS NOT COMPACTED ON TAPE____ BLOCKS NOT COMPACTED ON TAPE____ ___FEET WRITTEN ON___BPI TAPE		CCMP200	
4 The trailer record is written on the output tape. If the tape disposition was specified on the DUMP control statement, the tape is so positioned now. Control returns to the control statement read routine (GTCARD) to read and process the next control statement.	DMKDDR	EOJ	

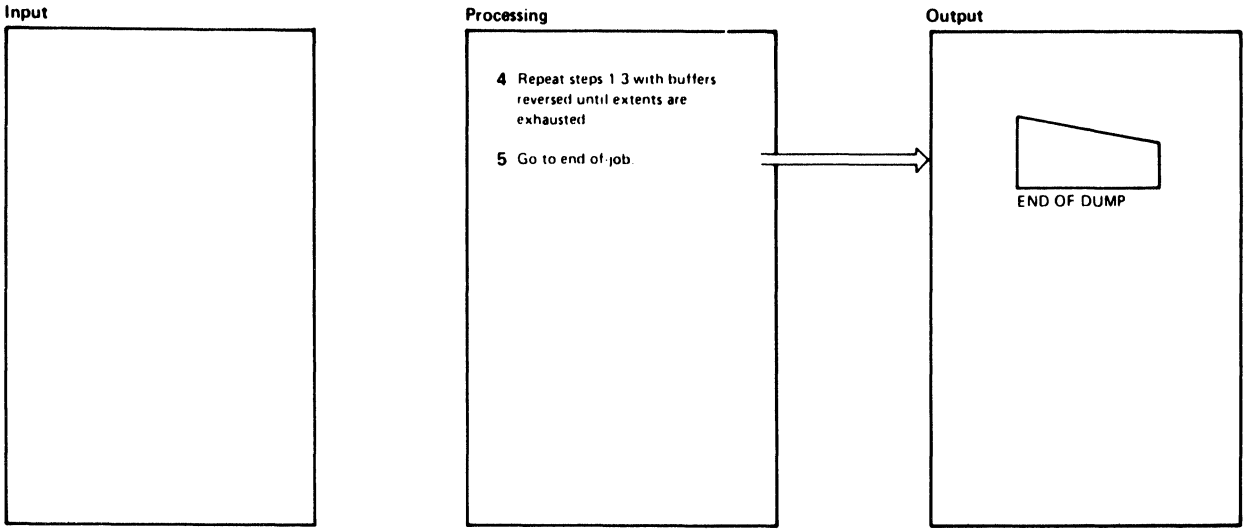
Diagram 6-3. The Dump Function



Notes	Module	Label	Ref
1 For the dump function, the work queues are initialized so that the DASD input queue (Q2) owns both buffers. Therefore, Q2 is selected first. A channel program is built to read blocks from the input queue. The number it reads is the number that will fit in the buffer or the number that remains in the current extent of the input disk (whichever is smaller). It starts reading from the block number in INBLADD. It sets control fields in the THR to describe the data as follows: THRFRSBL – the block number of the first block read THRLASBL – the block number of the last block read. THRNBLK – the number of blocks to be written. THROBLAD – the number to be assigned to the first block when it is output.	DMKDDR	QSEARCH	
	DMKDDR	FBAIN	
FBAIN calls the start I/O routine for overlapped I/O. The SIO is issued. When cc=0, control is passed to the caller (FBAIN). FBAIN passes control to QSEARCH. QSEARCH waits because there is no work; the output tape has no work and the input DASD is busy. When the FB-512 device interrupts, the I/O interruption routine goes to the interruption return address (FINIRA). Control fields in the DDR are updated in preparation for the next input operation. The fields are: INBLADD – the block number of the next block to be read from tape. OUTBLADD – the block number to which INBLADD should be written.	DMKDDR	QSEARCH	
	DMKDDR	FINIRA	
2 If the blocks just read completed an extent, another field (CUREXT) is updated to point to the next entry in the extent table. If there are no more blocks, return is to ENQBUF. This routes the now filled buffer to the output queue, but does not return INIOB to the input queue. This temporarily precludes further input.			

Notes	Module	Label	Ref
The IOB address (R15) is stored in the appropriate queue. The INIOB address is stored in the DASD or input queue (Q1). Then the THR address (IOBTHR) is stored in the appropriate buffer list. The address of buffer A is stored in the output's list (B2) to signal that a buffer is ready to be output. Both IOBs are available, and a buffer is also ready on both queues. Because the tape queue is inspected first, FBAIN gets control to read into buffer B.	DMKDDR	ENQIOB	
	DMKDDR	QSEARCH	
	DMKDDR	FBAIN	
A CCW string is built to write the THR. The write CCWs are for 4K each (the last one is short) and are command chained together. STARTIOO (the overlapped entry) is called to do the I/O. When the condition code from the SIO is 0, control is returned to the caller (TAPOUT). Then control is immediately passed to QSEARCH. QSEARCH selects FBAIN. The SIO is issued and, when cc=0, QSEARCH is reentered. QSEARCH waits because there is no work to do.	DMKDDR	TAPOUT	
	DMKDDR	STARTIOO	
	DMKDDR	QSEARCH	
	DMKDDR	FBAIN	
	DMKDDR	QSEARCH	
3 The FB-512 device interrupts. The I/O interruption routine goes to FINIRA. Control fields are updated as before and the IOB and the buffers are enqueued as before. QSEARCH waits for work. The tape interrupts and the I/O interruption routine goes to IOBIRA (CHKEOE).	DMKDDR	FINIRA	
	DMKDDR		
	DMKDDR	ENQIOB	
	DMKDDR	QSEARCH	

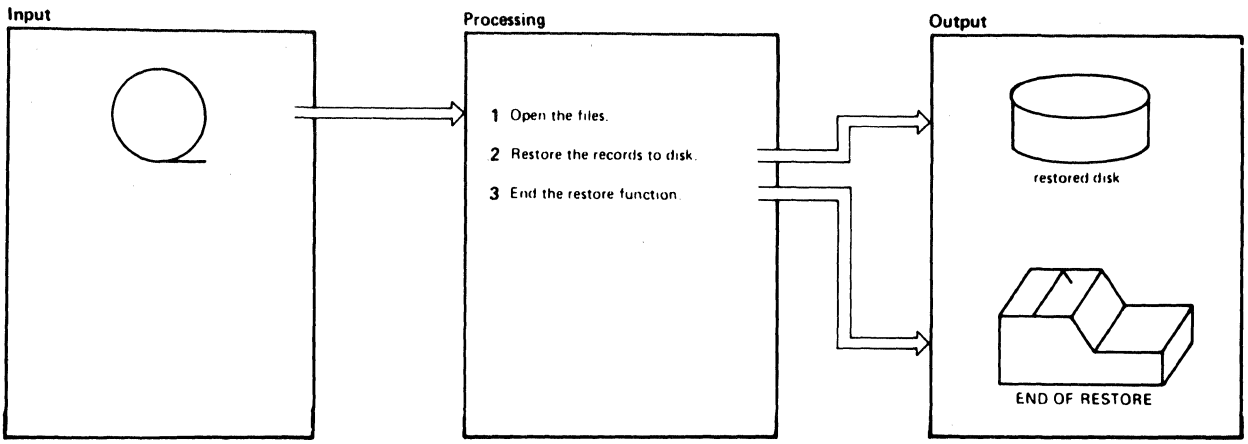
Diagram 6-4. The Dump Function with Streaming (Part 1 of 2)



Notes	Module	Label	Ref
4 CHKEOE determines if the THR just output is the last one to be output in a given extent. If not, return is to ENQIOB. If it is, a call is made to PRINTEXT to print the extent map. If this is the last THR extent, control is passed to the EOJ routine. Otherwise, control passes to ENQIOB. The tape IOB is enqueued and the buffer is routed to the output queue. Both input and output routines are eligible. Because the tape queue is inspected first, TAPOUT is given control. Steps 1 through 3 are repeated.	DMKDDR	CHKEOE	
	DMKDDR	ENQIOB	
	DMKDDR	QSEARCH	
5 The cycle continues until the extents are exhausted. Then the input is turned off. At the completion of the next output, control is passed to the EOJ routine.			

Notes	Module	Label	Ref

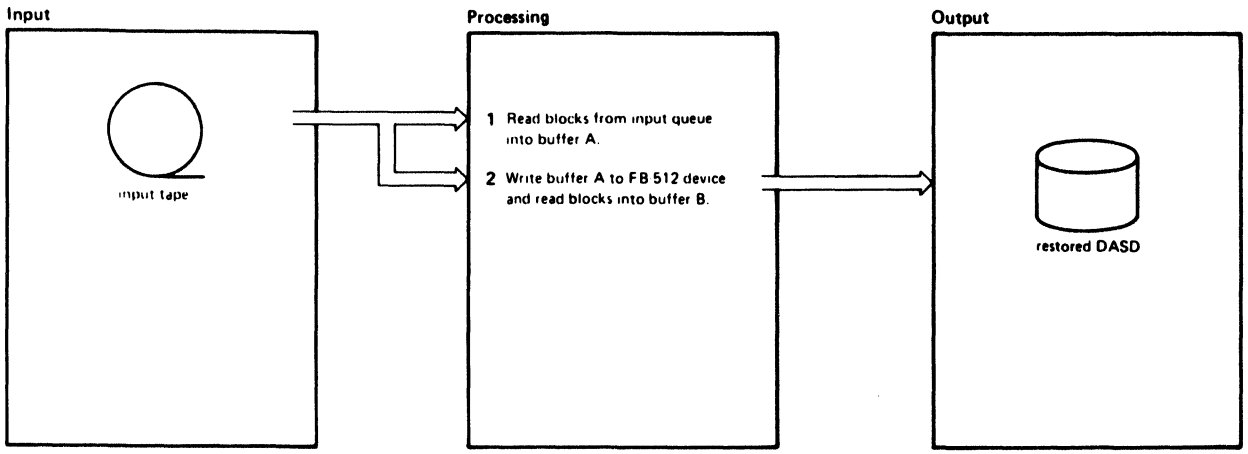
Diagram 6-4. The Dump Function with Streaming (Part 2 of 2)



Notes	Module	Label	Ref
1 The input tape is opened and positioned if the RESTORE control statement specified that records were to be skipped.	DMKDDR	OPENIN	
A check is made to ensure that the output disk has the correct volume serial number. If the volume serial number is incorrect, the message DMKDDR717R DATA DUMP FROM xxxxxx TO BE RESTORED TO xxxxxx is displayed. The operator must decide if the restore function is to continue.		SETDASD	
The extent table is updated to indicate the extents to be restored to disk.		GETEXT	
The output disk is opened by branching and linking to the OPENDASD routine.		OPENOUT	
2 The headings are printed, indicating that the restore function is starting.	DMKDDR	PRINTD MSG004	
The number of cylinders or blocks on the original DASD input device is compared with the number of cylinders or blocks on the DASD output device. If the input device was larger, the following message is issued: DMKDDR725R ORIGINAL INPUT DEVICE WAS(IS) LARGER THAN OUTPUT DEVICE			

Notes	Module	Label	Ref
The operator must determine if the restore function is to continue.			
The read and write loop continues until all the specified extents are restored to disk. The tape records are read from the tape that has been positioned. A check is made to see if the tape is in compact format. If it is, the data is decoded. If there is an error during decoding, the message DMKDDR728E DECODE ERROR ENCOUNTERED xx is displayed. 'xx' is the return code from the decoding routine. It can have the following values:		GETTHR	
	DMKDDC DMKDDT		
2 First byte of input is 0 or is greater than 5. This should not occur. It may be caused by using a set of encoding tables which do not match the decoding tables which are supplied.			
3 There is more data to be decoded, but the output buffer is not big enough to hold more. Decoding stopped when the output buffer became full.			
4 The decoding tables are malformed or the data in compact format was incorrectly transmitted. The program tried to decode a codeword which could not be decoded within its first 21 bits.			
The data is written on the indicated disk cylinders or blocks and the pointers to the disk are updated for the next record. The restore function is complete when the last cylinder (CYLSTOP) or block (BLKSTOP) is restored.	DMKDDR	DASDWRIT UPDTADD	
3 The message END OF RESTORE is displayed. If the data was decoded from compact format, the following message is displayed: BYTES RESTORED_____ Control returns to the GTCARD routine to read the next control statement.	DMKDDR	CLOSEJOB CCMP200	

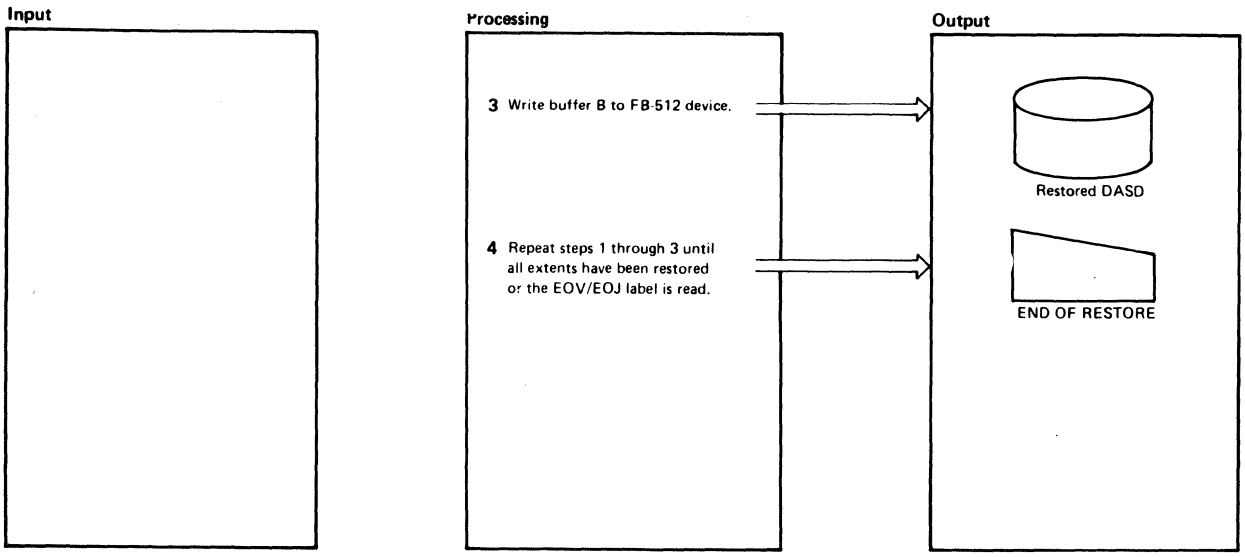
Diagram 6-5. The Restore Function



Notes	Module	Label	Ref
1 The work queues are initialized at OPEN time so that both buffers are on the tape input queue and both INIOB and OUTIOB are also in their respective queues. Therefore, QSEARCH selects TAPIN because the tape queue is searched first. TAPIN prepares to read the track header record tape block. It builds a CCW to read a 4K block and calls STARTIO (the unoverlapped entry point) to do the SIO. Because entry was to the unoverlapped entry of the start I/O routine, control is not returned until the I/O is complete. When the 4K tape record containing the THR record has been read, control returns to TAPIN. TAPIN uses the control data in the THR to construct a CCW chain to read the rest of the 4K tape records, plus the last short record. TAPIN calls STARTIOO (the overlapped I/O entry point) to do the SIO. An SIO is issued. Because entry was to the overlapped entry point, control is returned when the condition code returned from the SIO is 0. The tape is now filling buffer A. TAPIN routes control directly to QSEARCH.	DMKDDR DMKDDR DMKDDR DMKDDR	QSEARCH TAPIN TAPIN STARTIOO	
The remaining buffer (B) is still on the tape's input queue. However, the tape's IOB (INIOB) is not on the queue. Therefore, QSEARCH cannot select the tape input routine. Because no buffers are on the DASD or output queue, the output routine is also ineligible for selection. QSEARCH then loads an enabled wait PSW.	DMKDDR	QSEARCH	
When the tape completes the I/O interruption routine finds the owning IOB. In the IOB is the interruption return address (IOBIRA). When this routine is entered, register 15 equals the address of the IOB. TAPIRA now ensures that the FB-512 block number in question (INBLADD) is in the data just read. If not, the TAPIN routine is reentered. If INBLADD is in the THR, the control fields in the THR are initialized in preparation for routing the buffer to the output routine. The fields are	DMKDDR	TAPIRA	

Notes	Module	Label	Ref
THRWROTE – describes how may blocks in the buffer should be written. THRFRSBL – the first block in the buffer. This block is left-justified in the buffer THROBLAD – the block number on FB-512 where writing should begin. Next, control fields in the DDR are updated in preparation for the next input operation. These fields are INBLADD – the block number of the next block to be read from tape. OUTBLADD – the block number into which INBLADD should be written 2 If the blocks just read completed an extent, another field (CUREXT) is updated to point to the next entry in the extent table. If there are more blocks to process, return is to ENQIOB. If there are no more blocks to process, return to ENQBUF. This routes the now filled buffer to the output queue, but does not return INIOB to the input queue. This temporarily precludes further input			
The IOB address (R15) is stored in the appropriate queue. The INIOB address is stored in the tape or input queue (Q1). Then the THR address (IOBTHR) is stored in the appropriate buffer list. The address of buffer A is stored in the output's list (B2) to signal that a buffer is ready to be output.	DMKDDR	ENQIOB	
A channel program is built to read the next tape record. An SIO is issued and TAPIN builds the channel program to read the rest of the THR data. Unoverlapped SIO is issued.	DMKDDR	TAPIN	

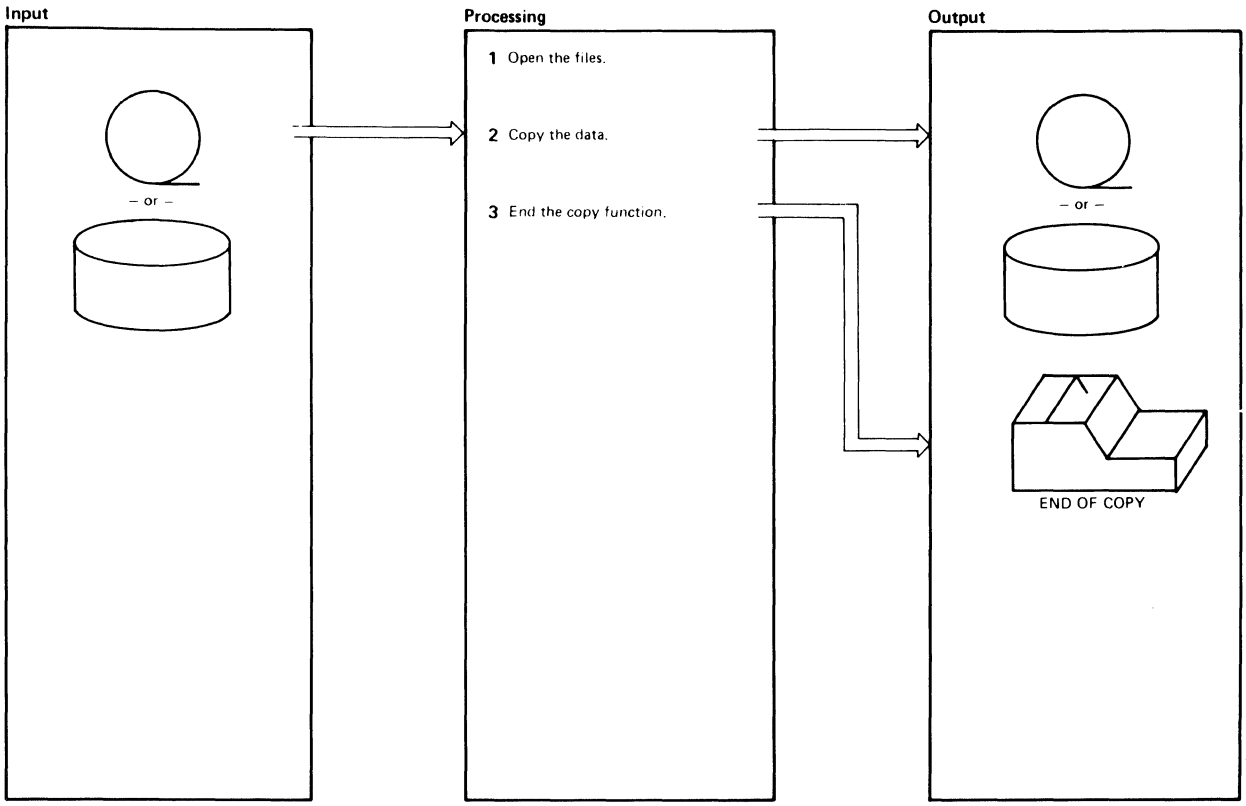
Diagram 6-6. The Restore Function with Streaming (Part 1 of 2)



Notes	Module	Label	Ref
3 The tape input routine cannot be selected because the IOB is not on the queue. However, the fixed block output routine (FBAOUT) is selected to output buffer B. The IOB and buffer are dequeued and FBAOUT is entered.	DMKDDR	QSEARCH	
Using the control data in the THR, a channel program is built to write the blocks. THROBLAD tells what block to LOCATE, and THRWROTE tells how many blocks to write. The overlapped entry to the start I/O routine is called. An SIO to DASD is issued. When cc=0, control is returned to the caller (FBAOUT). FBAOUT passes control to QSEARCH immediately.	DMKDDR	FBAOUT	
Neither queue contains an available IOB address of buffer address. Both I/O devices are busy. An enabled wait PSW is loaded. Normally, the DASD finishes before the tape. The I/O interruption routine finds the owning IOB (OUTIOB in this case) and passes control to IOBIRA or FOUTIRA.	DMKDDR	QSEARCH	
It is determined if the THR just output is the last one to be output in a given extent. If not, return is to ENQIOB. If it is, a call is made to PRINTEXT to print the extent map. If this is the last THR of the last extent, control passes to the EOJ routine. Otherwise, control is passed to ENQIOB. The IOB (OUTIOB) is enqueued to the appropriate queue (Q2), and the buffer (B) to the appropriate queue (Q1 the tape input queue). Control passes to QSEARCH.	DMKDDR	FOUTIRA	
	DMKDDR	ENQIOB	
The tape is still busy (INTIOB is not on a queue) and there is no work for DASD output. A wait PSW is loaded. The tape interrupts after buffer B is read. IOBIRA is given control as before.	DMKDDR	QSEARCH	

Notes	Module	Label	Ref
4 Steps 1 through 3 are repeated. The cycle continues until all extents have been restored, or an EOV or EOJ label is read. If EOV, the I/O is interrupted while the tape switch occurs. Then the cycles begin again with TAPIN building a channel program to read the next tape record. If EOJ is read, the EOJ routine is called. In both cases, the output DASD is allowed to finish its queued work before tape processing (EOV or EOJ) is resumed.			

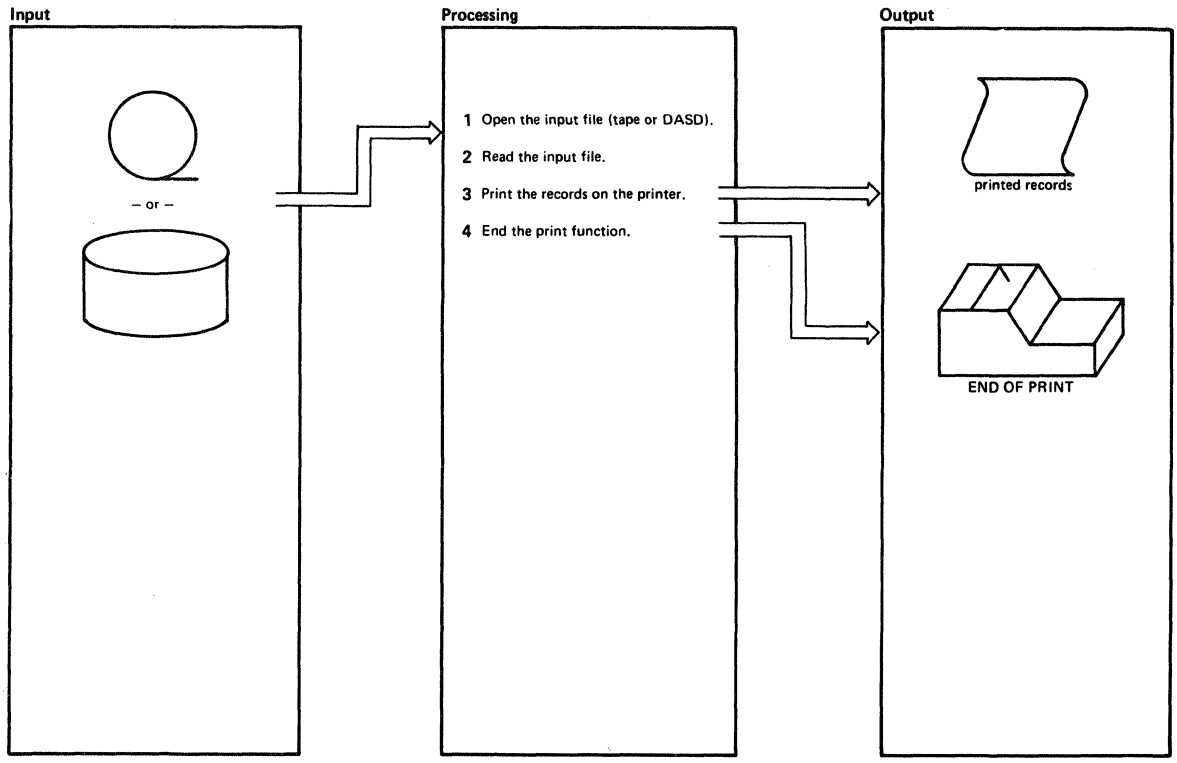
Diagram 6-6. The Restore Function with Streaming (Part 2 of 2)



Notes	Module	Label	Ref
1 The input file and output file are opened. The input and output devices must be the same device type. It is allowable if both DASD devices are FB 512, regardless of the particular type. The extent table is updated to reflect the amount of data to be copied from one device to another.	DMKDDR	OPENIN GETEXT OPENOUT	
2 The heading is written and the message indicating the start of the copy function is typed. The input file is read and the output file is written. If copying from disk to disk the pointers to the disk records are updated to the next record. The read write cycle continues until the specified data is copied. When copying data from tape to tape, the GETTHR routine performs the record read and the TESTOUT routine performs the record write. When copying data from disk to disk, the BUILDTHR routine performs the record read and the DASDWRIT routine performs the record write. If the COMPACT option was specified, the following message is displayed: DMKDDR731I COMPACT OPTION IS IGNORED FOR COPY OPERATIONS	DMKDDR	PRINTH MSG004 UPDTADD GETTHR TESTOUT DCMP500 BUILDTHR DASDWRIT SCANCOPY	

Notes	Module	Label	Ref
3 The message END OF COPY indicates the successful completion of the copy function. When copying data from tape to tape, the output tape is positioned as indicated on the COPY control card. When the disk to disk copy is complete, the disk is closed. Control returns to the GTCARD routine to read the next control statement.	DMKDDR	CLOSEJOB	

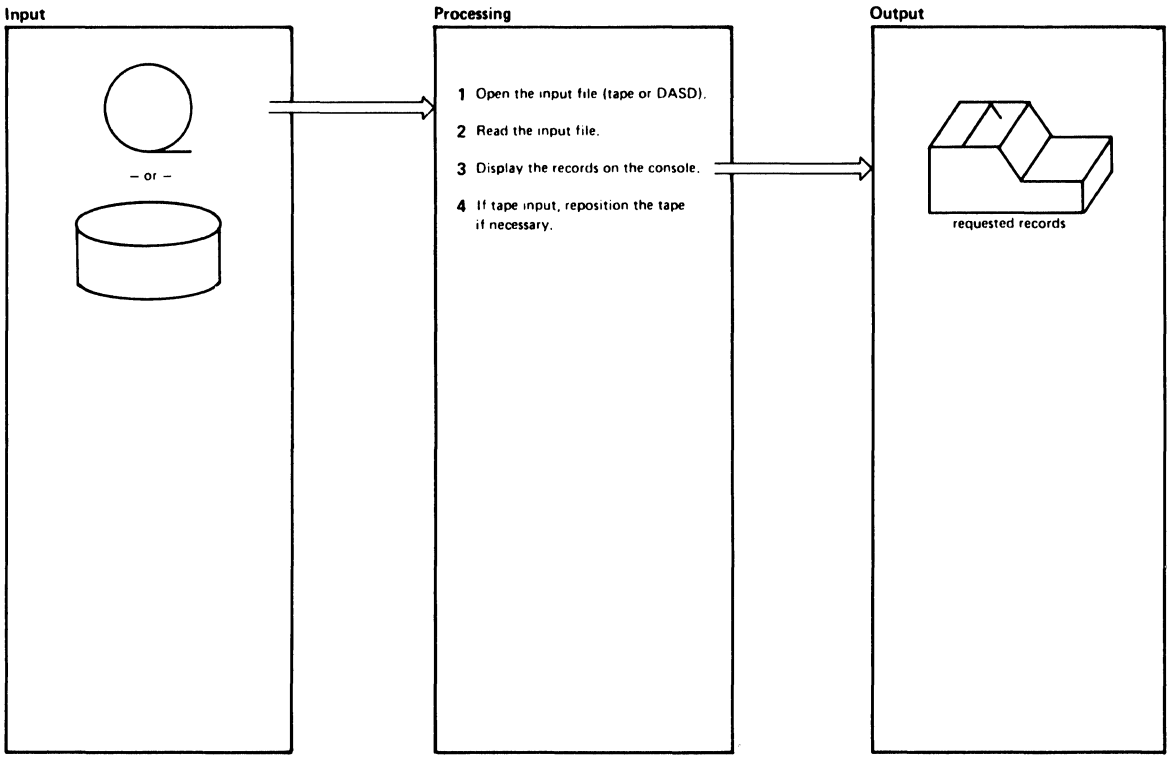
Diagram 6-7. The Copy Function



Notes	Module	Label	Ref
1 The input device is opened. If the input is on tape, the tape is spaced forward the designated number of records (if any). The extent table is updated to reflect the cylinders to be printed.	DMKDDR	OPENIN GETEXT	
2 The message PRINTING xxxxxxxx is displayed to indicate the start of the PRINT function.	DMKDDR	MSG004	
3 The data is read from the input device via the appropriate (disk or tape) read routine. The data is converted, de-coded from compact format, if needed, and printed on the system printer.	DMKDDC DMKDDT DMKDDR	GETTHR DCMP500 DISPLAY	
4 The message END OF PRINT indicates the successful completion of the PRINT function.	DMKDDR	EOJ	
5 Control returns to the GTCARD routine to read the next control statement.			

Notes	Module	Label	Ref

Diagram 6-8. The Print Function



Notes	Module	Label	Ref
1 The input device (either tape or disk) is opened. If input is on tape, the tape is spaced forward the designated number of records (if any). The extent table is updated to reflect the data to be typed.	DMKDDR	OPENIN GETEXT	
2 The records are read from the tape or disk by the appropriate read routine and decoded from compact format, if needed.	DMKDDR DMKDDC DMKDDT	BUILDTHR GETTHR DCMP500	
3 The records are displayed on the console. The read and type cycle is continued until all the specified records are typed.	DMKDDR	DISPLAY	
4 Control returns to the GTCARD routine to read the next control statement.	DMKDDR	EOJ	

Notes	Module	Label	Ref

Diagram 6-9. The Type Function

Program Organization

This section contains a program description of the DMKDDR module.

DMKDDR

The DASD dump restore program.

Attributes

Serially reusable.

Entry Point

DMKDDREP.

Registers at entry

R1: Points to a parameter list when DMKDDR is executed under the control of CMS.

Registers at exit

R15: Contains a return code when DMKDDR is executed under the control of CMS. The return codes are:

Return Code	Meaning
1	Invalid filename or file not found.
2	Error while running the program.
3	Flagged DASD track.
4	Permanent tape or DASD I/O error.
1xx	Error in the PRINTIO routine.
2xx	Error in the CONREAD routine.
3xx	Error in the RDBUF routine.
4xx	Error in the TYPLIN routine.
20	Error in the decoding routine.

where:

xx is the return code from the CMS routine.

Register Usage

R0: Work Register.
R1: Pointer to input field from SCANCONT.
Pointer to the output buffer (PRINT/TYPE).
Work register.
R2: Input count from SCANCONT.
Unit address for STARTIO.
Data block count (PRINT/TYPE).
Work register.
R3: End of current line (PRINT/TYPE).
Work register.
R4: Length of one line (PRINT/TYPE).
Pointer to key (PRINT/TYPE).
Work register.
R5: Total length of data (PRINT/TYPE).
Work register.
R6: Data count (PRINT/TYPE).
Number of records on the track (PRINT/TYPE).
Work register.
R7: Pointer to the extent table entry.
Current line pointer (PRINT/TYPE).
R8: Address of the data area used for DASD/tape input and output (THR).
R9: Base register 5.
R10: Base register 1.
R11: Base register 2.
R12: Base register 3.
R13: Base register 4.
R14: Return address.
R15: Pointer to the IOB.

External References

DMSACF, DMSCRD, DMSCWR, DMKDDC (Data decoding), DMKDDT (Decoding table), DMKDNC (Data encoding), DMKDNT (Encoding table).

Directory

Figure 6-2 is an alphabetic list of the major labels in the DASD Dump Restore program. The associated method of operation diagrams are indicated and a brief description is included of the operation performed at the point in the program that is associated with each label.

Label	Diagram	Description
ADDLINE		Checks for duplicate line.
ALL		Handles the ALL parameter.
ALLSET		Prepares to type or print data specified.
ALTTRACK		After errors, handles alternate tracks.
ALTXDEF		Handles defective alternate tracks.
BINCONV		Converts decimal numbers to binary.
BLDFBATH		Builds THR for FBA devices.
BLNKVSER		Clears the volume serial number on disk.
BSFILE		Backspaces if overran VHR on tape.
BUILDCCW		Builds a CCW string to put the key/data fields into the THR (track header record.)
BUILDIOB		Creates the IOBLOK for DDR.
BUILDTHR	6-3 6-7 6-8 6-9	Reads records from disk.
CCMP200	6-3 3-5	Subroutine prints encoding/decoding statistics.
CCMP300	6-3	Subroutine encodes data into compact format before writing to tape.
CHKCOPY		Initiates double buffering for the copy function.
CHKEOE	6-4	Checks if output was for last of an extent. If yes, prints the extent map.
CHKLOW		Insures nucleus starts in permanent space for FBA.
CHKSAME		Checks for same device type.
CHKSAM1		Checks for same FBA device type.
CHKSIZE		Checks incorrect length errors.
CHKTORE		Initiates double buffering for the restore function.
CHKTYPE		Sees if user and CP agree.
CKDEOV		Handles EOV conditions for CKD devices.
CKDVHRIN		Initializes VHR for CKD devices.
CKDVOL		Reads volid from DASD.
CKEXT		Checks for ECKD.
CLOSEJOB	6-3 6-5 6-7	Displays message indicating the end of a DDR function.
CLOSE1		Closes the tape and reads another.
CLOSIT		Ends a job step.
CLOSJ		Displays message indicating the end of a DDR function.
CMPCPY		Skips if COMPACT not specified for the copy function.
CMPPEND		Bypasses statistics for encoding routines.

Figure 6-2 (Part 1 of 7). The DASD Dump Restore Program Label Directory

Label	Diagram	Description
CMSA		Handles console output.
CMS1	6-1	Builds a PLIST (parameter list) if parameters passed from CMS.
CMS8	6-1	The end-of-job processing when DDR is running under VM/SP.
COMPARE		Compares keywords.
COMPBLK		Checks the addresses for FBA devices.
COMPCYL		Checks the home addresses for CKD devices.
COMPSIZ		Compares relative sizes of devices for the copy function.
CONERROR		Handles console errors.
CONRET		Returns here after console wait.
CONSOUT2		Handles sysprint console output.
CONTSCAN		Gets extent table for the type and print functions.
CORRCSW		Handles imprecise ending conditions.
CPVOL		Handles all active DRCT and PERM space.
CSWSTORE		Checks CSW after a SIO.
CYLSETOK		Sets up output cylinder id.
DASDRCVR		Gets the address of the CCW chain to write to DASD.
DASDWRIT	6-5 6-7	Writes records onto disk.
DATACHK1		Checks for read multiple CKD.
DATACHK2		Checks for permanent errors.
DATACOR1		Corrects ECC correctable errors and restarts the chain.
DCMP500	6-5 6-7 6-8 6-9	Subroutine invokes decompacting during reads.
DDR536		Issues DMKDDR536I message.
DDR700		Issues DMKDDR700E message.
DDR701		Issues DMKDDR701E message.
DDR702		Issues DMKDDR702E message.
DDR703		Issues DMKDDR703E message.
DDR704		Issues DMKDDR704E message.
DDR705		Issues DMKDDR705E message.
DDR707		Issues DMKDDR707E message.
DDR708		Issues DMKDDR708E message.
DDR709		Issues DMKDDR709E message.
DDR710		Issues DMKDDR710A message.
DDR711		Issues DMKDDR711R message.
DDR712		Issues DMKDDR712E message.
DDR713		Issues DMKDDR713E message.
DDR714		Issues DMKDDR714E message.
DDR715		Issues DMKDDR715E message.
DDR716		Issues DMKDDR716A message.
DDR717		Issues DMKDDR717R message.
DDR718		Issues DMKDDR718E message.
DDR719		Issues DMKDDR719E message.
DDR720		Issues DMKDDR720E message.
DDR721		Issues DMKDDR721E message.
DDR722		Issues DMKDDR722E message.
DDR723		Issues DMKDDR723E message.
DDR724		Issues DMKDDR724E message.
DDR725		Issues DMKDDR725R message.

Figure 6-2 (Part 2 of 7). The DASD Dump Restore Program Label Directory

Label	Diagram	Description
DDR726		Issues DMKDDR726E message.
DDR727		Issues DMKDDR727E message.
DDR728		Issues DMKDDR728E message.
DDR729		Issues DMKDDR729I message.
DDR731		Issues DMKDDR731I message.
DDR756		Issues DMKDDR756E message.
DECCONV		Converts decimal numbers to hexadecimal.
DEFTRACK		Handles defective tracks.
DEVICOK		Device is waiting, start the I/O.
DEVTEST		Tests the device types for the dump function.
DEVTST		Tests the device types for the restore function.
DISPFT3		Handles FTR format before the display function.
DISPIT		Displays the key/data message.
DISPLAY	6-8 6-9	Prints or types records.
DISPR0		Prints record 0.
DMKDDR		Start of the DMKDDR module.
DMKDDREP	6-1	Entry point to the DDR program.
DODIV		For FBA, finds the number of blocks/track.
DOTAPIO		Issues SIO to write to tape.
DOTAPIO2		Reads in the FTR records.
ENQIOB	6-4 6-6	Moves a ready buffer to the queue for processing.
EOJ	6-3 6-8 6-9	At the end of a DDR function, returns control to the GTCARD routine.
ERRCLOSE		Closes tape and reads alternate tape.
EXIT	6-1	Returns to CMS command environment or enters wait state at end of program.
EXTENTIN		Gets the cylinder extents for DASD.
EXTINT		Handles external interrupts.
FADD1		For FBA, converts allocation map to blocks.
FBACPVOL		Reads in the allocation extent map.
FBADISP		Displays FBA blocks.
FBAEOV		Handles EOV conditions for FBA devices.
FBAERR		Checks FBA device errors.
FBAEXTS		Builds the extent table for FBAs.
FBAIN	6-4	Calls STARTIOO to read the FB-512 device.
FBALLC		Handles nucleus formatting on FBA devices.
FBAOUT	6-6	Calls STARTIOO to write for FB-512 device.
FBARDCIO		RDC command - gets 32 bytes of device characteristics.
FBASTOP		Gets stop block for the type and print functions.
FBAUPADD		Updates pointers to next set of FBA blocks.
FBAWRIT		Writes blocks to FBA volumes.
FINIRA	6-4	Calls CHKEOE to see if an extent has just been finished.
FOUNDIT		Finds the error address.
FOUTIRA	6-6	Updates for the next input operation.
FTRDOK		Prepares to write out the THR.
FTREAD		Reads in FTR records for DASD.
GETCCHH		Returns CCHH address from sense data.
GETCPTYP		Sees if user and CP agree about device type.

Figure 6-2 (Part 3 of 7). The DASD Dump Restore Program Label Directory

Label	Diagram	Description
GETCSW		Picks up the error CSW command address.
GETDASD		Sets up for cpvol nucleus dump.
GETEXT	6-3 6-5 6-7 6-8 6-9	Builds extent table.
GETFTREC		Reads the full track records from tape.
GETHAR0		Reads home address and record 6.
GETNEWEX		Gets new extent for FBA.
GETOTHER		Scans for other options on DASD.
GETPARM		Handles tape options.
GETREOR		Tests for reorder parameter for CKD.
GETREOR1		Tests for reorder parameter for FBA.
GETR1		Checks for records that need to be printed.
GETSTART		Gets the next statement.
GETTHR	6-5 6-7 6-8 6-9	Reads tape records.
GETVHR		For tape input, gets VHR.
GETVSER		Opens the DASD unit.
GOODNAME		Finds a valid name and returns.
GOSUB1		Gets the next record.
GOTTHR		Finds the THR.
GRAPHID		Handles I/O for display terminals.
GTCARD	6-1	Reads control cards.
GTSCAN		Gets the first field on the card.
HEADOK		Fills in header record fields.
HEXCONV		Converts hexadecimal numbers to decimal.
INOUTER		Handles tape and DASD errors.
IOERROR		Handles I/O errors.
IOWAIT		Enables for I/O interruptions.
LASTONE		Checks for last record.
LOOP5		Scans control statements for next field.
LOOP12		Checks for last record to be displayed.
LOOP13		Determines the starting address.
MARKOPEN		Marks the IOB as open for the device.
MSGRET		After message is written, scans the next line.
MSGWRITE		Displays messages on the terminal.
MSG001		Writes 'END OF xxxxx' message.
MSG002		Writes header message.
MSG003		Writes 'ENTER EXTENTS' message.
MSG004	6-3 6-5 6-7 6-8	Prints message indicating start of Dump, Restore, Copy, or Print function.
MSG005		Writes 'MOUNT NEXT TAPE' MESSAGE.
MSG006		Writes 'MOUNT NEXT TAPE' MESSAGE.

Figure 6-2 (Part 4 of 7). The DASD Dump Restore Program Label Directory

Label	Diagram	Description
NEWADD	6-1	Prints heading when DDR program running standalone.
NEXTCYL		Updates pointer to next cylinder.
NEXTREC		Updates pointer to next record.
NEXTTCK		Updates pointer to next track.
NORECFND		Handles a 'NO RECORD FOUND' check.
NOSTART		Sets up starting address for DMKDDR721E message.
NOTCONS1		Ignores CONS option if not under CMS.
NOTFTRD		Does normal read processing.
OK		Points to read CCWs to read THR.
OPENCNT		Handles streaming for tapes.
OPENDASD		Opens a DASD.
OPENER		Handles user responses.
OPENIN	6-3 6-5 6-7 6-8 6-9	Opens input devices.
OPENOUT	6-3 6-5 6-7	Opens output devices.
PBUFFER		Points to the print buffer.
PCOUNT		Converts all addresses and data to decimal.
PDATA		Sets up print pointer.
PRINTBUF		Prepares to write out error message.
PRINTDAT		Prints the data.
PRINTER1		Updates the printer line count.
PRINTER2		Spaces the printer twice.
PRINTEXT		Handles printer output.
PRINTH	6-3 6-7	Prints function heading.
PRINTIT		Prints the header message.
PRINT1		Checks that device type is console.
PRINT2		Displays message on console.
PRTINIT		Initializes the printer.
PUBLKUP		Sees if there is a device waiting.
QSEARCH	6-4 6-6	Looks for work on queues. If found, dequeues IOB and a buffer and enters I/O routine.
QUIESCE		Handles queued work for double buffering.
READCKP		Reads nucleus.
READCONT		Reads control statements.
READCT		Reads the home address, record 0, and the count fields.
READKEYD		Reads the key and data records.
READTAPE		Reads in the records.
READ66		Reads data from graphics devices.
REORBLOK		Reorders FBA blocks for output.
REORCYL		Writes out the THR.
REPOTAPE		Repositions the tape after an error.
RESPONSE		Handles user responses from DDR questions.
RESTART		Restarts the I/O.
RETRY		Sets up for retry of the SIO.
RSPCPY		Issues responses for the copy function.

Figure 6-2 (Part 5 of 7). The DASD Dump Restore Program Label Directory

Label	Diagram	Description
SAVECT		Saves the printer line count.
SCANCONT		Scans control statements for next operand.
SCANCOPY	6-2 6-7	Scans the COPY function statement.
SCANDATA		Scans control statements for special characters.
SCANDUMP	6-2	Scans the DUMP function statement.
SCANFBA		Scans for starting and ending blocks for FBA type and print functions.
SCANFTR		Scans for FULL TRACK READ option.
SCANINPU	6-2	Scans the INPUT control statement.
SCANLEAV		Scans for LEAVE option.
SCANMODE		Scans for MODE option.
SCANNAME		Scans the name table (TABLE1) for a matching control statement name.
SCANOUTP	6-2	Scans the OUTPUT control statement.
SCANPRIN	6-2	Scans the PRINT function statement.
SCANREST	6-2	Scans the RESTORE function statement.
SCANSKIP		Scans for SKIP option.
SCANSYSP	6-2	Scans the SYSPRINT control statement.
SCANTYPE	6-2	Scans the TYPE function statement.
SCANUNIT		Scans the device table (TABLE2).
SCANUNLO		Scans for UNLOAD option.
SCRATCH		For scratch volser, skips label verification.
SENSIO		Does a sense on the device.
SETDASD SETEND	6-5	Checks volume serial number of output disk. Prints the cylinder map at end-of-job.
SETEXT		Picks up the cylinder number that starts the next extent.
SETEXT1		Picks up the block number that starts the next extent.
SETMK		Builds CCW chain for reading the records.
SETQ		Initiates double buffering.
SETSTOP		Gets stop cylinder for the type and print functions.
SETUPADD		Sets up input and output addresses.
SETUPBUF		Clears the print buffer.
SETUPERR		Handles errors writing to the console.
SETVSN		Sets up volume serial number.
SET4K		Reads in Non-FTR records.
SKIPMSG		Prints record overflow message.
STARTIO		Starts I/O devices.
STARTIOO	6-4 6-6	Issues SIO.
STMSHRT1		Issues CCW for writing short records.
STOREADD		Starts here after the first read.
SUPMSG		Prints the suppress line message.
TABLE1		Generated name/function table.
TABLE2		Generated unit/device table.
TAPE		Checks for an alternate tape device address.
TAPEER		Checks tape device errors.

Figure 6-2 (Part 6 of 7). The DASD Dump Restore Program Label Directory

Label	Diagram	Description
TAPIN	6-6	Reads the THR and calls STARTIOO to read the remaining tape records.
TAPIRA	6-6	Updates DDR in preparation for the next input operation.
TAPNCMP		Checks for FTR output format.
TAPOUT	6-4	Writes THR to tape.
TAPWRIT		Prepares to write to the tape.
TESTALLF		Tests for the 'RESTORE ALL' function.
TESTCARD		Checks for card input at end-of-job.
TESTCMS	6-1	Exits by entering wait state when DDR program is running standalone.
TESTCOMR		Tests for command reject.
TESTDACK		Tests for data checks.
TESTDASD		Tests for DASD in printer routine.
TESTDEV		Tests device status after SIO.
TESTEND		Terminates when blank card read.
TESTFLAG		Checks for NUCLEUS option.
TESTGRAP		Tests for a graphics device.
TESTIN		Checks for tape input.
TESTINF		Checks input block numbers.
TESTIO		Does a TEST I/O on the device.
TESTMD		Tests for FTR errors.
TESTNPAG		Skips printer to channel 1.
TESTOPT		Handles the type and print function options after left parenthesis.
TESTOUT	6-3 6-7	Writes tape output records.
TESTPERM		Tests for permanent allocated space.
TEST800		Tests for 800 MODE option.
TEST1600		Tests for 1600 MODE option.
TEST3278		Tests for a 3278 device.
TES3270T		Tests for a 3270 device.
TPSWP		Closes the old tape and opens the next tape.
TRANS		Translates data to printable characters.
TRKCOND		Recovery procedure for track condition check (alternate track).
TSTCOUNT		Prints the end of the track.
TSTDEV		Tests the device types for the copy function.
TSTEXT		Tests the extents for output devices.
TSTINPUT		Common code for the type and print functions when opening device.
UNITCHK		Handles unit checks.
UPDTADD	6-3 6-5 6-7	Updates disk addresses.
UPDTEXT		Restores entire track.
UPDTEXT1		Restores blocks for FBA devices.
USENBLK		Initializes THR fields.
VALEXT		Validates the cylinder or block number.
WCKDSET		Sets up extended CCWS.
WDSIO		Writes the THR (track header record).
WRITENUC		Handles transferring of nucleus.
WRTSFMT		Handles overflow records.
WRT66		Writes data to graphics devices.
WTDASD		Writes THR to DASD.
YEARSET		Calculates the Greenwich Mean Time.

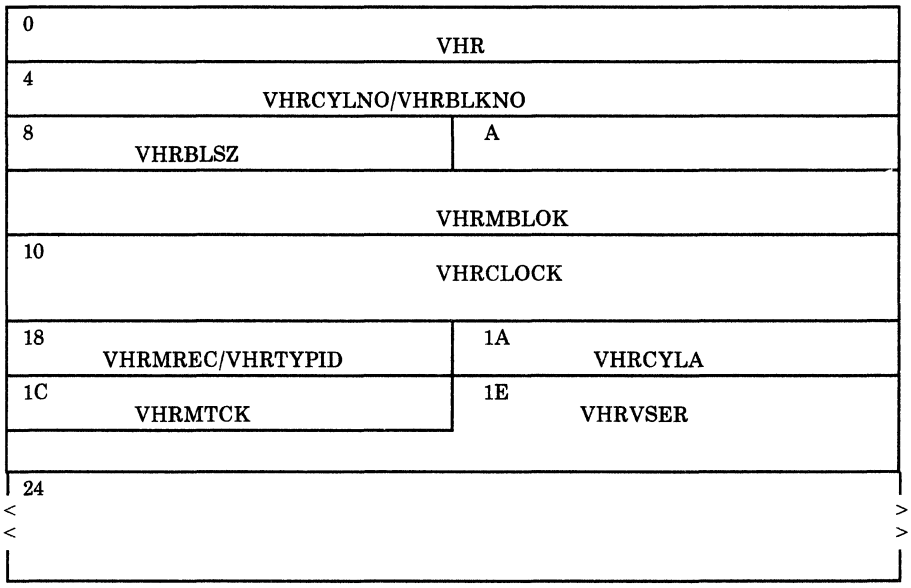
Figure 6-2 (Part 7 of 7). The DASD Dump Restore Program Label Directory

Data Areas

This section contains a description of a:

- Track header record
- Cylinder header record
- IOB
- Trace table

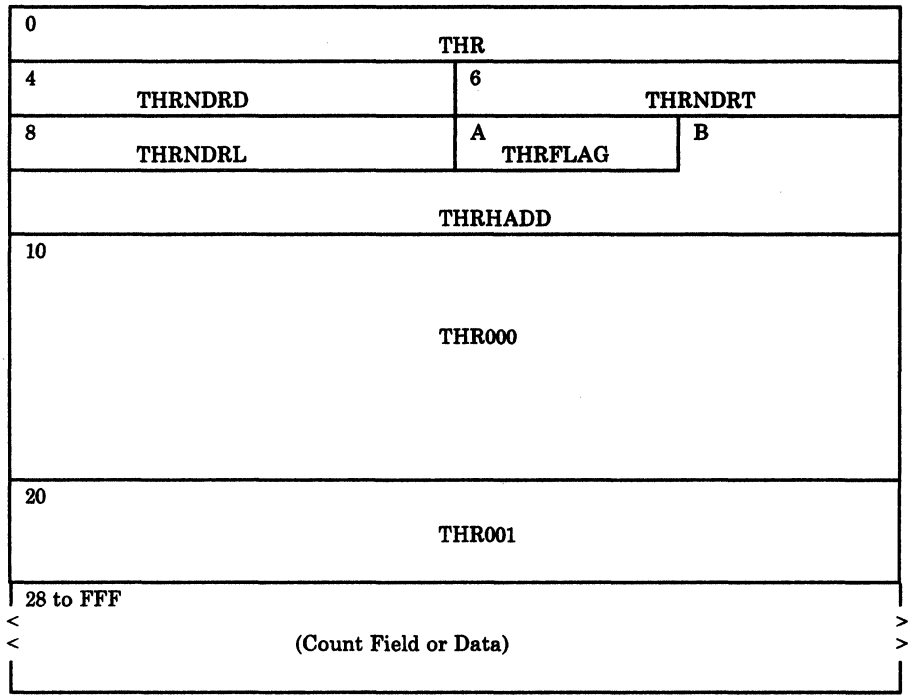
Cylinder Header Record



Displacement Hex	Dec	Field Name		Description	
0	0	VHR	DC	CL4'VHR'	
4	4	VHRCYLNO	DS	CL6'0'	BBCCHH of input DASD unit
4	4	VHRBLKNO	DS	F'0'	FB-512 block number of first block
8	8	VHRBLSZ	DS	H	Block size of originating device
			DS	H	Not used (for alignment)
A	10		DS	XL6' '	Not used if count-key-data
C	12	VHRMBLOK	DS	F	Maximum block number of originating device
10	16	VHRCLOCK	DS	D'0'	Time of day clock value
18	24	VHRTYPID	DS	H'0'	Halfword of zeros identifies that this volume contains FB-512 data.
1A	26	VHRCYLA	DS	H'0'	CC address of last cylinder on this type of DASD.
1C	28	VHRMTCK	DS	H'0'	
1E	30	VHRVSER	DS	CL6'VOLSER'	Volume serial number of input DASD unit
24	36		DS	CL44' '	

Figure 6-3. Cylinder Header Record

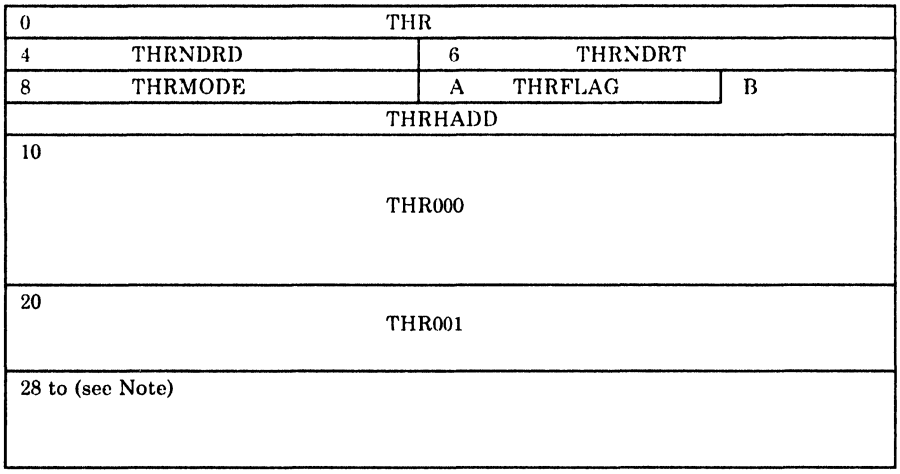
Track Header Record for Count-Key-Data (non-FTR)



Displacement Hex	Dec	Field Name		Description	
0	0	THR	DC	CL4'THR '	ID of track header record
4	4	THRNDRD	DC	H'0'	The number of count fields in the THR
6	6	THRNDRT	DC	H'0'	The number of 4K data records on tape
8	8	THRDRL	DC	H'0'	Length of the short (last) data record
A	10	THRFLAG	DC	XL1'0'	Flag
		Bit settings for THRFLAG			
		SPECIAL	EQU	X'01'	Overflow
B	11	THRHADD	DC	XL5'0'	The home address reordered
10	16	THR000	DC	XL16'0'	Record 0 from the DASD unit
20	32	THR001	DC	XL8'0'	Count field of the first record
28	40				Count fields and data

Figure 6-4. Track Header Record for Count-Key-Data (non-FTR)

Track Header Record For Count-Key Data (FTR)

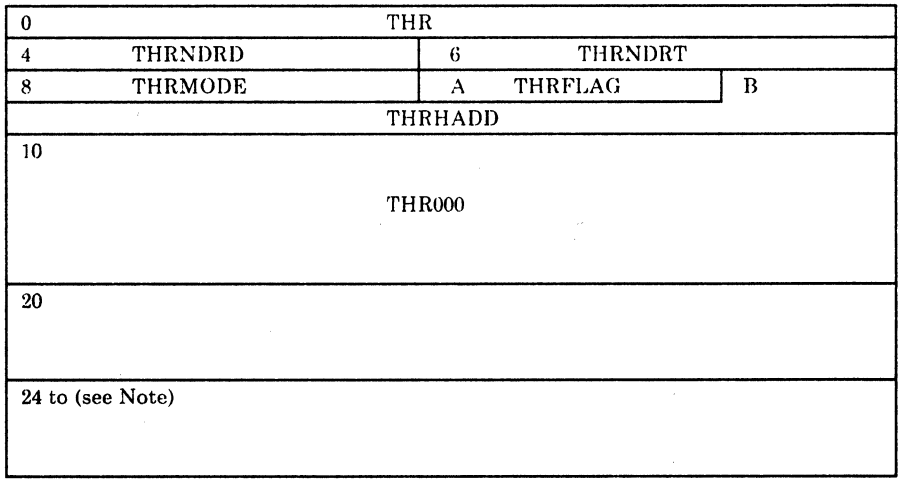


Displacement Hex	Dec	Field Name			Description
0	0	THR	DC	CL4'THR '	ID of track header record
4	4	THRNDRD	DC	H'0'	The number of records in the track
6	6	THRNDRT	DC	H'0'	The track length in bytes
8	8	THRMODE	DC	XL1'0'	bpi setting for tape
		Bit settings for THRMODE			
		MODE6250	EQU	X'00'	For 6250 bpi tape
		MODE1600	EQU	X'01'	For 1600 bpi tape
		MODE800	EQU	C'02'	For 800 bpi tape
A	10	THRFLAG	DC	XL1'0'	Flag for track status
		Bit settings for THRFLAG			
		SPECIAL	EQU	X'01'	Special
		FTRMODE	EQU	C'02'	Header in FTR mode
B	11	THRHADD	DC	XL5'0'	The home address reordered
10	16	THR000	DC	XL16'0'	Record 0 from the DASD unit
20	32	THR001	DC	XL8'0'	Count fields of the first record
28	40				Count-key-data fields

Note: 28 to 1FFF for 800 bpi, 28 to 2FFF for 1600 bpi, and 28 to BFFF for 6250 bpi tape.

Figure 6-5. Track Header Record for Count-Key-Data (FTR)

Track Header Record for Count-Key Data (Compacted, FTR or Non-FTR)

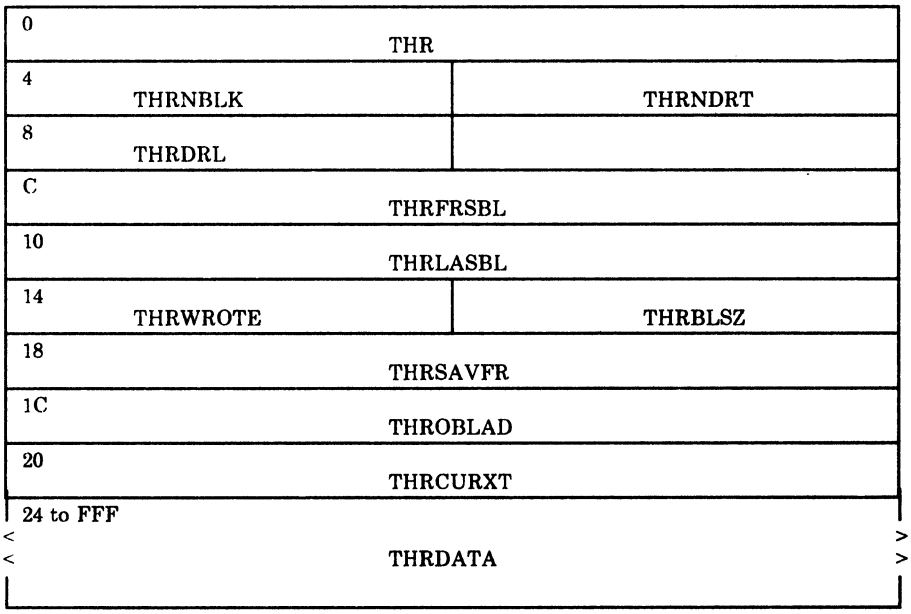


Displacement Hex	Dec	Field Name		Description	
0	0	THR	DC	CL4'THR '	ID of track header record
4	4	THRNDRD	DC	H'0'	The number of records in the track
6	6	THRNDRT	DC	H'0'	The number of compacted bytes
8	8	THRMODE	DC	XL1'0'	bpi setting for tape
		Bit settings for THRMODE			
		MODE6250	EQU	X'00'	For 6250 bpi tape
		MODE1600	EQU	X'01'	For 1600 bpi tape
		MODE800	EQU	C'02'	For 800 bpi tape
A	10	THRFLAG	DC	XL1'0'	Flag for track status
		Bit settings for THRFLAG			
		SPECIAL	EQU	X'01'	Special
		FTRMODE	EQU	X'02'	Header in FTR mode
		CMPCOMP	EQU	X'80'	Encoded data
		CMPLBLK	EQU	X'40'	Compaction flag
B	11	THRHADD	DC	XL5'0'	The home address reordered
10	16	THR000	DC	XL16'0'	Record 0 from the DASD unit
20	32		DS	F	Number of byes of non-compacted data
24	36				Compacted data

Note: 24 to 1FFF for 800 bpi, 24 to 2FFF for 1600 bpi, and 24 to BFFF for 6250 bpi tape.

Figure 6-6. Track Header Record for Count-Key-Data (Compacted, FTR or Non-FTR)

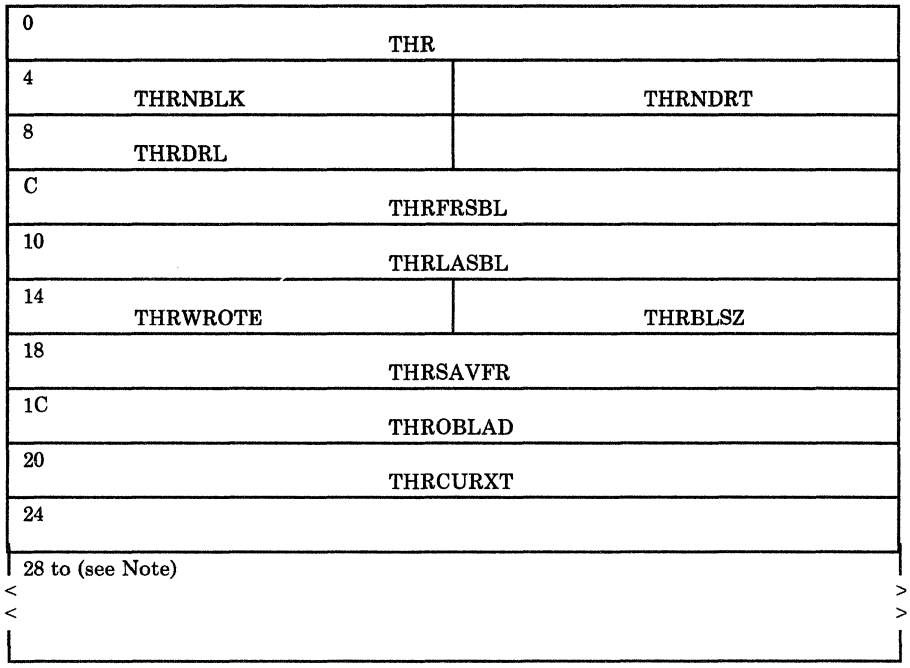
Track Header Record for FB-512



Displacement Hex	Dec	Field Name		Description	
0	0	THR	DC	CL4'THR'	ID of track header record
4	4	THRNBLK	DS	H	The number of blocks in the record
6	6	THRNDRT	DC	H	The number of 4K data records on tape
8	8	THRDRL	DC	H	Length of the short (last) data record on tape
A	10	THRFLAG	DC	X'0'	Track Header Record Flag
B	11		DS	X	Reserved for IBM use
C	12	THRFRSBL	DS	F	First block in record
10	16	THRLASBL	DS	F	Last block in record
14	20	THRWROTE	DS	H	Number of blocks to be output
16	22	THRBLSZ	DS	H	512 - the size of one block
18	24	THRSAVFR	DS	F	Save area for first block number
1C	28	THROBLAD	DS	F	Block number where output should begin
20	32	THRCURXT	DS	F	Address of entry in extent
24	36	THRDATA			The actual FB-512 data

Figure 6-7. Track Header Record for FB-512

Track Header Record for FB-512 (Compacted)



Displacement Hex	Dec	Field Name		Description	
0	0	THR	DC	CL4'THR'	ID of track header record
4	4	THRNBLK	DS	H	The number of blocks in the record
6	6	THRNDRT	DC	H	The number of compacted bytes
8	8	THRDRL	DC	H	Length of the short (last) data record on tape
A	10	THRFLAG	DC	X'0'	Track Header Record Flag
		Bit settings for THRFLAG			
		SPECIAL	EQU	X'01'	Special
		FTRMODE	EQU	X'02'	Header in FTR mode
		CMPCOMP	EQU	X'80'	Encoded data
		CMPLBLK	EQU	X'40'	Compaction flag
B	11		DS	X	Reserved for IBM use
C	12	THRFRSBL	DS	F	First block in record
10	16	THRLASBL	DS	F	Last block in record
14	20	THRWROTE	DS	H	Number of blocks to be output
16	22	THRBLSZ	DS	H	512 - the size of one block
18	24	THRSAVFR	DS	F	Save area for first block number
1C	28	THROBLAD	DS	F	Block number where output should begin
20	32	THRCURXT	DS	F	Address of entry in extent
24	36		DS	F	The number of bytes of non-compacted data
28	40				Compacted data

Note: 28 to 1FFF for 800 bpi, 28 to 2FFF for 1600 bpi, and 28 to BFFF for 6250 bpi and 38K bpi tape.

Figure 6-8. Track Header Record for FB-512 (Compacted)

IOB

0 IOBSTAT	1 IOBOPT	2 IOBUADD
4 IOBCCW		
8 IOBERROR		
C IOBCSW		
14 IOBCLASS	15 IOBTYPE	16 IOBMREC
18 IOBCYLP		1A IOBCYLA
1C IOBMTCK		1E IOBMODE / 1F IOBDISP
20 IOBVSER		
24		26 IOBATAPE
28 IOBFLAG	29 Reserved for IBM use	
2C IOBWHATQ		
30 IOBTHR		
34 IOBIRA		
38 IOBOUTBF		

Displacement Hex	Dec	Field Name			Description
0	0	IOBSTAT	DS	X'80'	Status of IOB
		Bit settings for IOBSTAT			
		IOBST	EQU	X'80'	I/O unit is to be started
		IOBSTACK	EQU	X'40'	I/O error has been stacked
		IOBLAST	EQU	X'20'	Last IOB
		IOBNOPER	EQU	X'10'	Device is not operational
		IOBCPVOL	EQU	X'08'	Unit is a CPVOL
		IOBOPEN	EQU	X'04'	The IOB is open
		IOBSCRAT	EQU	X'02'	The DASD device is a scratch volume
		IOBTPSWP	EQU	X'01'	Switch to alternate tape in progress
1	1	IOBOPT	DS	1X	IOB flags

Figure 6-9 (Part 1 of 2). IOB (Input/Output Block) Format

Displacement Hex	Dec	Field Name			Description
		Bit settings for IOBOPT			
		IOBDEW	EQU	X'80'	Wait for device end interrupt
		IOBERST	EQU	X'40X'.	Stop on I/O error and wait for next interrupt
		IOBEEXIT	EQU	X'20'	Repeat CCW on error
		IOBSIO	EQU	X'10'	Do not use Diagnose I/O
		IOBOVER	EQU	X'08'	Used only by SIO routine. Means entry was via STARTIO overlay entry; therefore, do SIO and return to caller when subsequent cc=0
		IOBFTR	EQU	X'02'	Use full track read feature
2	2	IOBUADD	DS	1H	Unit address of device
4	4	IOBCCW	DS	1F	Pointer to CCW
8	8	IOBERROR	DS	A	Address of IO error routine
C	12	IOBCSW	DS	2F	CSW of IO error stacked
14	20	IOBCLASS	DS	X'0'	Device class
15	21	IOBTYPE	DS	X'0'	Device type
16	22	IOBSKIP	EQU	*	IOB type skip count
16	22	IOBMREC	DS	H'0'	Maximum number of records that will fit a track
18	24	IOBCYLP	DS	H'0'	Maximum primary cylinder address of DASD device.
1A	26	IOBCYLA	DS	H'0'	Maximum alternate cylinder address of DASD device.
1C	28	IOBMTCK	DS	H'0'	Maximum number of tracks (numbering 0-N)
1E	30	IOBMODE	DS	X	IOB tape mode command code
1F	31	IOBDISP	DS	X	IOB tape disposition command code
20	32	IOBVSER	DS	CL6' '	Volume serial number of DASD unit
26	38	IOBATAPE	DS	X'0000'	Address of an alternate tape unit
28	40	IOBFLAG	DS	X'0'	IOB flag
29	41		DS	3'0'	Reserved for IBM use
2C	44	IOBWHATQ	DS	F	Address of the double buffering queue that this IOB will service
30	48	IOBTHR	DS	F	Address of I/O area being used by this job
34	52	IOBIRA	DS	F	Interruption return address for overlapped I/O
38	56	IOBOUTBF	DS	F	Address of anchor where THR will be enqueued when buffer becomes available
		IOBSIZE	EQU	*-IOB	Address of an alternate tape unit

Figure 6-9 (Part 2 of 2). IOB (Input/Output Block) Format

Trace Table

Figure 6-10 shows the trace table format. Trace table addresses may be obtained by referencing the module and finding these labels:

- TRACEST - Beginning of trace table
- TRACEND - End of trace table
- TRACEPT - Pointer to next available entry

Byte 0 is overlaid by the trace identification code.

Event	Id. Code	Format of Trace Entry
Start I/O IPL	E2	X'E2' | CAW | First IOB word | Return Address | IOB 1 4 8 12
Interrupt[1] (native)	C9	X'C9' | I/O Old PSW | CSW
Error	C5	X'C5' | CAW | Device address | Sense Information
Interrupt (virtual)	C9	X'C9' | CAW or sense | Diagnose 20 RC | CSW

[1] Byte 0 is overlayed by the trace identification code.

Figure 6-10. DDR Trace Table Format

Diagnostic Aids

Figure 6-11 lists the messages issued by the DASD Dump Restore Program. The associated label and method of operation diagram are included in the list.

Message Code	Label	Diagram	Message Text
DMKDDR536I	DDR536		raddr devname REPORTS DISABLED INTERFACE; FAULT CODE = cccc; NOTIFY CE
DMKDDR700E	DDR700		INPUT UNIT IS NOT A CPVOL
DMKDDR701E	DDR701	6-2	INVALID OPERAND - operand
DMKDDR702E	DDR702		CONTROL STATEMENT SEQUENCE ERROR
DMKDDR703E	DDR703	6-2	OPERAND MISSING
DMKDDR704E	DDR704		DEV raddr NOT OPERATIONAL
DMKDDR705E	DDR705		I/O ERROR addr CSW = csw SENSE = sense INPUT = bbcchh OUTPUT = {bbcchh|nnnnnn} CCW = ccw
DMKDDR707E	DDR707		MACHINE CHECK
DMKDDR708E	DDR708		INVALID INPUT OR OUTPUT DEFINITION
DMKDDR709E	DDR709		WRONG INPUT TAPE MOUNTED
DMKDDR710A	DDR710		DEV raddr INTERVENTION required
DMKDDR711R	DDR711		VOLID READ IS volid2 [NOT volid1] DO YOU WISH TO CONTINUE? RESPOND YES NO OR REREAD:
DMKDDR712E	DDR712		NUMBER OF EXTENTS EXCEEDS 20
DMKDDR713E	DDR713		OVERLAPPING OR INVALID EXTENTS
DMKDDR714E	DDR714		RECORD {bbcchh/nnnnnn} NOT FOUND ON TAPE
DMKDDR715E	DDR715		LOCATION bbcchh IS A FLAGGED TRACK
DMKDDR716R	DDR716		NO VOL1 LABEL FOUND [FOR volser]
DMKDDR717R	DDR717	6-4	DATA DUMPED FROM volid1 TO BE RESTORED TO volid2 DO YOU WISH TO CONTINUE? RESPOND YES OR NO OR REREAD:
DMKDDR718E	DDR718		OUTPUT UNIT IS FILE PROTECTED
DMKDDR719E	DDR719		INVALID FILENAME OR FILE NOT FOUND

Figure 6-11 (Part 1 of 2). The DASD Dump Restore Program Messages

Message Code	Label	Diagram	Message Text
DMKDDR720E	DDR720		ERROR IN {routine|nnnnnn}
DMKDDR729I	DDR729		FULL TRACK READ FEATURE NOT AVAILABLE
DMKDDR721E	DDR721		RECORD {cchhr/nnnnnn} NOT FOUND
DMKDDR722E	DDR722		OUTPUT UNIT NOT PROPERLY FORMATTED FOR THE CP NUCLEUS
DMKDDR723E	DDR723		NO VALID CP NUCLEUS ON THE INPUT UNIT
DMKDDR724E	DDR724		INPUT TAPE CONTAINS A CP NUCLEUS DUMP
DMKDDR725R	DDR725	6-2,6-4	ORIGINAL INPUT DEVICE WAS (IS) LARGER THAN OUTPUT DEVICE. DO YOU WISH TO CONTINUE? RESPOND YES NO OR REREAD.
DMKDDR726E	DDR726		MOVING DATA INTO THE ALTERNATE TRACK CYLINDER(S) IS PROHIBITED
DMKDDR727E	DDR727		FLAGGED TRK xxxxxxxxxxxx HAS NO PROPER ALTERNATE; SKIPPING THIS TRK
DMKDDR728E	DDR728	6-5	DECODE ERROR ENCOUNTERED: xx
DMKDDR729I			FULL TRACK READ FEATURE NOT AVAILABLE
DMKDDR731I	DDR731	6-7	COMPACT OPTION IS IGNORED
DMKDDR756E	DDR706		PROGRAM CHECK PSW = psw
	MSG002	6-1	VM/370 DASD DUMP/RESTORE PROGRAM RELEASE n
	NEWADD		
	MSG02A		ENTER CARD READER ADDRESS OR CONTROL STATEMENTS
	MSG003		ENTER CYLINDER EXTENTS OR ENTER BLOCK EXTENTS
	MSG03B		ENTER NEXT EXTENT OR NULL LINE
	MSG005		END OF VOLUME CYL xxx HD xxx, MOUNT NEXT TAPE OR END OF VOLUME BLOCK xxxxxxxx, MOUNT NEXT TAPE
	MSG004		RESTORING xxxxxx
	MSG004		COPYING xxxxxx
	MSG004		DUMPING xxxxxx
	MSG004	6-6	PRINTING xxxxxx
	MSG001	6-3	END OF DUMP
	CLOSEJOB	6-3	END OF RESTORE
		6-3	END OF COPY
	MSG001	6-3	END OF PRINT
	EOJ		
	MSG001		END OF JOB
	RESPMSG		DO YOU WISH TO CONTINUE? RESPOND YES NO OR REREAD:
	RESPMSG2		DO YOU WISH TO CONTINUE? RESPOND YES OR NO

Figure 6-11 (Part 2 of 2). The DASD Dump Restore Program Messages

Chapter 7. The Installation Verification Procedure

Introduction

The Installation Verification Procedure (IVP) is designed to exercise the generated system to verify that basic facilities are operable. The IVP is contained in two files using the EXEC facility of CMS, and uses two virtual machines in addition to the system operator's virtual machine.

The tests exercise the following areas of CP:

- Multiple virtual machine support
- I/O spooling
- Transferring of spooled data to other virtual machines
- Offline I/O operations
- Sending of messages to the system operator
- Paging operations
- Task dispatching and scheduling
- Disk I/O support
- Automatic warm start following abnormal termination.

The following facilities of CMS are exercised:

- Normal CMS command processing
- Disk formatting
- Copying of files
- Creation and modification of files via EDIT command
- Assembly of executable programs
- Execution of user programs
- Creation and execution of user-written commands
- Printing and punching of CMS files
- Issuing of commands to CP
- Use of multilevel nested EXEC procedures
- Stacking and unstacking of command and data input from the terminal
- Communication with user from EXEC procedures.

Several other system facilities, incidental to the primary IVP tests, are exercised. Certain system facilities, such as preferred execution options, virtual=real, OS ISAM, and VSAM and Access Method Services under CMS, are not exercised by the IVP.

The IVP requires operator intervention only when an operational decision is to be made, or to initiate the IVP tests themselves. All file creation, erasure, management, and logoff of the virtual machines (with the exception of the system operator) at test completion is performed automatically without operator or user action.

Method of Operation

This section describes the execution of the two EXEC procedures of the IVP (Installation Verification Procedure).

Figure 7-1 shows the relationship of the diagrams.

Diagram 7-1 describes the highest level EXEC procedure, IVP.

Diagram 7-2 describes the major functions of the nested EXEC procedure IVPX.

Diagram 7-3 describes test procedure 1.

Diagram 7-4 describes test procedure 2.

Diagram 7-5 describes the error processing.

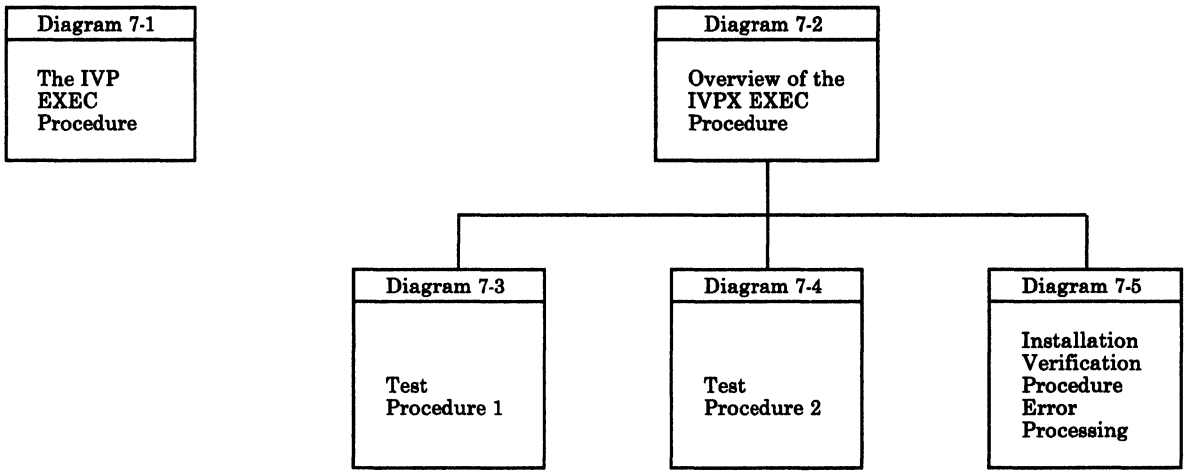


Figure 7-1. Key to the Installation Verification Procedure Method of Operation Diagrams

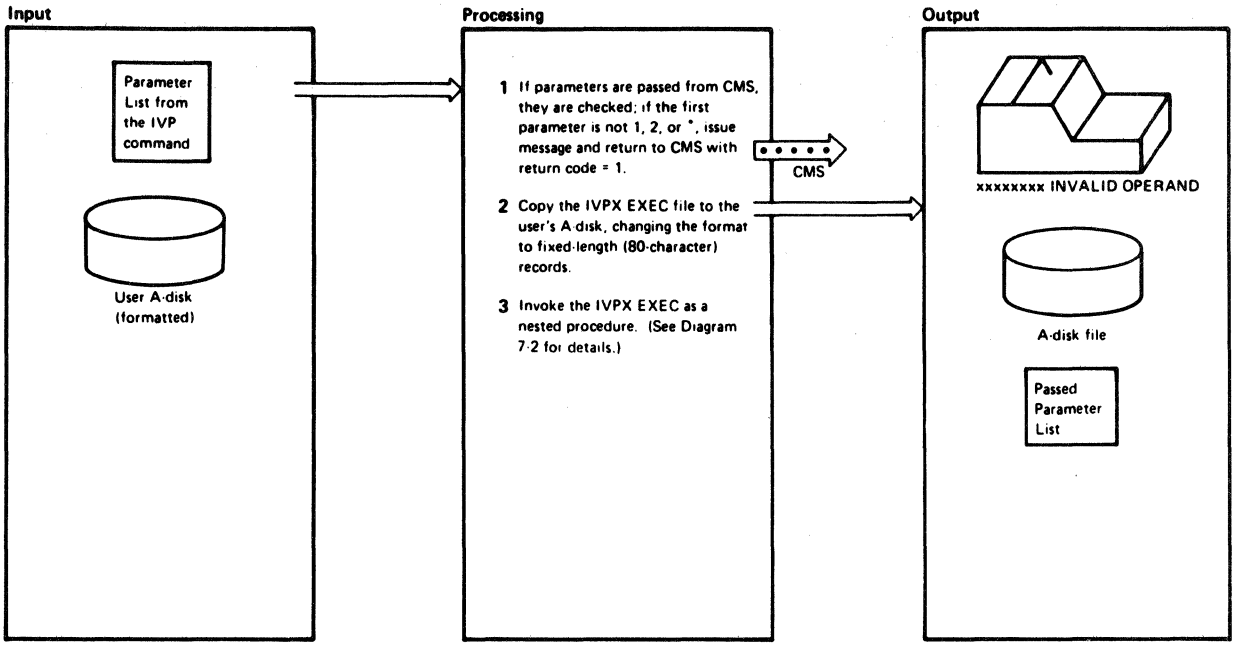
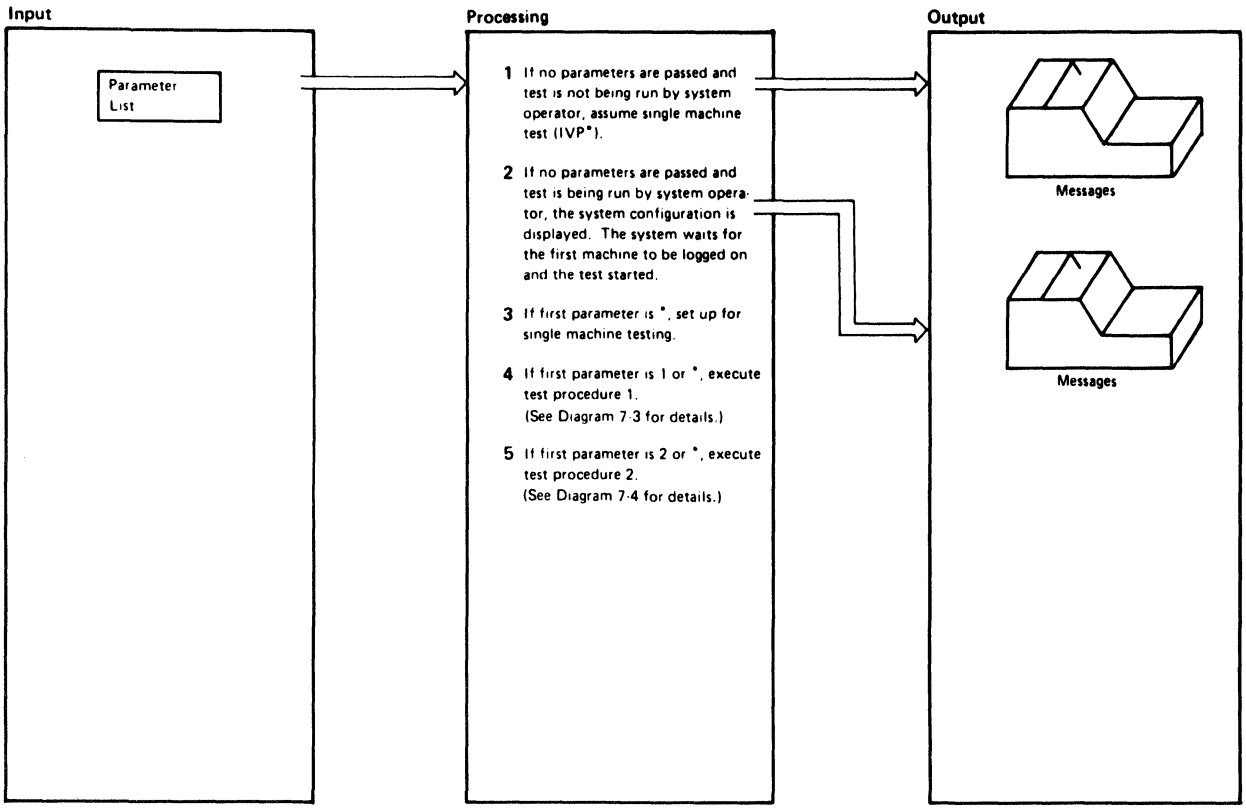


Diagram 7-1. The IVP EXEC Procedure



Notes	Module	Label	Ref
1 When no parameters are specified on the IVP command, the message *** ARE YOU THE SYSTEM OPERATOR? ENTER "YES" OR "NO" is displayed. If the response is NO, the message *** NOT SYSTEM OPERATOR- DEFAULT TO IVP* is displayed, single machine testing is set up (-INIT), and the testing starts at test procedure 1.	IVPX	-CKOP	
2 The real system configuration is displayed. Then the virtual machine enters a dormant state which can be interrupted by signaling attention from the terminal. The message *** THIS PORTION OR IVP NOW GOING TO SLEEP is displayed and the system waits.	IVPX	-CKOP	

Notes	Module	Label	Ref
3 Set &GLOBAL2=4 to indicate single machine test. Erase all CMS files with filenames IVPTST and IVPTST2. If return code is other than 0 or 2, the ERASE command (to erase the EXEC file) is stacked in the terminal and control returns to the CMS command environment. If the return code is 0 or 2, test procedure 1 (MAIN1A) is executed.		-INITB -GETOUT	

Diagram 7-2. Overview of the IVPX EXEC Procedure

Input

Processing

1 If userid not specified, set it to IVPM2.

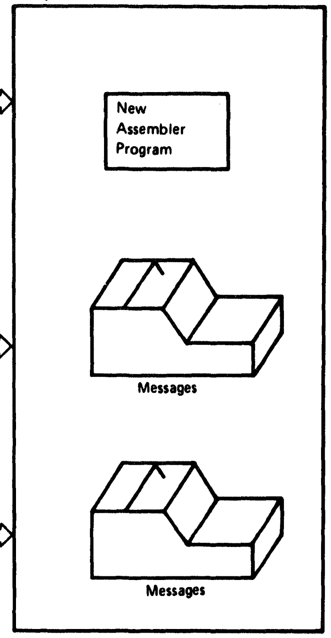
2 Test Section 1
- For multiple machine test, spool the punch to the userid.
- Create a new assembler language file.

3 Test Section 2
- Assemble the test program.
- Create an executable module.

4 Test Section 3
- Execute the test program by name.
- Read from the virtual card reader.
- For multiple machine test, disconnect the first machine and test the second machine.

5 Test Section 4
- Erase all files created by test procedure 1. If single machine, go to test procedure 2 (-MAIN2A), otherwise LOGOFF.

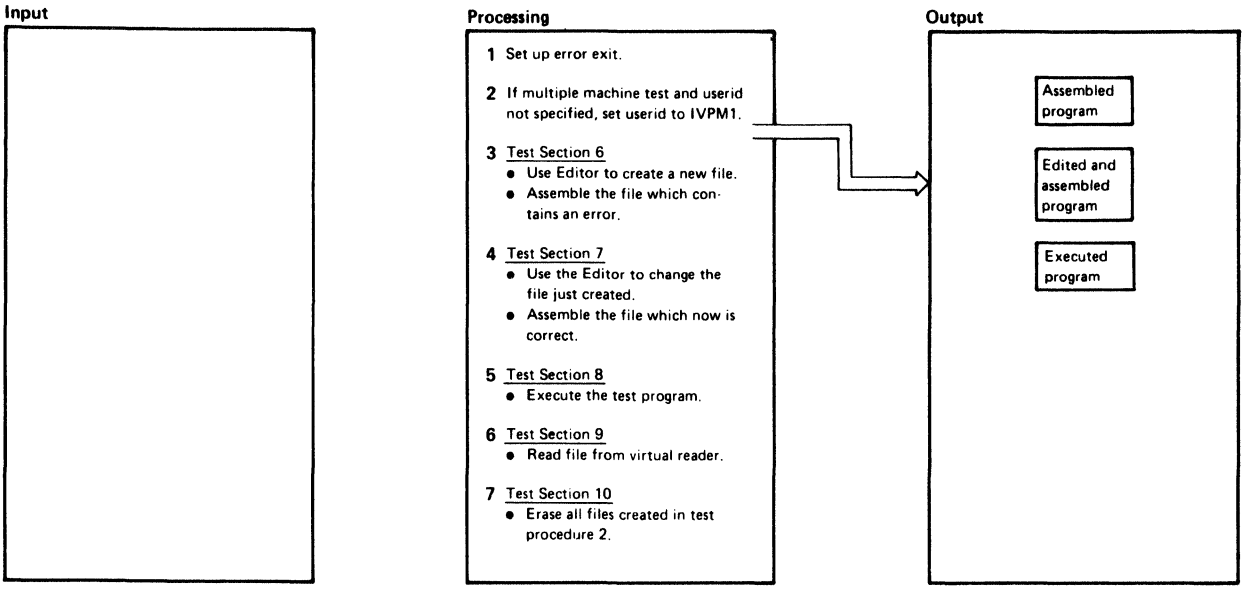
Output



Notes	Module	Label	Ref
1 For a multiple machine test, the userid is set to IVPM2 or to the userid specified as the second operand of the IVP command. When the userid is set to IVPM2, &GLOBAL5 is set to 2 to indicate the standard test.	IVPX	MAIN1	
2 The assembler language statements are stacked in the terminal input buffer and edited.	IVPX	-MAIN1A	
3 The test program created in test section 1 is first assembled (ASSEMBLE command) and then made executable by issuing the LOAD and GENMOD commands.	IVPX	-K256	
4 The test program, IVPTST, is executed Next a READ is issued to the virtual reader and a return code is requested.	IVPX	-LOOPA	
If the return code is other than 0 or 8, the ERASE command to erase the EXEC file is stacked in the terminal, and control returns to the CMS command environment.		-GETOUT	
When testing multiple machines, the following message is issued: *** THIS PORTION OF IVP NOW GOING TO SLEEP The first machine is then disconnected. The operator enters the above commands to start the second machine. The procedure loops (control keeps returning to -LOOPA) until the file to start the second machine is spooled to the reader. The STATE command is issued to verify the existence of the file. The second machine is started.	-FINIS		

Notes	Module	Label	Ref
5 All the IVPTST files are erased. If the test machine is still connected (&GLOBAL2≠3) the following messages are issued. *** TEST SECTION 5 RESERVED FOR FUTURE USE *** *** IVP TEST 1 SUCCESSFULLY COMPLETED These same messages are sent to the punch if the test machine is already disconnected (&GLOBAL=3). The single machine test resumes at -MAIN2A, test procedure 2. If the standard test is running the message. *** IVP TEST 1 FINISHED	IVPX	-INLINE	
is sent to the system operator. If &GLOBAL5=1, the test is running in 256K bytes of storage. If running machine tests, go to the LOGOUT routine. The following commands are stacked. ERASE IVPX EXEC A1 CP LOGOUT The LOGOUT routine closes all files including the punch containing the messages issued after test machine 1 was disconnected. The multiple machine test resumes at -MAIN2, test procedure 2.		-LOGOUT	

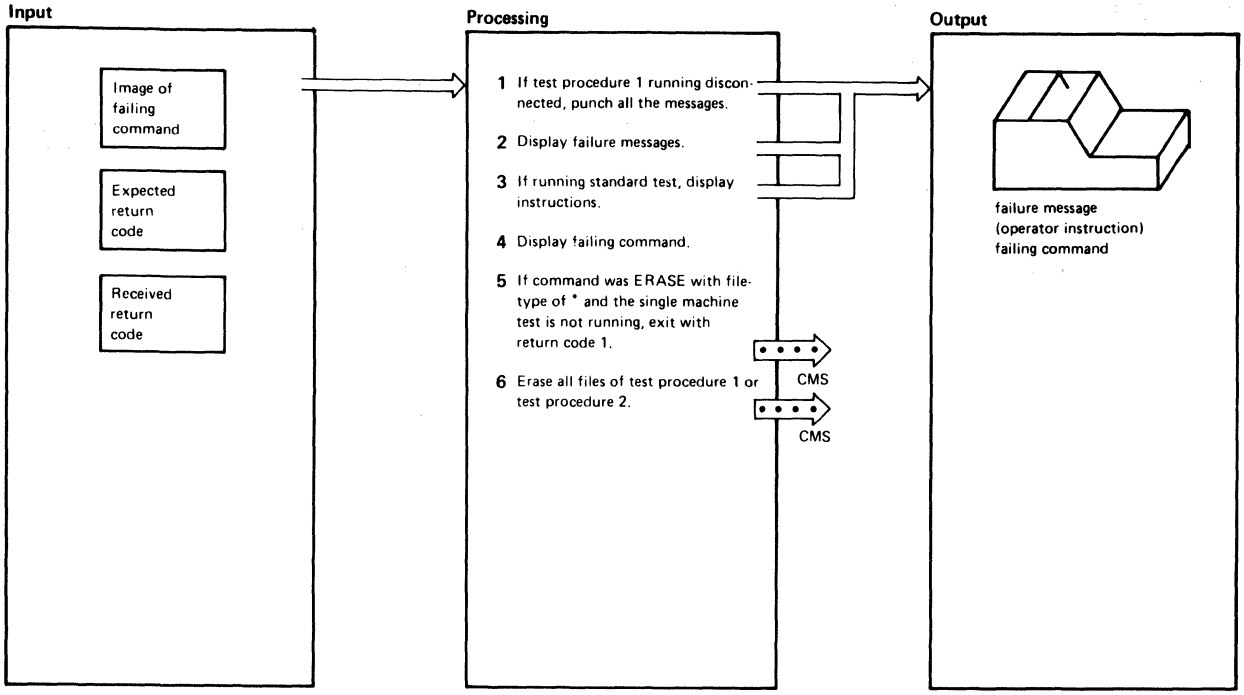
Diagram 7-3. Test Procedure 1



Notes	Module	Label	Ref
1 Set the error exit to –FAIL2. For a single machine test, edit directly to the CMS command environment. The ERASE and LOGOUT commands are stacked in the terminal and the EXEC procedure exits with a return code of 1. Execution is now ended within the nested EXEC. The return code of 1 forces the next level EXEC to exit to the CMS command environment.	IVPX	–FAIL2	
2 For a multiple machine test, the userid is set to IVPM1 or to the userid specified as the second operand of the IVP command. When the userid is set to IVPM1, &GLOGAL5 is set to 2 to indicate the standard test.	IVPX	–MAIN2	
3 The input data is stacked for the editor, which creates the IVPTST2 ASSEMBLE file. The file just created is assembled. Error 8 occurs because the ASSEMBLE file contains one error.	IVPX	–MAIN2A	
4 The statement in error is corrected. The file is then assembled. Since the error is corrected the TEXT file is created.	IVPX		
5 The test program is loaded and then started.	IVPX	–MAIN2A	
6 The file is read from the virtual reader. If there is no file in the reader on the first loop, a file is created, punched, and spooled to the reader.	IVPX	–LOOP –LOOP2	

Notes	Module	Label	Ref
For a single machine test, a dummy message file is created, punched, and spooled to the reader on the same machine. For a multiple machine test, the messages are spooled to the reader on the userid system. The input is stacked in the terminal for the editor. A dummy message is edited and punched. Control returns to –LOOP.			
The STATE command is issued to be sure the file is successfully read onto disk. The contents of the file are displayed. For multiple machine standard test, the message DON'T START SPOOL DEVICES UNTIL TOLD is sent to the system operator. The multiple machine test determines that the file was successfully read and punches and prints that file.		–LOOP1	
7 All files are erased and messages are displayed. ***IVP TEST 2 SUCCESSFULLY COMPLETED ***IVP PROCEDURE FINISHED	IVPX	–NOSPL	
If a single machine test, the command to erase the EXEC file is stacked in the terminal and control returns to the CMS command environment. If a multiple machine test, the commands to erase the EXEC file and LOGOUT are stacked for CMS. If running the standard test, the messages ***IVP TEST NOW FINISHED ***SIGNAL ATTN AND ENTER: BEGIN are sent to the system operator. For the the multiple machine test, control then returns to the CMS command environment.		–GETOUT	

Diagram 7-4. Test Procedure 2



Notes	Module	Label	Ref
1 If test machine 1 is disconnected the messages are sent to the punch, rather than the virtual machine console.	IVPX	·CHECK1	
2 The message *** IVP FAILURE HAS OCCURRED *** is displayed.	IVPX		
3 The messages *** IVP HAS FAILED – REPLY NO TO ABORT MESSAGE *** SIGNAL ATTN AND ENTER: BEGIN are sent to the system operator.	IVPX		
4 The messages *** COMMAND: xxxxxxxx *** EXPECTED RETURN CODE xxx *** RECEIVED RETURN CODE xxx are displayed.	IVPX	CHECK2	
5 Control returns to the next level EXEC procedure and the return code of 1 forces that level to return to the CMS command environment.	IVPX		
6 If the number of the test section is less than 6, all the IVPTST files are erased. If the number of the test section is greater than 5, all the IVPTST2 files are erased. Because this is a nested EXEC procedure, exit with a nonzero return code. A nonzero return code forces the next level EXEC to return to the CMS command environment.	IVPX	·QUIT	

Notes	Module	Label	Ref

Diagram 7-5. Installation Verification Procedure Error Processing

Program Organization

The IVP (Installation Verification Procedure) consists of two EXEC procedures: IVP and IVPX. Figure 7-2 shows the structuring of the major routines of the IVP. Figure 7-3 relates the test sections to the CP or CMS functions being exercised.

Installation Verification Procedure Routine Structuring

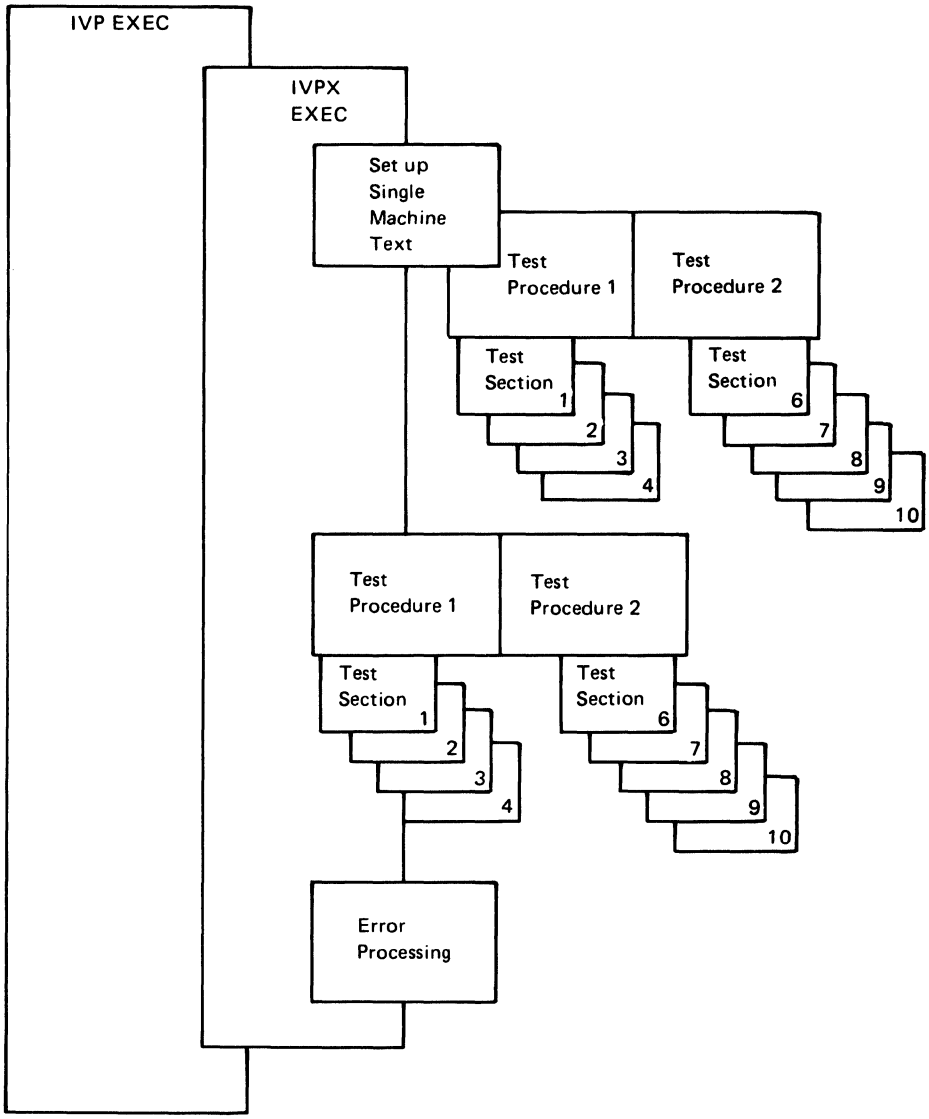


Figure 7-2. Structure of Installation Verification Procedure Routines

Installation Verification Procedure Testing

Program	Function Tested	Test Section and Comments
CP	Multiple virtual machine support	Test Procedures 1 and 2 test multiple virtual machine support when IVP * is not specified or assumed.
	I/O Spooling	Test Section 9.
	Transferring of spooled data to other virtual machines	Test Section 9 when IVP * is not specified or assumed.
	Offline operations	Test Section 9.
	Sending messages to system operator.	Test Sections 4 and 9.
	Page operations	Used throughout IVP.
	Task dispatching and scheduling	Used throughout IVP.
	Disk I/O support	Used throughout IVP.
	Automatic warm start	Error processing.
CMS	Command processing	Used throughout IVP.
	Copying of files	The IVP EXEC procedure.
	Creation and modification of files via EDIT command	Test Sections 1, 6, and 7.
	Assembly of executable modules	Test Sections 2, 6, and 7.
	Execution of user programs	Test Sections 3 and 8.
	Creation and execution of user-written commands	Test Section 3.
	Printing and punching of CMS files.	Test Section 7.
	Multilevel EXEC procedures.	Used throughout IVP.

Figure 7-3. Installation Verification Procedure Tests

Directory

This section contains an alphabetical list of the labels in the IVPX EXEC procedure. Figure 7-4 describes the function performed at the point in the program corresponding to each label; the associated method of operation diagram is referenced.

Label	Diagram	Description
-CHECK1	7-5	Sends messages to punch when machine is disconnected.
-CHECK2	7-5	Displays the failing command.
-CKOP	7-2	Sets up for execution when IVP is invoked without any parameters specified.
-FAIL2	7-4	Exits to CMS command environment if single machine test is running. Issues instructions if multiple machine test is running.
-FINIS	7-3	End of Test Procedure 1.
-GETOUT	7-2 7-3 7-4	Error exit for single machine test.
-INITB	7-2	Sets up for single machine test.
-INLINE	7-3	Erases all files created during Test Procedure 1.
-K256	7-3	Assembles and executes the program created in Test Section 1.
-LOGOUT	7-3 7-4	Error exit for multiple machine test.
-LOOP	7-4	Reads file from the virtual reader during Test Procedure 2.
-LOOPA	7-3	Reads from the virtual reader during Test Procedure 1.
-LOOP1	7-4	Checks that file is read to disk successfully.
-LOOP2	7-4	Creates file, punches it, and spools it to reader when there is no file in the reader.
-MAIN1	7-3	Beginning of Test Procedure 1.
-MAIN1A	7-3	Point in Test Procedure 1 where the single machine test begins.
-MAIN2	7-4	Beginning of Test Procedure 2.
-MAIN2A	7-4	Point in Test Procedure 2 where the single machine test begins.
-NOSPL	7-4	Erases all files created in Test Procedure 2.
-QUIT	7-5	Abnormal end exit from a nested EXEC procedure.

Figure 7-4. Installation Verification Procedure Label Directory

Diagnostic Aids

Figure 7-5 is a list of all the messages that the IVPX EXEC procedure issues, the label nearest to the point where the message is issued, and the associated method of operation diagram.

Label	Diagram	Message Text
-CKOP	7-2	*** ARE YOU THE SYSTEM OPERATOR? ENTER "YES" OR "NO".
-CKOP	7-2	*** NOT SYSTEM OPERATOR - DEFAULT TO IVP *
-CKOP	7-2	*** FROM A TERMINAL, ENTER THE FOLLOWING FOUR COMMANDS: LOGIN IVPM1 (WHEN REQUESTED, ENTER THE PASSWORD IVPASS) DEFINE STORAGE AS 16384K IPL 190 IVP 1
-CKOP	7-2	*** THIS PORTION OF IVP NOW GOING TO SLEEP. *** STARTING SYSTEM ABORT ROUTINE.
-ABMSG		*** ENTER "GO" TO CONTINUE OR "NO" TO QUIT.
-ABMSG		*** THIS IS THE LAST STEP OF THE IVP PROCEDURE. *** FOLLOWING SYSTEM RESTART (WARM START), START SPOOLING DEVICES.
-ABMSG		MANUALLY DEPRESS CPU RESTART KEY TO ABORT SYSTEM.
-PERFORM		*** STARTING TEST SECTION x
-CHECK1	7-5	*** IVP FAILURE HAS OCCURRED ***
-CHECK1	7-5	*** IVP HAS FAILED - REPLY NO TO ABORT MESSAGE *** SIGNAL ATTN AND ENTER: BEGIN
-CHECK2	7-5	*** COMMAND: xxxxxxxx *** EXPECTED RETURN CODE xxx *** RECEIVED RETURN CODE xxx
-LOOPA	7-3	*** WHEN "VM/370 ONLINE" APPEARS, ENTER THE FOLLOWING THREE COMMANDS: LOGIN xxxxxxxx (WHEN REQUESTED, ENTER THE APPROPRIATE PASSWORD) (IF LOGGING IN IVPM2, THE PASSWORD IS: IVPASS) IPL 190 IVP 2 *** THIS PORTION IS NOW DISCONNECTING
-INLINE	7-3	*** TEST SECTION 5 RESERVED FOR FUTURE USE ***
-INLINE1	7-3	*** IVP TEST 1 SUCCESSFULLY COMPLETED *** IVP TEST 1 FINISHED
-LOOP1	7-4	DON'T START SPOOL DEVICES UNTIL TOLD.
-NOSPL	7-4	*** IVP TEST 2 SUCCESSFULLY COMPLETED *** IVP PROCEDURE FINISHED
-NOSPL	7-4	*** IVP TEST 2 FINISHED *** SIGNAL ATTN AND ENTER: BEGIN
-FAIL2	7-4	*** WHEN "VM/370 ONLINE" APPEARS, ENTER THE FOLLOWING TWO COMMANDS: LOGIN xxxxxxxx (WHEN REQUESTED, ENTER THE APPROPRIATE PASSWORD) LOGOUT

Figure 7-5. The Installation Verification Procedure Messages

Chapter 8. Procedures for Generating and Updating CP and CMS

Introduction

The procedures covered in this chapter provide for the updating of files with several levels of updates and any number of program temporary fixes (PTFs). For Assembler language source statement files, procedures are supplied for assembling the updated source code to produce a uniquely defined text deck. The deck has a unique name and some control cards to identify the origin of the updates, macro libraries, and source statements. For macro library files, a copy file is produced to identify the origin of the input and any updates applied.

Procedures are provided for generating load files from various object modules, for generating MACLIB files from various COPY and MACRO files, and creating text libraries from TEXT files.

The procedure for updating VM/SP HPO has a file naming convention for update and text files, a set of programs to support the processing, and a set of EXEC procedures and modules to process the files.

- The VMFASM procedure incorporates PTFs or updates.
- The VMFLOAD procedure creates a new CP or CMS nucleus.
- The VMFMAC procedure generates a new macro library.
- The VMFNLS procedure updates national language related files.
- The VMFTXT procedure creates a new text library.

Update Files

Files used to update another file are given a filetype of UPDTxxxx, where xxxx is a unique *update identifier* for programmer and system use. The filename of the update file must be the same name as the file to be updated. For instance, the file PROGRAM ASSEMBLE could be updated by the file PROGRAM UPDTGN30 or the file PROGRAM UPDTGC61.

The creation and use of update files are described in the UPDATE command discussion in the *VM/SP CMS Macros and Function Reference.*

TXT Files

Text files are produced by the assembler as a part of the VMFASM procedure. The filename of the text file is the same as the filename of the ASSEMBLE file. The filetype of the completed text deck is TXTnamex, where 'namex' represents a unique *update level identifer*. The value of 'namex' is taken from a control file, and corresponds to the highest level of update applied. In addition, the text deck is produced from a combination of the assembler text deck and an auxiliary control file containing data describing the origin of the files used. The auxiliary file is called 'filename UPDATES' and is produced by a program called VMFDATE. The filename is the same as the filename of the UPDTxxxx file.

Control Files

Each user may have several control files to specify various combinations of updates and macro libraries to be used. A control file must have a filetype of CNTRL. These control files contain records in the following format:

nam00 MACS maclib1 maclib2 ...
nam01 UPDTup1
nam02 UPDTup2
nam03 UPDTup3
nam04 AUXxxxxx

The suffixes up1, up2, up3, and xxxxx are *update identifier* fields, and the fields nam00, nam01, nam02, nam03, and nam04 are *update level identifiers.*

The first record is the MACS record that lists in search order the macro libraries (maclib1 maclib2) to be used in the assembly. Up to 29 libraries may be specified (subject to the character limit of the MACS record line).

Records 2, 3, and 4 are update identification records. They define the UPDTxxxx files that were created (via update control cards and source statements) to update some particular file. Record 2 defines a UPDTup1 file, and records 3 and 4 define UPDTup2 and UPDTup3 updates, respectively. None, some, or all of the updates may exist to be applied.

Record 5 defines an auxiliary file that specifies an auxiliary list of PTFs or updates that are to be applied. Record 5 defines an auxiliary file identified as 'filename AUXxxxxx', where 'filename' is the same as the filename of the input file and xxxxx is an update identifier (the update identifier for an auxiliary control file cannot be "aux"). Records in the auxiliary file have the following format for PTFs to be applied:

```
PTF   PTF001          comments
      PTF002
PTF   PTF003
*     Any comment
```


The PTF field is an optional identifier, and the second field (for example, PTF001) defines a specific PTF to be applied. The PTF has a 'filename PTF001' identification, where 'filename' is the same as the filename of the file to be updated. The comment field is used to describe the function of the particular PTF. The * record is ignored and is used to provide additional comments on any updates or PTFs.

The updates (PTFs included) are applied in the reverse order in which they appear. In the previous example, the updates would be applied in the following order:

```
PTF003
PTF002
PTF001
UPDTup3
UPDTup2
UPDTup1
```

The PTF records can be directly included in the CNTRL file if desired, but it is usually more convenient to place them in a separate auxiliary (AUXxxxxx) file.

There can be any number of UPDTxxxx definition and auxiliary control file definition records, but only one MACS record. The complete CNTRL file can have any filename, but typically has the same name as the first specified UPDTxxxx control record. In the example, the file could be named UP1 CNTRL.

The underlined fields in each record mark the level identification fields. The highest level (last) update to be applied selects the name that can be used to identify updated files. In the example, if UPDTup3 was the last update applied, then the name selected would be nam03. The value for the identification usually consists of a combination of the update identifier up1, up2, ... (up to four characters) and additional characters up to a maximum of 5 for the combined update identifier and additional characters. If no updates are applied, then the nam00 field is selected to identify the TXTnam00 produced. This name can be used to uniquely identify updated files. The text files described above, for instance, can have a filetype of TXTup3. It is desirable, on occasion, to have entries in the user CNTRL file that specify a level identification but no update. A record of the following format, for example, is allowed:

```
nam05
```

This is because the control file serves a double purpose and is used for loading text decks as well as updating input files. An identifier of TEXT as a name causes special handling in the VMFASM EXEC procedure, whether or not an update is used with it. A name of TEXT is used without level identification catenation. Thus, TEXT becomes the filetype.

System EXEC Procedures

Several system control files provide for system update and creation. Some EXEC procedures invoke others or make use of user-supplied control files to accomplish various functions such as multilevel updating, text generation, and macro library generation.

VMFASM EXEC Procedure

The VMFASM procedure performs the multilevel update function by invoking the DMSUPD module (via the CMS UPDATE command) before assembling the desired files. To update and assemble a source file, the VMFASM procedure is invoked in the following way:

```
VMFASM filename control options
```

where 'filename' is the name of the ASSEMBLE file to be processed and 'control' is the name of the user CNTRL file that contains the MACS (macro library), update, and any AUXxxxx control records. The VMFASM procedure invokes the DMSUPD module via the CMS UPDATE command, passing the values 'filename', 'ASSEMBLE', and 'control'.

The UPDATE command returns a level identifier and a MACLIB list from the MACS record of the control file. If the identifier is TEXT, then that becomes the filetype of the complete text deck; otherwise the filetype is TXTxxxxx (for example, TXTup3m1). The EXEC procedure then reads the MACLIB list passed by UPDATE and issues a GLOBAL command to prepare for the assembly using the specified libraries.

The ASSEMBLE program is invoked with the specified options. If no options are specified for the ASSEMBLE command, the defaults are: PRINT, NOTERM, LIST, NODECK, NORENT, SYSPARM(), and XREF(FULL). The options that can be specified for the VMFASM EXEC are: DISK, NOTERM, NOLIST, DECK, RENT, EXP, XREF, and RLD. The defaults for the VMFASM EXEC are: PRINT, TERM, LIST, NODECK, NORENT, SYSPARM(SUP), XREF(SHORT), and NORLD.

The VMFDATE program is used to construct a record for each MACLIB used and for the ASSEMBLE file. Each record is placed in the auxiliary file 'filename UPDATES'. The text deck produced by the assembler is combined with the file produced by the VMFDATE program and is named 'filename TXTxxxxx', where 'filename' is that of the ASSEMBLE file, and 'TXTxxxxx' is constructed from the update level identifier returned by the UPDATE command. All intermediate files are erased, leaving only the original ASSEMBLE and UPDTxxxx files, and the newly created text file.

VMFLOAD Procedure

The VMFLOAD procedure uses two user-supplied files, a loadlist EXEC and a 'control' file identical in format to the CNTRL file used by VMFASM and UPDATE, to produce a punched deck comprised of several text files. VMFLOAD is invoked as a CMS command as follows:

```
VMFLOAD loadlist ctlfile [langid]
```

The loadlist is a user-supplied EXEC file consisting of several records of the following format:

```
&CONTROL OFF
&1 &2 fn [ft] [ (LANG]
&1 &2 fn [ft] [ (LANG]
  .
  .
  .
```

The 'filename' specifies the name of a text file to be punched. The text files are punched in the order specified. If a filetype is specified, a search is made for that specific file, and if it is found it is punched without a header card, and the search then bypasses the control file. LANG is a special option you use for national language-related files, such as message repositories. Any entry with the LANG option is punched with a header card. If you specify *langid* on the VMFLOAD command, VMFLOAD determines the filetype of the object module you want to punch.

If the filetype is not given, the specified control file is used to search for the highest level text file available, and it is punched.

VMFLOAD displays a confirmation or error message upon completion. Before invoking the loadlist procedure, a SPOOL PCH CONT command line is executed to assure that the punched files appear as one deck. The command lines SPOOL PCH NOCONT and CLOSE PCH are executed upon completion.

The control field is used only if the filetype is not specified. The control field specifies a user-supplied control file with a filename of 'control' and a filetype of CNTRL. This control file is of the same type and format as the one used to perform multilevel updates. Indeed, most often the file used to produce the updated and assembled text decks is the one used to load the text decks.

VMFLOAD uses the control file to search for the desired text deck in the order in which the identifiers are specified in the file. The first file located is punched, and all lower files are ignored. If the end is reached without finding a text file, VMFLOAD displays the message 'filename TEXT' NOT FOUND, and continues processing with the next entry in the loadlist EXEC. It is quite possible to have a completed load deck comprised of different levels of text decks.

DMKLD00E Service Program

The loader (DMKLD00E) is a service program that is used to generate a CP or CMS nucleus. The loader loads the text decks supplied with it, resolves CCW addresses, and resolves address constants. The same loader is used whether a virtual=real or standard CP system is generated.

The loader is distributed with the following default I/O addresses:

Console=009
Printer=00E

These addresses can be overridden by a control card that must be placed between the last card of the loader and the first card of the text decks. The format of the control card is:

Column	Contents
1	12-2-9 multipunch (X'02')
2-4	DEV
5	blank
6-13	PRNT=xxx or xxxx (xxx or xxxx is the printer address)
14	blank or comma
15-22	TYPW=xxx or xxxx (xxx or xxxx is the console address)
23-72	blank

The format of the other control cards can be found in the discussion of the LOAD command in the *VM/SP CMS Macros and Function Reference.*

The loader is self-relocating, that is, it is initially loaded at address 8000 (decimal); it then relocates itself to the top of storage. (For example, if the size of the loader is 10K, and the storage size of the system is 256K, the loader will occupy the area of storage between 246K and 256K.) After relocating itself, the loader clears the storage it was originally loaded in. As the loader needs free storage to perform its operations, it extends downward through storage.

The text decks being loaded must not try to overlay either the loader or any address between zero and 100 (hexadecimal). The text decks are loaded into storage in a positive direction (that is, upward through storage). If the text decks are going to overlay the loader's free storage, the operation is terminated.

The VMFMAC Macro Library Update Procedure

The VMFMAC procedure applies updates to copy or macro files and builds a new macro library. The VMFMAC EXEC procedure is invoked with the following command line:

VMFMAC maclibname cntrlname

where:

maclibname is the filename of the file that contains a list of the macro and copy files that are to be included, or updated and included, in the new macro library. This list file must have a filetype of EXEC and each entry in the maclibname EXEC file has the following format:

```
&1 &2 filename1
&1 &2 filename2
      .
      .
      .
```

cntrlname is the filename of the control file used to apply the updates. The control file (filetype CNTRL) may contain the actual update or only the names of other files that contain the updates.

The UPDATE command is issued for each macro or copy file. If the update procedure is successful, the member is added to the NEWMAC MACLIB. After all macro and copy files have been processed, any existing libname MACLIB file is erased and the NEWMAC MACLIB is renamed to libname MACLIB.

VMFNLS Procedure

The VMFNLS EXEC automatically applies updates to source files, generates text files, and renames them so they can be loaded into the system. The format of the VMFNLS command is:

```
VMFNLS fn  ft  [(options. ... [)]]
```

where:

`fn` is the filename of the source file that is to be converted to text.

`ft` is the filetype of the source file that is to be converted to text. Only REPOS, DLCS, and ASSEMBLE are allowed.

`cntrl` is the name of the control file that is used to apply updates to the source file before text is generated.

`options` are options for the three commands that VMFNLS can issue. These commands are GENMSG, CONVERT COMMANDS, and ASSEMBLE.

The VMFNLS exec does different tasks, depending on the type of input source file.

If the input source file is a message repository file or a command syntax definition file:

- VMFNLS applies updates to the source file, producing the file *$fn ft*. If necessary, VMFNLS changes the filename of this temporary *$fn ft* file to match the filename required for the text file; it does not use the filetype, however.
- VMFNLS then determines the langid associated with the source file. If the source filename is only six characters, VMFNLS assigns the langid AMENG as a default; otherwise, it extracts the country code from the 7th and 8th characters of the source filename.

  VMFNLS LANGLIST contains a list of all valid country codes, along with the associated langid and language name. VMFNLS uses this list to convert the source filename to the text filename.
- Next, VMFNLS compiles the source file with the appropriate command.

  1. If the source file is a message repository, it has a filetype of REPOS. VMFNLS invokes GENMSG to produce a text file and a listing file from the source file. The text file that has the same filename as the input file, and a filetype of TXT*langid*. The listing file has the same filename as the text file; however, VMFNLS changes it to instead match the filename of the source file.
  2. If the source file is a definition language for command syntax (DLCS) file, it has a filetype of DLCS. VMFNLS invokes the CONVERT COMMANDS command to produce two text files from this input file. The filenames of these text decks depend on the :DLCS statement contained with the input file. This statement identifies the applid, langid, and whether the input file is a user or system DLCS file.

     For a system DLCS file, the filenames of the text decks are *applid*SPA for the command syntax definition file and *applid*SSY for the translation and synonym table. For a user DLCS file, the filenames of the text decks are *applid*UPA for the command syntax definition file and *applid*USY for the translation and synonym table.

     CONVERT COMMANDS assigns the filetype TXT*langid* to the text files.
- VMFNLS appends the summary of updates to the front of the text file that is produced.

If the input source file is an ASSEMBLE file:

VMFNLS invokes the VMFASM EXEC to apply updates to the source, sends the update log to the printer, and produces an associated text deck with a filetype of TEXT. It also determines the langid associated with the source file.

VMFTXT EXEC Procedure

The VMFTXT EXEC procedure creates text libraries. VMFTXT rebuilds a named TXTLIB file using a member list in an EXEC file with the same name. The VMFTXT EXEC is invoked as a CMS command as follows:

```
VMFTXT libname [ctlfile]
```

where:

`libname` is the filename of the text library you want to update, and of the EXEC file that contains the names of the library members. The recommended format of the EXEC file is as follows:

```
&TRACE OFF
*Optional comments may be included
 &1 &2 [&3] fn [ft] [(FILename [) ] ]
 &1 &2 [&3] fn [ft] [(FILename [) ] ]
    .
    .
    .
```

where fn and optionally ft, are the filename and filetype of an object file you want to add to the library.

- If you specify a filetype, VMFTXT looks for the specific file.
- If you do not specify a filetype, and you do not specify a CNTRL file, then VMFTXT looks for a filetype of TEXT.
- If you do not specify a filetype, but you do specify a CNTRL filename, VMFTXT searches those filetypes for the specified member.

Each entry in the member list EXEC file may also specify an optional filename parameter (FILename) to be passed directly to the TXTLIB command. This parameter indicates that the member name is to be taken from the filename and not from the CSECT name within the file.

`ctlfile` is the filename of an optional file which VMFTXT uses to determine the filetypes of the object files being added to the text library. The filetype of the *ctlfile* must be CNTRL.

This is usually the same control file used to apply updates to modules using the VMFASM or UPDATE commands. This file identifies the filetype search order if you do not specify the filetype in the member list.

VMFTXT uses the EXEC file to determine which members to include. VMFTXT takes the files from the member list and adds them to the library. They are added in the order they appear in the member list.

If you specify a filetype in the member list, then VMFTXT adds that specific file. Normally you do not specify a filetype, in which case VMFTXT uses the update level identifier in the control file to determine the filetype.

For each member VMFTXT adds, a message is issued verifying this. When all the members have been added, a message is issued stating that member VMFTXT TEXT has been added.

Method of Operation

This section describes the following procedures for generating and updating CP and CMS:

- Update procedure
- Nucleus loading facility
- The MACLIB generation facility.

Figure 8-1 shows the relationship of the diagrams.

Diagram 8-1 shows the major functions of the VMFASM procedure.

Diagram 8-2 shows the initialization of the VMFASM procedure.

Diagram 8-3 describes the assembling portion of the VMFASM procedure.

Diagram 8-4 describes the VMFDATE program.

Diagram 8-5 describes the major functions of the DMSUPD (update) program.

Diagram 8-6 describes the operand and option checking for the Update program.

Diagram 8-7 describes the multiple level update procedure.

Diagram 8-8 describes the processing of control records for the Update program.

Diagram 8-9 describes the single level update procedure.

Diagram 8-10 shows how inserting is done.

Diagram 8-11 describes the exit procedure for the Update program.

Diagram 8-12 describes the module load program.

Diagram 8-13 describes the procedure that builds the MACLIB.

Diagram 8-14 describes the procedure for updating national language related files.

Diagram 8-15 describes the procedure that builds the TXTLIB.

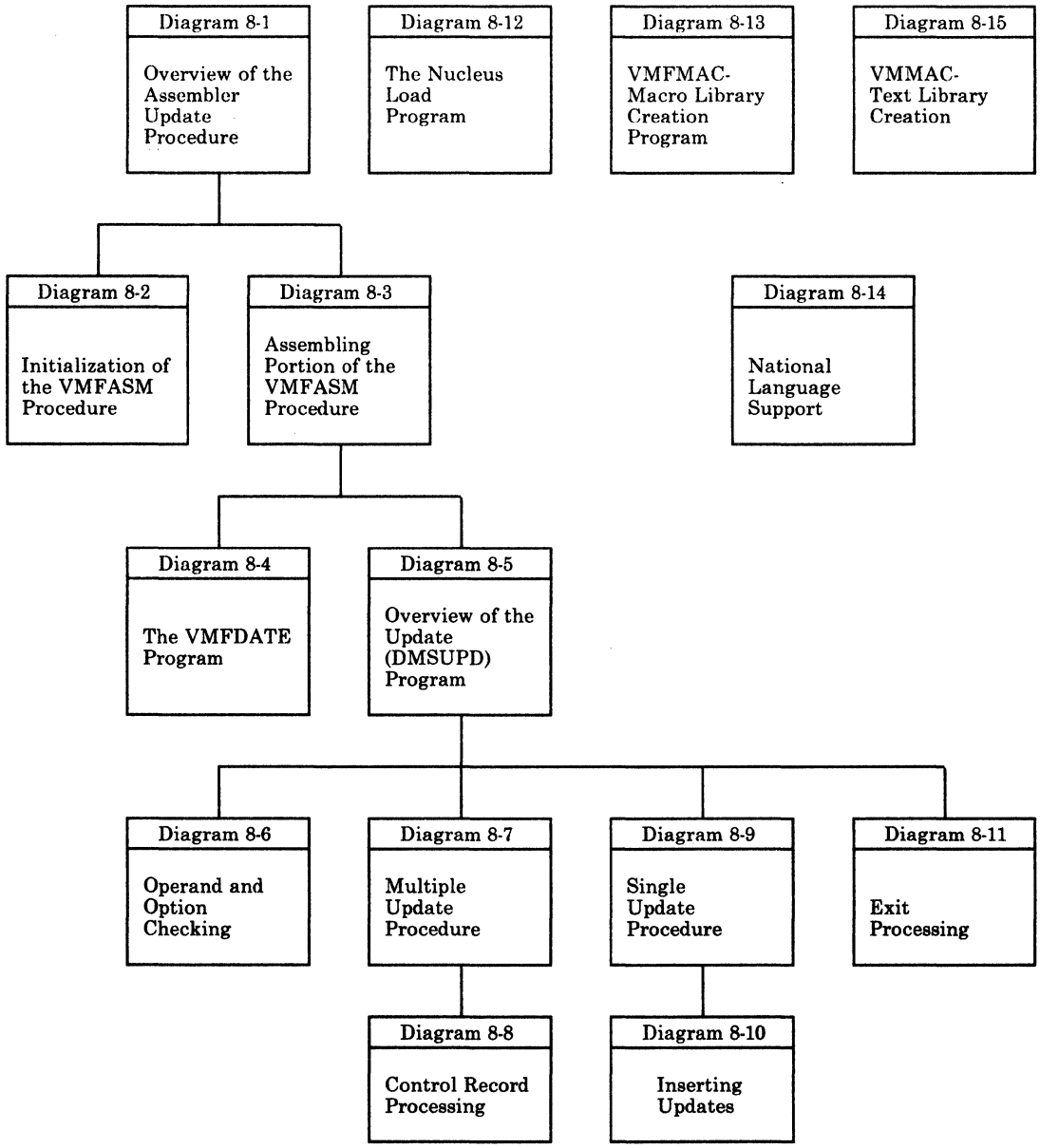


Figure 8-1. Key to the Procedures for Generating and Updating CP and CMS Method of Operation Diagrams

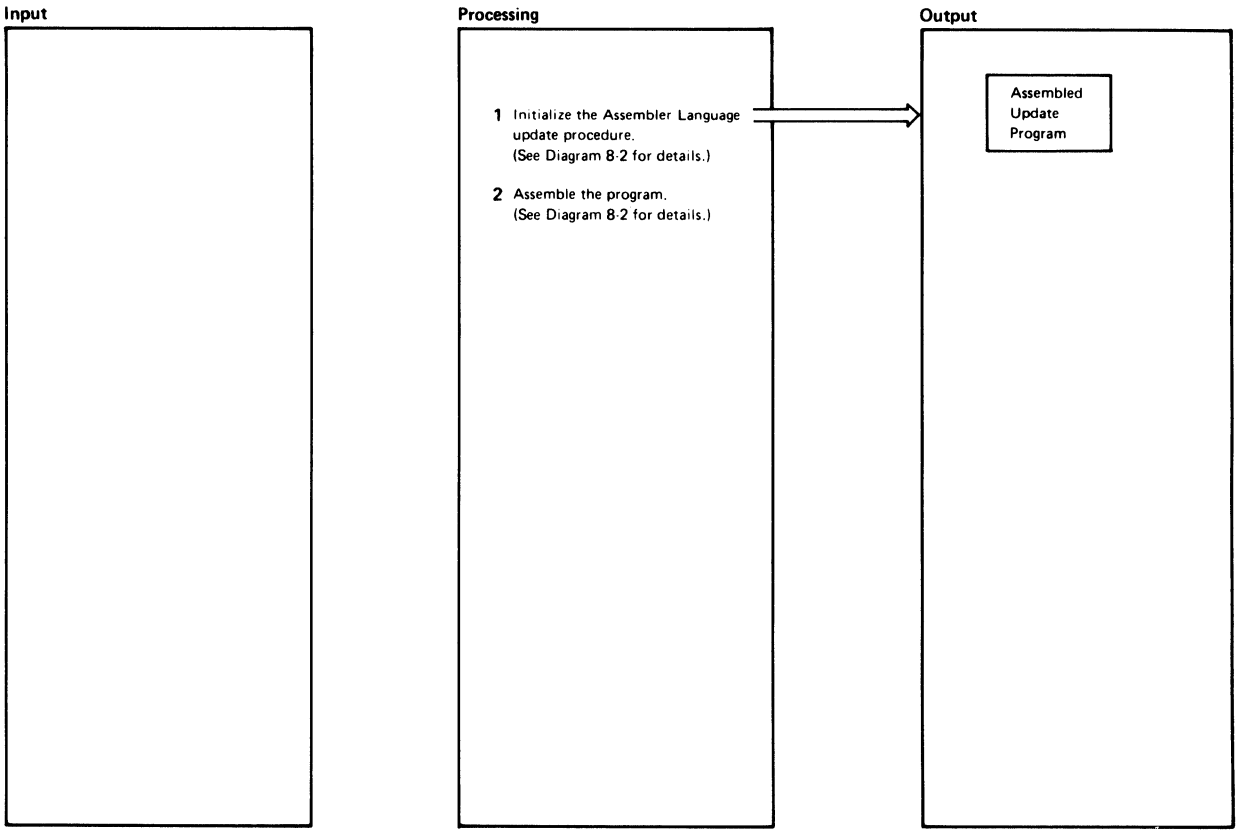
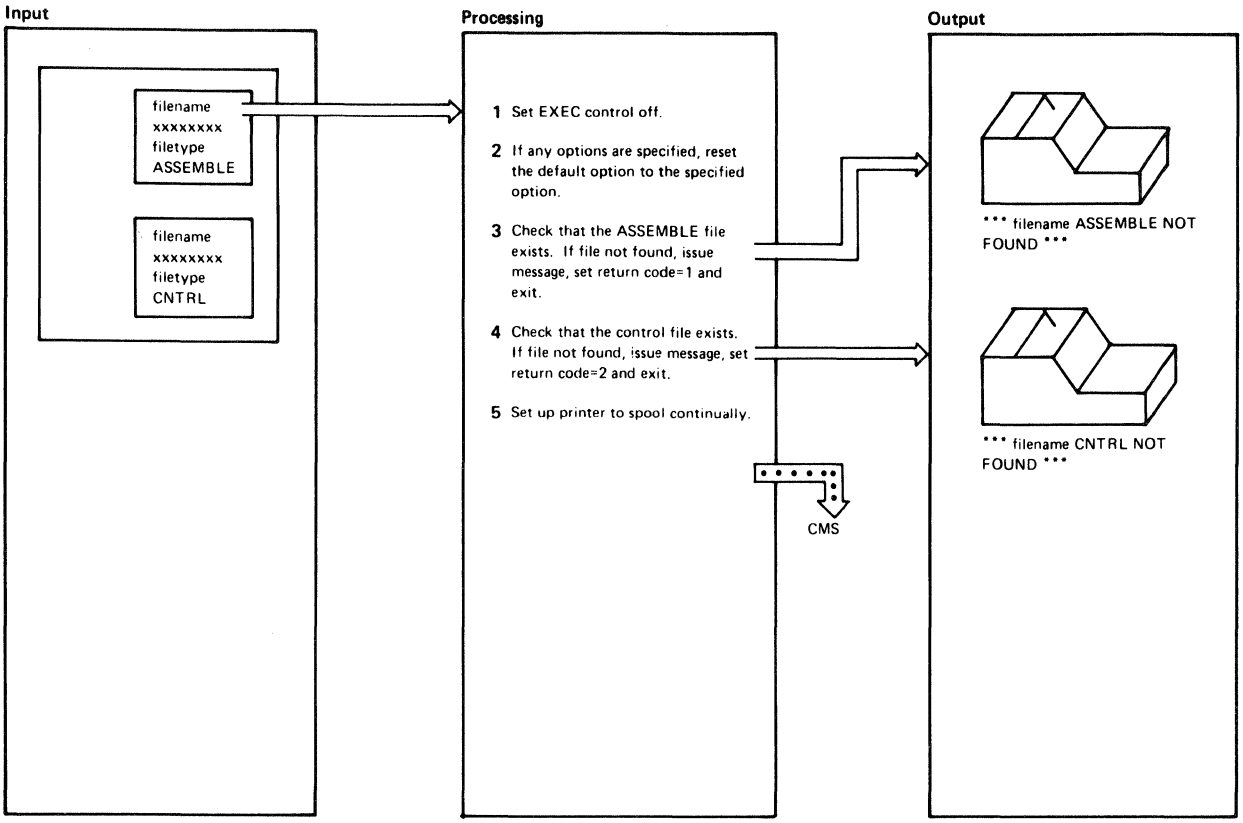


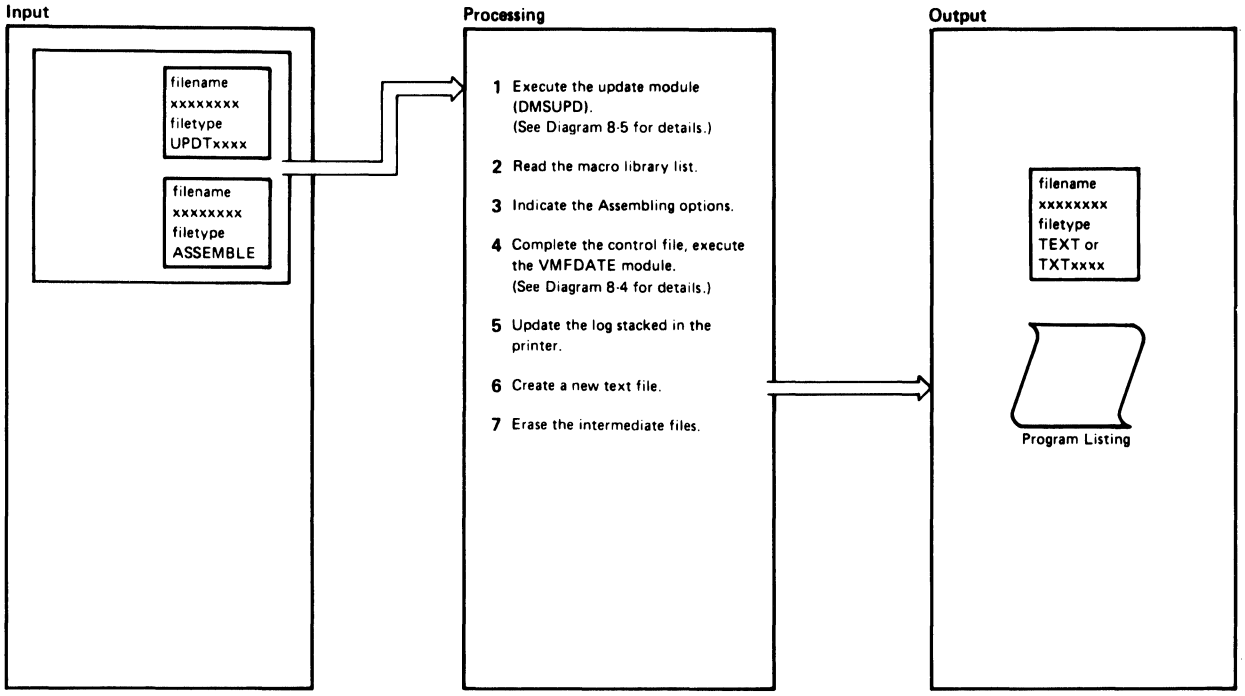
Diagram 8-1. Overview of the Assembler Update Procedure



Notes	Module	Label	Ref
1 The CMS commands executed and the return codes that result will not be displayed on the virtual machine console.	VMFASM		
2 The default options are: PRINT, TERM, LIST, NODECK, NORENT, SYSPARM(SUP), XREF(SHORT), and NORLD. The options specified for the VMFASM EXEC are: DISK, NOTERM, NOLIST, DECK, RENT, EXP, XREF, and RLD.	VMFASM		
3 The CMS STATE command is executed. A nonzero return code indicates that the ASSEMBLE file was not found.	VMFASM	-STSYS	
4 The CMS STATE command is executed. A nonzero return code indicates that the CNTRL file was not found.	VMFASM	-STCTL	
5 The CP SPOOL command is executed.	VMFASM	-FUPD	

Notes	Module	Label	Ref

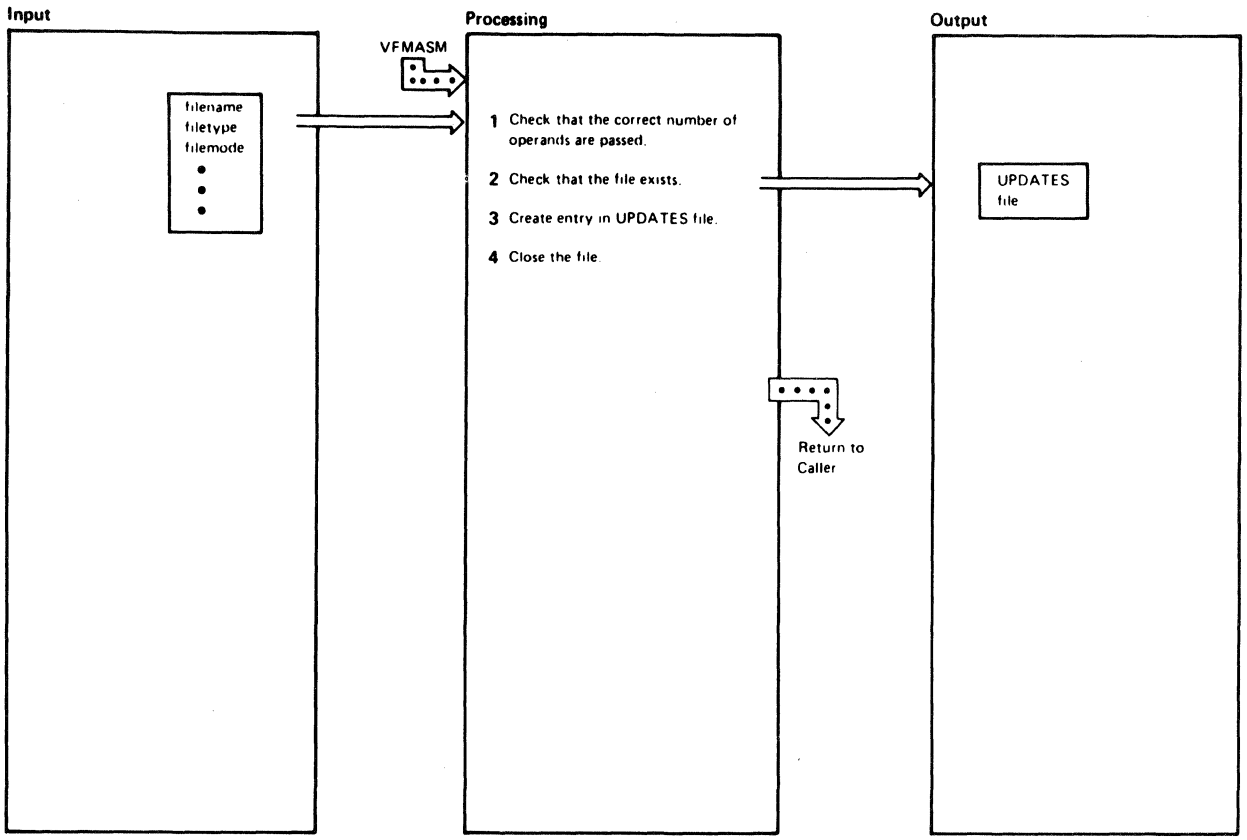
Diagram 8-2. Initialization of the VMFASM Procedure



Notes	Module	Label	Ref
1 The DMSUPD module is executed. The name of the ASSEMBLE and CNTRL files and a filetype of ASSEMBLE are passed to the DMSUPD module. The DMSUPD module returns a level identifier and a MACLIB (macro library) list. A return code between 20 and 36 causes the VMFASM EXEC procedure to display the message *** ERROR UPDATING filename and return control to the CMS command environment. If the level identifier is TEXT, TEXT becomes the filetype of the completed text deck. If the level identifier (xxxxx) is not TEXT, the filetype becomes TXTxxxxx. If the return code is 40 (no updates), the filename is the same as the filename of the original ASSEMBLE file. Otherwise, the filename is set to the updated filename.	VMFASM	FUPD	
2 The MACLIB list is read. The VMFDATE module is executed once for each MACLIB. The CMS GLOBAL command is issued to identify the macro libraries that will be used during the assembly.	VMFASM		
3 If any options were specified on the VMFASM command, the message ASMBLING filename (options . . .) is displayed indicating the specified options.	VMFASM		

Notes	Module	Label	Ref
If no options were specified on the VMFASM command, the default options are assumed and the message ASMBLING filename is displayed.		ASMP	
4 The VMFDATE module is executed once more to complete the UPDATES file.	VMFASM	DTF	
5 The UPDATES file is printed on the virtual printer and then erased.	VMFASM	DTF	
6 The updated file is assembled. If ASSEMBLE returns a nonzero code, the message *** ERROR ASMBLING filename *** is displayed. The STATE command is issued to see if a text deck actually exists. If the text deck does not exist, the message *** NO TEXT FOR filename *** is displayed, the VMFASM EXEC procedure terminates, and control returns to the CMS command environment.	VMFASM	DTF	
7 The new text file, original ASSEMBLE file, and any UPDTxxxx files are saved. The message filename {TEXT / TXTxxxx} CREATED is displayed. All intermediate files are erased. The printer is closed and control returns to the CMS command environment.	VMFASM	COMB EXIT	

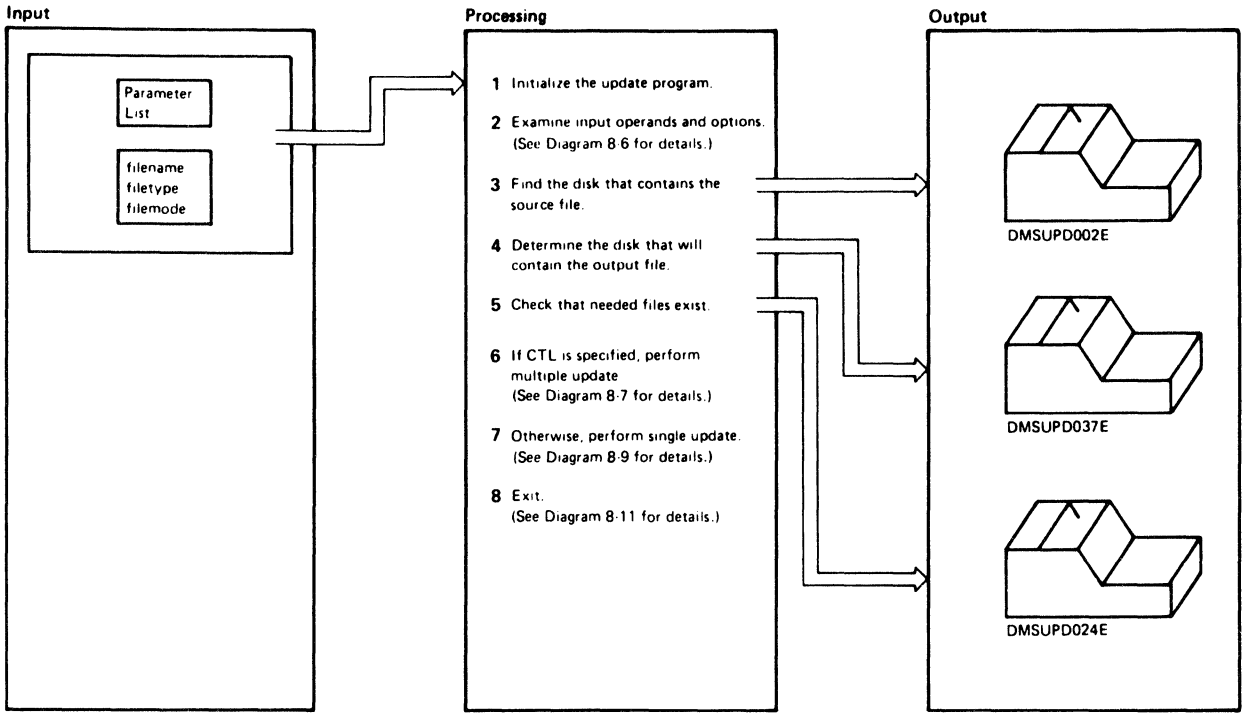
Diagram 8-3. Assembling Portion of the VMFASM Procedure



Notes	Module	Label	Ref
1 Six operands should be passed to the VMFDATE module. The first three operands are the filename, filetype, and filemode of the input file. The next three operands are the filename, filetype, and filemode of the output file.	VMFDATE	VMFDATE	
2 If the input file does not exist, control returns to the calling routine.	VMFDATE	TEST	
3 Each time the VMFDATE module is called, it creates an entry in the VMCNTRL file indicating that an update was applied. The format of each entry is: * filename filetype filemode volid date time The disk label is picked up from the ADT (Active Disk Table).	VMFDATE		
4 The UPDATES file is closed and control returns to the calling routine.	VMFDATE		

Notes	Module	Label	Ref

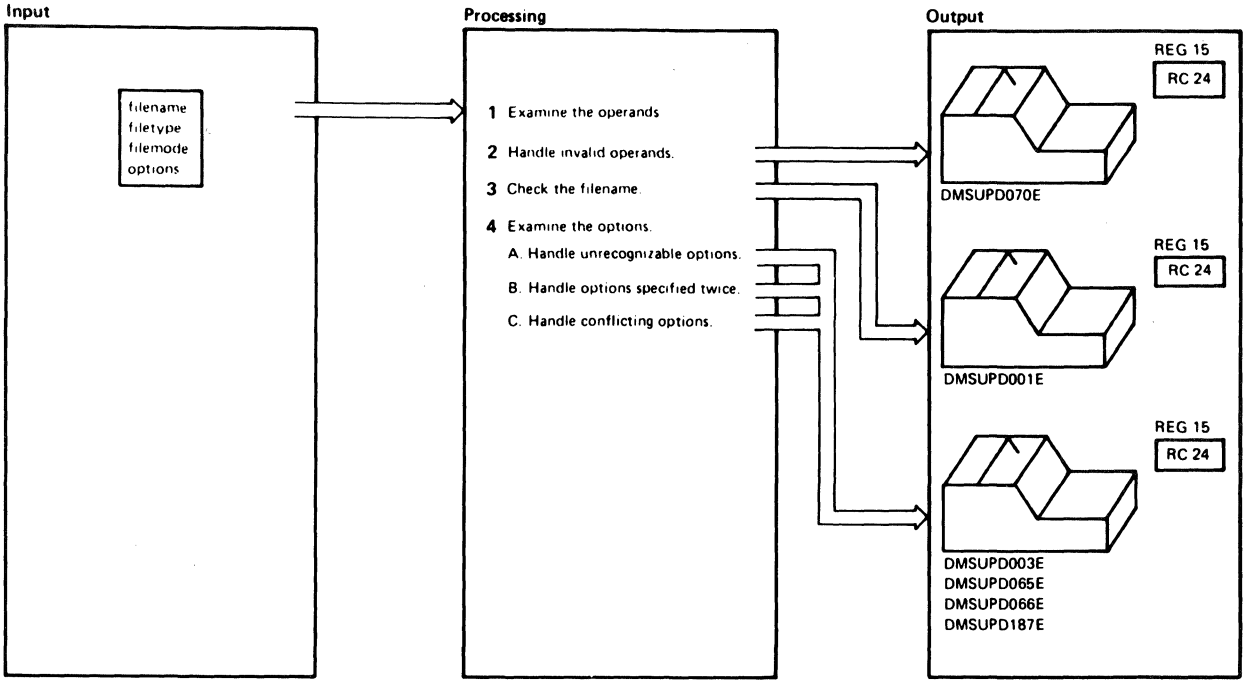
Diagram 8-4. The VMFDATE Program



Notes	Module	Label	Ref
1 Registers 12, 11, and 9 are set up as base registers. All indicators are set off.	DMSUPD	DMSUPD	
2 The filename operand is required.	DMSUPD	DMSUPD	
3 DMSUPD checks that the source input file exists. If not, the message DMSUPD002E FILE 'fn ft fm' NOT FOUND is displayed and control returns to the CMS command environment with a return code of 28 in register 15.	DMSUPD	PROCESS NOFILE	
4 The DMSUPD module searches for a suitable disk to hold the output files. First, an attempt is made to place the files on the same disk that contains the original input. If the input disk is read-only, but is an extension of a read/write disk, an attempt is made to place the files on that disk. Lastly, an attempt is made to place the files on the A-disk. If all these attempts fail, the message DMSUPD037E DISK 'A' IS READ/ONLY is displayed and control returns to the CMS command environment with a return code of 36 in register 15.	DMSUPD	PROCESS	
5 DMSUPD issues the STATE command to see if the UPDATE CMSUT1 file already exists; it should not exist. If the CMSUT1 file exists, the message DMSUPD024E FILE 'UPDATE CMSUT1 fm' ALREADY EXISTS is displayed and control returns to the CMS command environment with a return code of 24 in register 15.	DMSUPD	PROCESS	

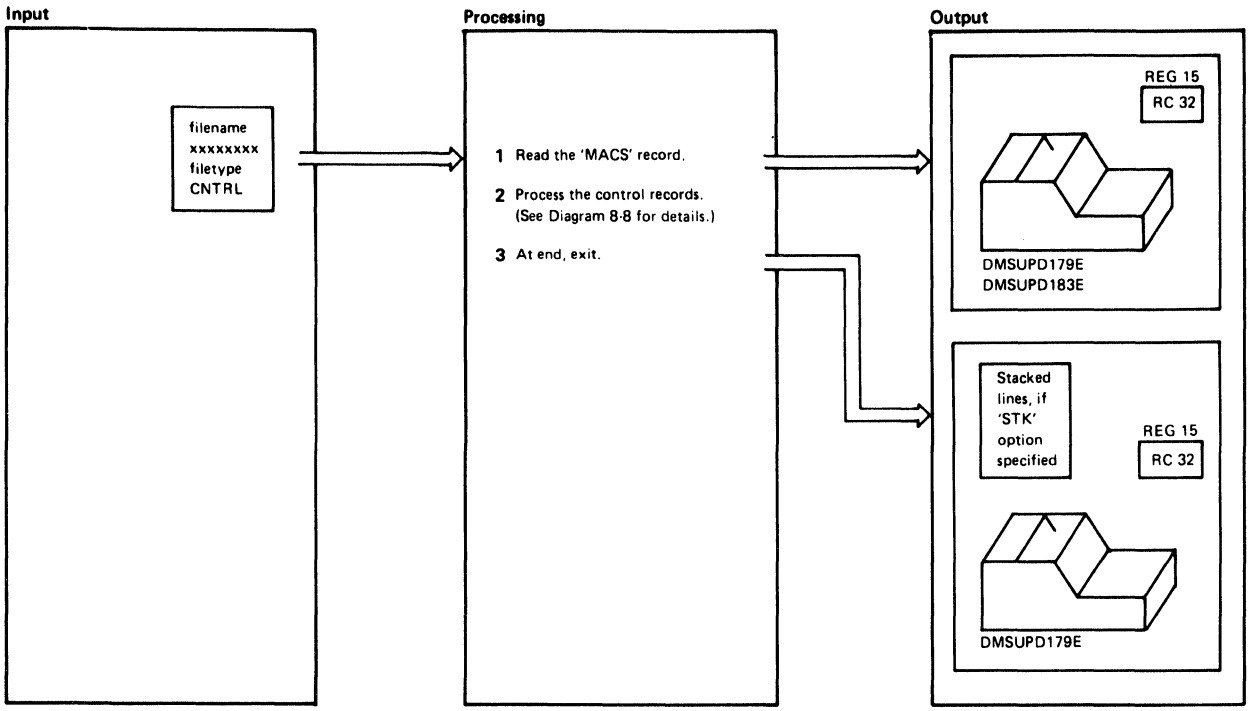
Notes	Module	Label	Ref
If the DISK option wa specified, an old copy of 'filename UPDLOG' is erased (if one exists).			
If the control file option (CTL) is specified DMSUPD checks that the control file exists and continues processing at the CTLMUTL (multiple update) routine.		NOERASE	
If the control file option is not specified, DMSUPD checks that the single update file exists and continues processing at the single update (SINGUPD) routine.		LOCTUPD	
6 See Diagram 8-7.	DMSUPD	CTLMULT	
7 See Diagram 8-9.	DMSUPD	SINGUPD	
8 See Diagram 8-11.	DMSUPD	RETR001	

Diagram 8-5. Overview of the Update (DMSUPD) Program



Notes	Module	Label	Ref
1 DMSUPD uses the filename operand to set up the disk parameter lists for input, update log, and auxiliary files. All the operands (except the required filename) and all the options are read by branching and linking to the OPTSCAN routine. The first three operands are the filename, filetype, and filemode of the file to be updated. The next three operands are the filename, filetype, and filemode that describe the update or control file to be applied.	DMSUPD	DMSUPD	
2 If more than six operands are specified before the left parenthesis, the message DMSUPD070E INVALID PARAMETER 'param' is displayed and control returns to the CMS command environment with a return code of 24 in register 15.		EXCESIV	
3 Only the first operand must be specified. If no operands are found, the message DMSUPD001E NO FILENAME SPECIFIED is displayed and control returns to the CMS command environment with a return code of 24 in register 15.	DMSUPD	NOFNAME	
4 The options assumed, if not otherwise specified are: SEQ8, NOINC, NOREP, NOCTL, NOSTK, TERM, and DISK. When the last option is processed, control returns to the PROCESS routine.	DMSUPD		
A. If an unrecognizable option is specified, the message DMSUPD003E INVALID OPTION 'option'		INVOPTN	
is displayed and control returns to the CMS command environment with a return code of 24 in register 15.			
B. If an option is specified twice, the message DMSUPD065E 'option' OPTION SPECIFIED TWICE is displayed and control returns to the CMS command environment with a return code of 24 in register 15.		OPTDUP	
C. If two conflicting options are specified, the message DMSUPD066E 'option' AND 'option' ARE CONFLICTING OPTIONS is displayed and control returns to the CMS command environment with a return code of 24 in register 15. The conflicting pairs of options are: SEQ8, and NOSEQ8, INC and NOINC, REP and NOREP, STK and NOSTK, TERM and NOTERM, CTL and NOCTL, INC and NOINC, and DISK and PRINT.		OPTCONF	
If the STK option is specified without the CTL option, the message DMSUPD187E OPTION 'STK' INVALID WITHOUT 'CTL' is displayed, and control returns to the CMS command environment with a return code of 24 in register 15.		ERSC	

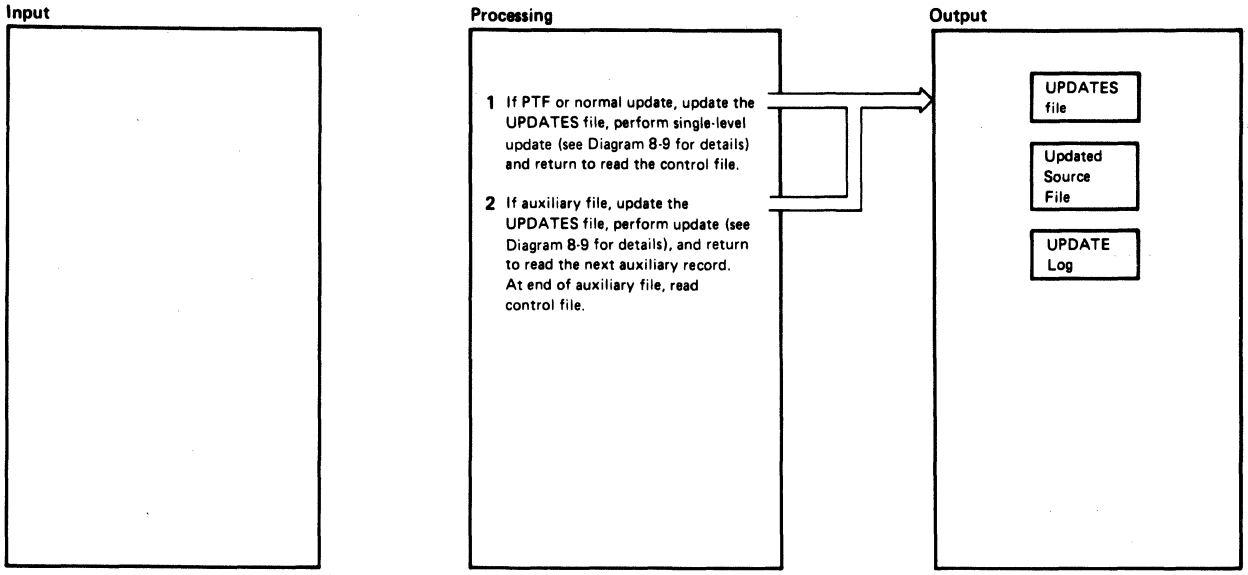
Diagram 8-6. Operand and Option Checking



Notes	Module	Label	Ref
1 The macro library (MACS) record is read from the beginning of the control file and saved. If the MACS card is not found, or is not the first noncomment card in the control file, the message DUMSUPD179E MISSING OR DUPLICATE 'MACS' CARD IN CONTROL FILE 'fn ft fm' is displayed and control returns to the CMS command environment with a return code of 32 in register 15.	DMSUPD	CTMULT ERMACS	
If the MACS control card is invalid, the message DMSUPD183E INVALID CONTROL FILE CONTROL CARD is displayed and control returns to the CMS command environment with a return code of 32 in register 15.		BATCTLC	
2 See Diagram 8-8.	DMSUPD	CTLGETM	
3 If a 'MACS' record is read, the file is completely processed. The control file is closed.	DMSUPD	CTLDONE	
If this MACS card does not have an item number identical to that of the MACS control card originally read, the control file contains duplicate MACS control cards. The message DMSUPD179E MISSING OR DUPLICATE 'MACS' CARD IN CONTROL FILE 'fn ft fm'		ERMACS	

Notes	Module	Label	Ref
is displayed and control returns to the CMS command environment with a return code of 32 in register 15. If STK is specified, the updated level ID is stacked in the terminal input stack.			

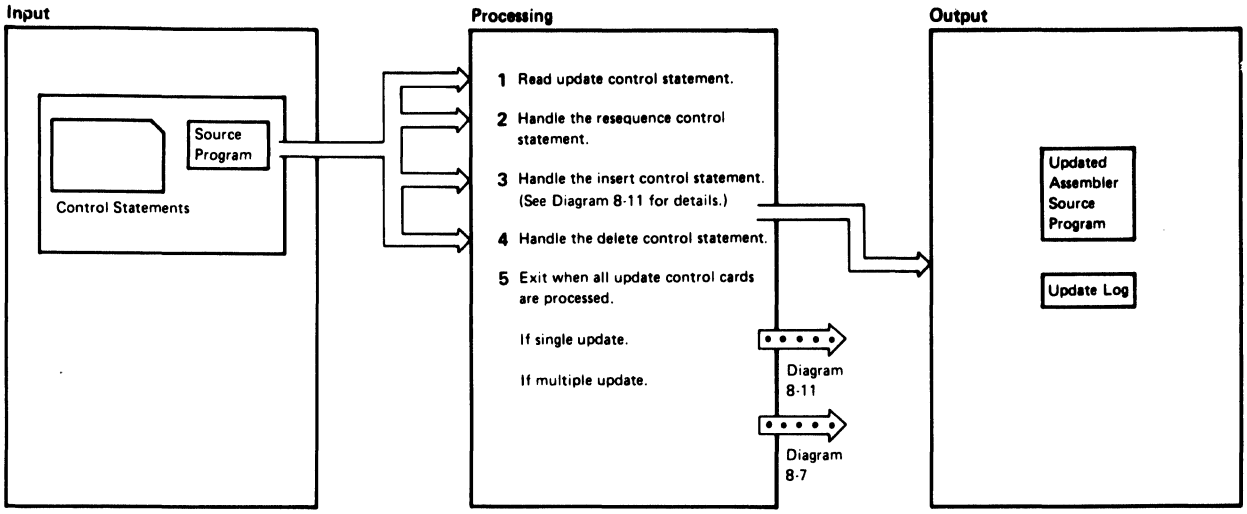
Diagram 8-7. Multiple Update Procedure



Notes	Module	Label	Ref
1 The control file is read from the bottom up. If the control record is valid, the message	DMSUPD	CTLGETM CTLREAD	
DMSUPD183E INVALID CONTROL FILE CONTROL CARD is displayed, and control returns to the CMS command environment with a return code of 32 in register 15.		BADCTLC	
If the PTF or update file is not found, control returns to the read routine (CTLREAD). If the file is found and the update is not being performed in storage, the message		CTLIPTF CTLOCUP	
DMSUPD178I UPDATING 'fn ft fm' WITH 'fn ft fm' is displayed and an entry is made in the UPDATES file. If the update is being performed in storage, free storage is acquired to contain the input file. The message		CTLUMSG CTLUMSS	
DMSUPD300E INSUFFICIENT STORAGE TO BEGIN UPDATE is displayed if the input file is too large for the acquired storage.		SMALLCOR	
If the STOR option was not specified explicitly, the message			
DMSUPD304E UPDATE PROCESSING WILL BE DONE USING DISK is also displayed. If the STOR option was specified, control returns to CMS with a return code of 40 in register 15. If processing continues, the input file is read into the acquired storage, the message		IMPLICIT	
DMSUPD178I UPDATING 'fn ft fm' WITH 'fn ft fm' is displayed, and an entry is made in the UPDATES file.		CTLUMSS	

Notes	Module	Label	Ref
Then a branch to the SINGUPD routine transfers control to the single update routine. After the update is performed, control returns to CTLCONT.			
2 DMSUPD checks that the auxiliary file exists. If not, control returns to the read routine (CTLREAD). If the auxiliary file is found, it is read from the bottom up.	DMSUPD	AUXFIND	
If the PTF file within the auxiliary file is not found, the message		NOFILEW	
DMSUPD180W MISSING PTF FILE 'fn ft fm' is issued to the console and written to the 'fn' UPDLOG. The RETCODE value is set to 12 if it has not been set higher previously. Processing continues with the next record from the auxiliary file (AUXREAD).			
When a valid record is read from the auxiliary file, the message		CTLUMSG	
DMSUPD178I UPDATING 'fn ft fm' WITH 'fn ft fm' is displayed and an entry is made in the UPDATES file. Then the SINGUPD routine applies the update. After the update is performed, control returns to CTLCONT which returns control to AUXREAD. This loop continues until the entire auxiliary file is processed. At the end of the auxiliary file, the file is closed and control returns to the control file read routine (CTLREAD).		CTLUMSS AUXREAD AUXFINT	
If an invalid card is found in the auxiliary file, the message		BADAUXC	
DMSUPD183E INVALID AUX FILE CONTROL CARD is displayed and control returns to the CMS command environment with a return code of 32 in register 15.			

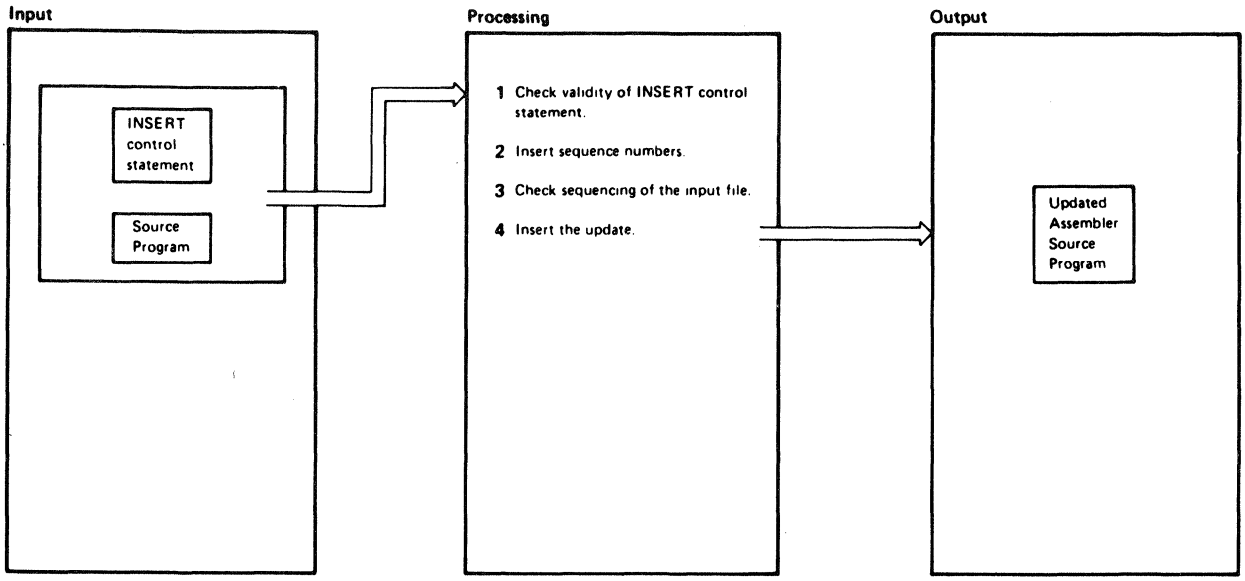
Diagram 8-8. Control Record Processing



Notes	Module	Label	Ref
1 An update card is read and checked. If an invalid control card is read, the message DMSUPD207W INVALID UPDATE FILE CONTROL CARD is issued. The value of RETCODE is set to 12, if it was not previously set higher. Processing continues ignoring the invalid card.	DMSUPD	SINGUPD	
2 DMSUPD checks the resequence card. If the resequence card is not the first card in the update file, the message DMSUPD184W './S' NOT FIRST CARD IN UPDATE FILE – IGNORED is issued. The value in RETCODE is set to 12 if it has not been set higher previously. The './S' card is ignored and processing continues.	DMSUPD	FCTRSEQ RSEQERR	
If an invalid character is specified in one of the sequence fields, the message DMSUPD185W INVALID CHAR IN SEQUENCE FIELD 'xxxxxxxx' is issued. The value of RETCODE is set to 12 if it was not set higher previously. The './S' card is ignored and processing continues.		INVCHAR	
If the specified sequence increment is zero, the message DMSUPD182W SEQUENCE INCREMENT IS ZERO is issued. The value of RETCODE is set to 8 if it has not been set higher previously. Processing continues and the file is resequenced with a sequence increment of zero.		ZERSEQ	
If no errors are found, the sequencing is set to 5 or 8 characters depending on the options specified (SEQ8 or NOSEQ8). The UPDFLAG is set for resequencing and the next update control card is read (UPDREAD).		RSEQDEF RSEQFIN	

Notes	Module	Label	Ref
3 See Diagram 8-11.	DMSUPD	FCTINST	
4 The update control card is checked. The indicated cards are removed. The control statement and the message DELETING . . . are sent to the UPDLOG file. If the delete is being performed in storage, the records in storage are rechained, eliminating the deleted records.	DMSUPD	FCTDELT DELTINE XDELE	
5 When all the update control cards are processed, the UPDREAD (read) routine takes its error exit (UPDFERR). The UPDFERR routine branches to the INPUTRD routine on and end-of-file condition to flush (write out) the rest of the input source file is the update was not performed in storage. If the update was performed in storage, and resequencing is requested, a logical replace is done on each line in the file. The error exit (INPFERR) is taken from the INPUTRD routine. The INPFERR routine closes the updated file and the input file. If processing a control file (multiple update), control returns to CTLCONT. Otherwise, the single-level update is complete and control is returned to CMS (RRETURN exit routine).	DMSUPD	UPDREAD XDELE	

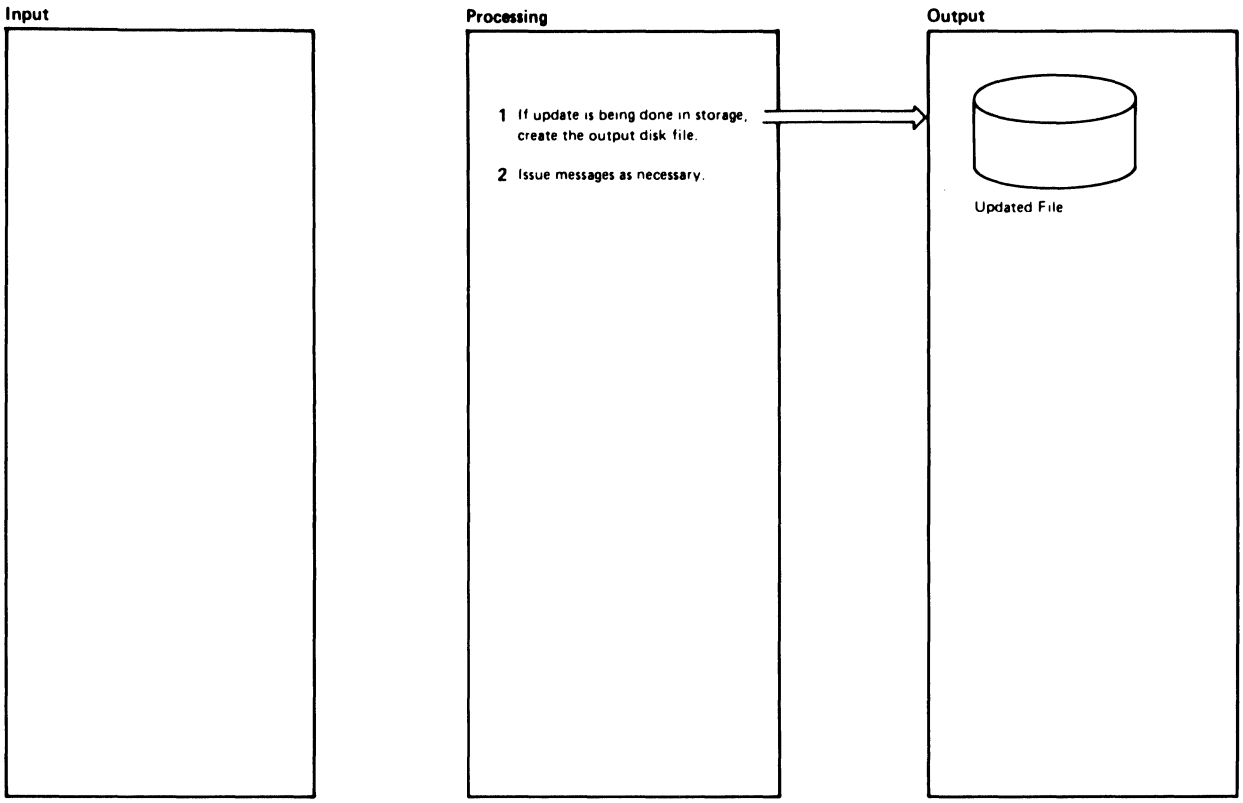
Diagram 8-9. Single Update Procedure



Notes	Module	Label	Ref
1 The INSERT card is checked. If invalid, the message DMSUPD207W INVALID UPDATE FILE CONTROL CARD is issued. The value of RETCODE is set to 12 if was not set higher previously. The invalid card is ignored and processing continues.	DMSUPD	FCTINST INVUPCD	
2 If requested, the sequence numbers are put in the inserts. Otherwise, the sequence number field contains •••••••• If a specified sequence number is not found, the message DMSUPD186W SEQUENCE NUMBER 'xxx' NOT FOUND is issued. The value of RETCODE is set to 12 if it has not been set higher previously. The invalid card is ignored and processing continues.	DMSUPD	FCTREPL UPDSERR	
3 If the input file sequence numbers are out of order, the message DMSUPD210W INPUT FILE SEQUENCE ERROR 'xxx' TO 'xxx' is issued. The value of RETCODE is set to 4 if it was not set higher previously. Processing continues.	DMSUPD	INSEQW	
4 DMSUPD inserts the cards. The control statement and the INSERTING . . . message are sent to the 'UPDLOG' file. If the sequence errors are introduced in the output file, the message DMSUPD174W SEQUENCE ERROR INTRODUCED IN OUTPUT FILE 'xxx' TO 'xxx' is issued. The value of RETCODE is set to 8 if it was not set higher previously. Processing continues.	DMSUPD	INSLOOP	

Notes	Module	Label	Ref
If sequence overflow occurs while cards are being inserted, the message DMSUPD176W SEQUENCE OVERFLOW FOLLOWING SEQUENCE NUMBER 'xxx' is issued. The value of RETCODE is set to 8 if it was not previously set higher. Processing continues. When the appropriate cards are successfully inserted in the file, control returns to the read routine to read the next control card.		WOVF	

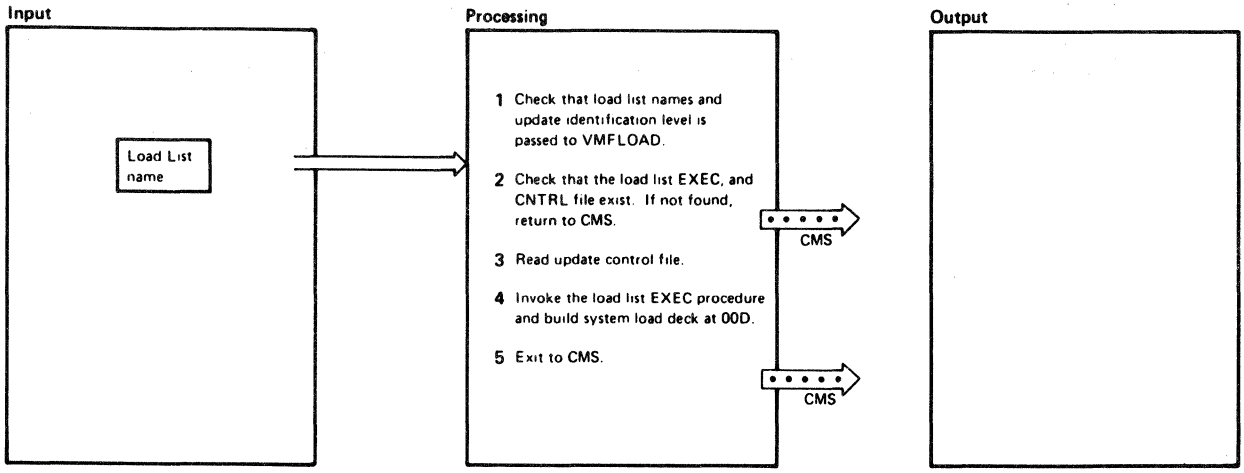
Diagram 8-10. Inserting Updates



Notes	Module	Label	Ref
1 If the update is being performed in storage, the updated file in storage is read line by line and a disk file is created with the filename and filetype UPDATE CMSUT1. The filemode specifies the disk where the final output file resides. The disk file is then closed. The UPDATE CMSUT1 file is then renamed $fname after the old $fname is erased.	DMSUPD	RETR001 RETRD	
2 If RETCODE is not equal to zero, warning messages were issued during the update. If warning messages are issued and the NOTERM option is specified, while the REP option is not, the message DMSUPD177I WARNING MESSAGES ISSUED (SEVERITY=nn) is displayed (nn is the value in RETCODE). If warning messages are issued and the REP option is specified, whether or not the NOTERM option is specified, the message. DMSUPD177I WARNING MESSAGES ISSUED (SEVERITY = nn) 'REP' OPTION IGNORED is displayed (nn is the value of RETCODE). In either case, control returns to the CMS command environment with the value of RETCODE in register 15.	DMSUPD	WRETURN	

Notes	Module	Label	Ref
If no warning messages are issued and the REP option is specified, the '$fname' file is renamed to 'fname', after the old file is erased. If the CTL option is specified and no update files are found, the message DMSUPD181E NO UPDATE FILES WERE FOUND is displayed and control returns to the CMS command environment with a return code of 40 in register 15. If no warning messages are issued, and no errors detected, control returns to the CMS command environment with a return code of 0 in register 15.		NOUPDATS	

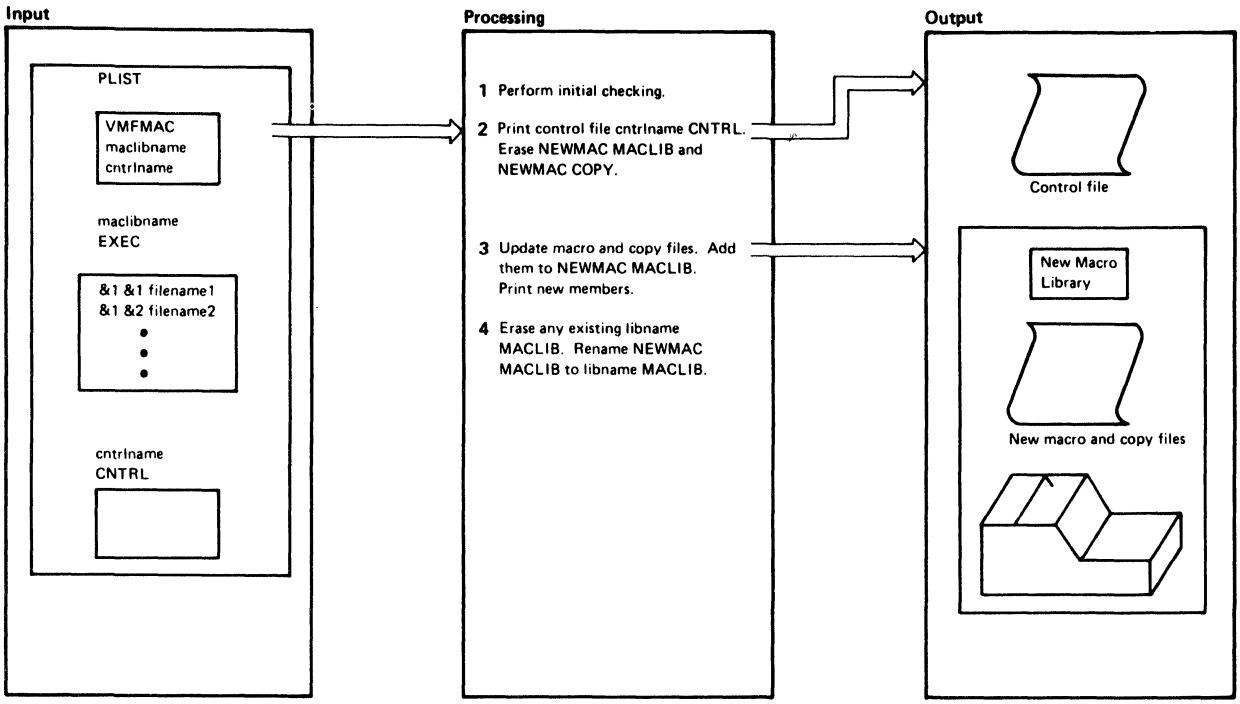
Diagram 8-11. Exit Processing



Notes	Module	Label	Ref
1 The load list name is moved into the filename portion of a STATE command line for an EXEC file and a CNTRL file. The update identification level is moved into the filename portion of a RDBUF command line for an EXEC and CNTRL file. The langid is retrieved from the command line if entered.	VMFLOAD	VMFLOAD	
2 Issue the STATE command via an SVC 202 to make sure that the load list EXEC and CNTRL files exist. If the load list EXEC file is not found, the message NO LOAD LIST is displayed and control returns to the CMS command environment with a return code of 4 in register 15.	VMFLOAD	NOLDL	
If the load list CNTRL file is not found, the message NO CONTROL FILE is displayed and control returns to the CMS command environment with a return code of 2 in register 15.		NOCTR	
3 The first record of the control file is read and the class on the macro library record is saved.	VMFLOAD	DINITA	
The rest of the control file is read. The control records are chained together in the proper hierarchy.		RDCTR	
If an error occurs while reading the control file, the message ERROR IN CONTROL FILE is displayed and control returns to the CMS command environment with a return code of 3 in register 15.		BDCTR	
4 The punch is set to spool continuously. The load list EXEC procedure is invoked by an SVC 202. The text files are punched in the order specified in the load list.	VMFLOAD	DINITB	

Notes	Module	Label	Ref
The resident nucleus modules are loaded first and the pageable modules follow. The DMKLD00E (nucleus loader) resident nucleus module must be loaded first and followed by DMKPSA. The DMKCPE module must be the last resident nucleus module loaded. The pageable nucleus modules are ordered so that they efficiently utilize page frames. The DMKSAV module must be loaded last. When the filename and filetype are both specified, that specific file is searched for and punched, if found. When (LANG is specified, the file filename TXTlangid is searched for and punched. If the langid is not specified, then LANG is ignored. If the file is not found, it is skipped, the message filename filetype NOT FOUND is displayed, and processing continues with the next item in the load list.		NOFILE	
When only the filename is specified, the specified control file is used to search for the highest level text file available. The fist text file located is punched. If the search ends before a text file is found, the "filename TEXT" file is punched if it exists. If the file is not found, it is skipped, the message filename filetype NOT FOUND is displayed, and processing continues with the next item in the load list.		FNDM DINITD SRTXT NOFILE	
This process continues until every item in the load list is processed.			
5 At this point, the text decks are loaded in the proper sequence in the specified reader. All files not found were identified by messages to the terminal. The message SYSTEM LOAD DECK COMPLETE is displayed. The punch is set to stop spooling and is then closed.	VMFLOAD	ENDL	
Control returns to the CMS command environment.		RETERR	

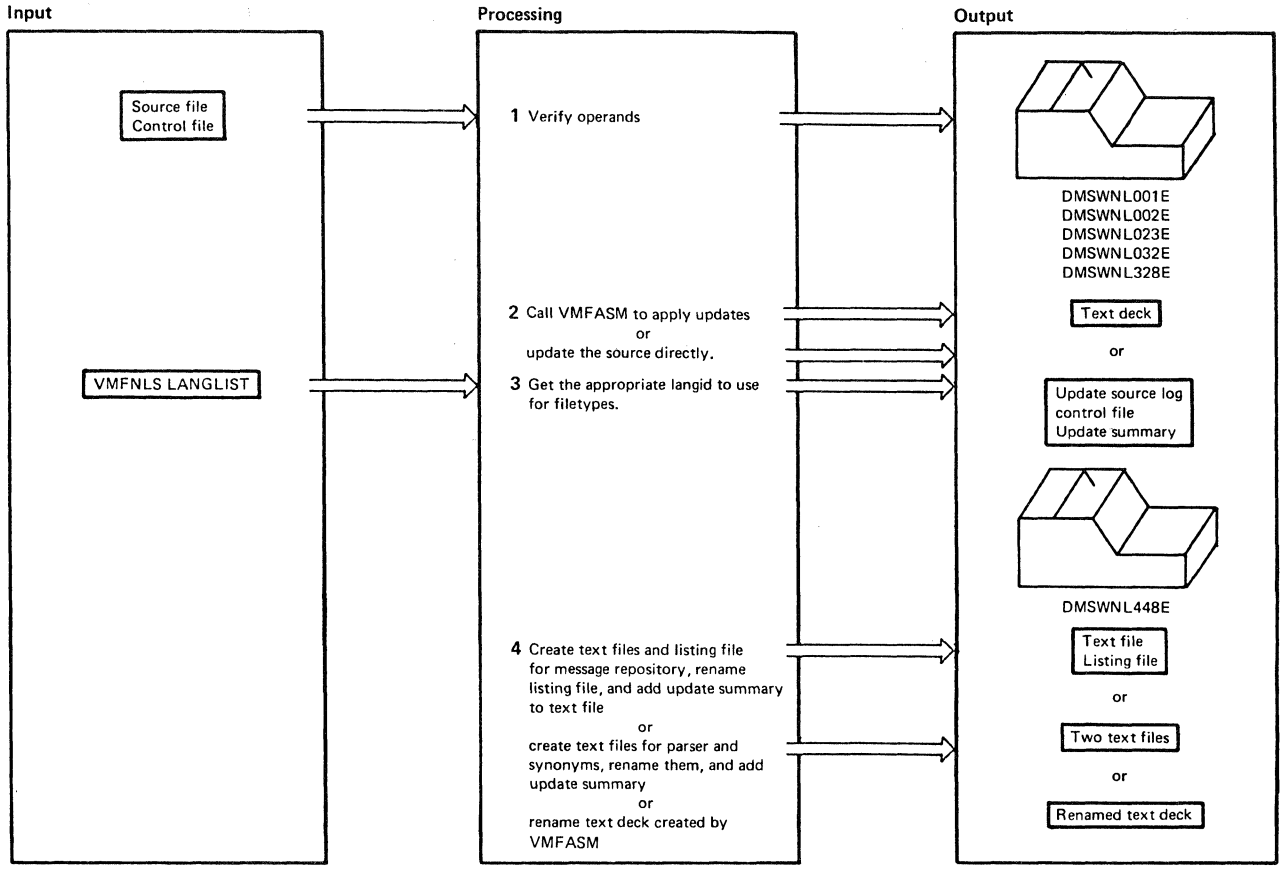
Diagram 8-12. The Nucleus Load Program



Notes	Module	Label	Ref
1 If a list of the members to be put in the macro library (maclibname EXEC) is not found, the message maclibname EXEC NOT FOUND is displayed and control returns to CMS with a return code of 101.	VMFMAC	–ASGN	
If the file containing the updates is not found, the message cntrlname CNTRL NOT FOUND is displayed and control returns to CMS with a return code of 102.		–STCTL	
2 The control file cntrlname CNTRL is printed. The files NEWMAC MACLIB and NEWMAC COPY are erased.	VMFMAC	–STKL	
3 If a macro or copy file is not found, the message ***filename COPY OR MACRO NOT FOUND*** is displayed. The final return code is set to 104 and processing continues with the next member.	VMFMAC	–AREAD	
The UPDATE command is issued for each macro or copy file. If an error occurs, the message ***ERRORS UPDATING membername membertype*** membername membertype NOT INCLUDED IN MACLIB is displayed on the terminal, the files membername UPDATES and membername membertype are printed. The final return code is set to 105 and processing continues with the next member.		–MACUP –UPDERR	

Notes	Module	Label	Ref
If the update procedure is successful, VMFDATE is executed to date stamp the file, and the member is added to the NEWMAC MACLIB. The new member is printed. To maintain a history of the updates that were applied, a line is added to NEWMAC COPY, a dummy copy file.		–MACUP	
4 After all macro and copy files have been processed, the NEWMAC COPY file is renamed to libname COPY and added to NEWMAC MACLIB. Any existing libname MACLIB file is erased and the NEWMAC MACLIB is renamed to libname MACLIB.		–RENEWCO	
If the update procedure is unsuccessful, the message DUE TO PREVIOUS ERRORS, THE RESULT OF THIS MACLIB BUILD IS CALLED 'NEWMAC MACLIB' libname MACLIB HAS NOT BEEN REPLACED is displayed at the terminal and a return is made to CMS with the final return code as previously described.		–ERR2	

Diagram 8-13. VMFMAC—The Macro Library Creation Procedure

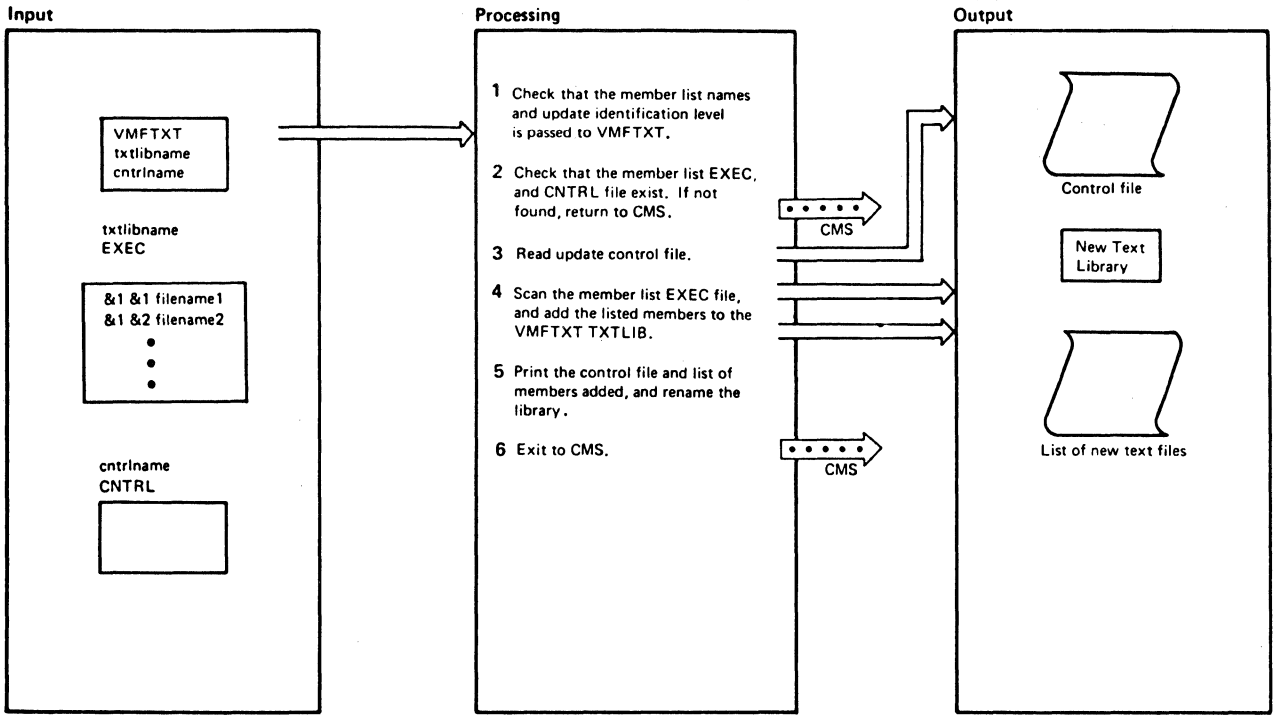


Notes	Module	Label	Ref
1 Checks that: ● A source filename was specified; if it was not, issue the message DMSWNL001E FILENAME NOT SPECIFIED ● A source filetype was specified; if it was not, issue the message DMSWNL023E FILETYPE NOT SPECIFIED ● A control file was specified; if it was not, issue the message DMSWNL328E CONTROL FILE NOT SPECIFIED ● The source filetype is either REPOS, DLCS, or ASSEMBLE; if it was not, issue the message DMSWNL032E INVALID FILETYPE *ft*	VMFNLS		
2 The source file, control file, and VMFNLS LANGLIST file all exist; if one (or more) of them don't, issue the message DMSWNL002E FILE *fn ft* NOT FOUND			
2 If the source filetype is ASSEMBLE, call VMFASM to apply the updates. If the source file is DLCS or REPOS, do the following: ● Update the input source file and print update source log If the update fails, quit the program. ● Print control file and update summary file.			
3 If the source file contains a country code in the 7th and 8th characters of its filename, use this code to search a file called VMFNLS LANGLIST for the appropriate langid; If the country code is not found in VMFNLS LANGLIST, issue the message DMSWNL448E COUNTRY CODE *code* NOT IN VMFNLS LANGLIST If the source filename only has 6 characters, the langid defaults to AMENG.			

Diagram 8-14. VMFNLS—Updating National Language Files (Part 1 of 2)

Notes	Module	Label	Ref
4 If the source filetype is REPOS, do the following: ● Create the message repository text file by issuing GENMSG. (Quit the program if GENMSG fails.) ● Rename the GENMSG listing file so it matches the filename of the input source file, and print the listing. ● Append the summary of updates to the front of the text file, then erase the summary of updates. 5 If the source filetype is DLCS, do the following: ● Create the parser and synonym text files by issuing CONVERT COMMANDS. (Quit the program if CONVERT COMMANDS fails.) ● Rename the text files so they match the filename of the input source file, and print them. ● Append the summary of updates to the front of the text file, then erase the summary of updates. 6 If the source filetype is ASSEMBLE, rename the text deck created by VMFASM.			

Diagram 8-14. VMFNLS—Updating National Language Files (Part 2 of 2)



Notes	Module	Label	Ref
1 Checks that: • The message issuing routine exists. If the file cannot be found, issue the message: DMSWTX002E File *fn ft fm* not found • There are no leftover work files. If the file already exists, issue the message: DMSWTX024E File *fn ft fm* already exists • All required system files exist. If the file(s) cannot be found, issue the message: DMSWTX002E File *fn ft fm* not found • The invocation is correct. If no filename is specified, issue the message: DMSWTX001E No filename specified If the format is incorrect, issue the message: DMSWTX026E Invalid parameter *parameter* for *function* function Verifies that: • The given file names are valid. If they are not, issue the message: DMSWTX062E Invalid character *char* in fileid *fn ft* • The specified files exist. If the file cannot be found, issue the message: DMSWTX002E File *fn ft fm* not found • the A-disk is R/W. If the A-disk is not R/W, issue the message: DMSWTX006E No read/write A disk accessed	VMFTXT	MAIN	

Diagram 8-15. VMFTXT—The Text Library Creation Procedure (Part 1 of 2)

Notes		Module	Label	Ref
	Builds the list of filetypes to be searched. Determines the number of records in the member list. Loops through the entries in the member list. Blank lines, lines beginning with an *, &CONTROL or &TRACE, and words beginning with an & are ignored. The remaining entries are checked for valid format and content. If the entry is valid, the member is added. If the entry is not valid or a minor error occurs while trying to add any member, issue the message: DMSWTX056E File *fn ft [fm]* contains invalid record formats and continue processing the remaining members. If a major error has occurred, quit. Print the CNTL file (if used), and the library contents file (VMFTXT TEXT). Issue the message: DMSWTX895I Member *fn ft* added Add the library contents member to the library. If there have been no errors, erase any existing A-disk copy of the txtlib and rename VMFTXT TXTLIB to the requested name. If there were any errors, issue the message: DMSWTX897E Due to previous errors, the result of this TXTLIB build is called VMFTXT TXTLIB; your *fn* TXTLIB has not been replaced.			
2	Verify that the specified filename/filetype does in fact exist. If found, call ADD_TXT. If not found, issue the message: DMSWTX896E File *fn ft fm* not found.	VMFTXT	WITH_TYPE	
3	Search through the list of filetypes defined in the user specified CNTRL file. If found, call ADD_TXT. If not found, issue the message: DMSWTX896E File *fn* TEXT or *fn* TXT* not found	VMFTXT	FIND_TYPE	
4	If the given filetype is not TEXT and there is a copy of the given file on the A-disk with a filetype of TEXT, then rename the existing TEXT file to VMFTXT CMSUT1 A. Record the original name in VMFTXT CMSUT2 A. If the given filetype is not TEXT, make a copy of the given file on the A-disk with the given filename and a filetype of TEXT. Add the specified file to the VMFTXT TXTLIB. Also, add the date/time stamp information on the new member to the VMFTXT TEXT file. Issue the message: DMSWTX895I Member *fn ft* added If the given filetype is not TEXT, erase the temporary TEXT file from the A-disk, and restore any previous copy the user may have.	VMFTXT	ADD_TXT	
5	If a CTL filename was NOT given on the VMFTXT command, the search list consists of 1 filetype, which is TEXT. Otherwise ... Stack the entire CNTRL file for processing. Determine how many entries are on the stack. Read in the stacked lines, ignoring blank lines or lines beginning with *. If the first token on the line is not TEXT, add a prefix of TXT. Verify that the first record is a MACS record. Check for any extra MACS records. If an error, issue the message: DMSWTX179E Missing or duplicate MACS card in control file *fn ft fm* Verify that all entries are valid filetypes. If there's an error, issue the message: DMSWTX183E Invalid control file control card	VMFTXT	CNTRL	
6	Check whether the filename/filetype parameter contains characters other than those that are valid for a CMS file identifier. Check whether the filename/filetype parameter is longer than 8 characters.	VMFTXT	VERFN	

Diagram 8-15. VMFTXT—The Text Library Creation Procedure (Part 2 of 2)

Program Organization

The procedures for generating and updating VM/SP HPO consist of VMFASM, VMFNLS, VMFMAC, VMFTXT, VMFDATE, DMSUPD, and VMFLOAD.

The Assembler language update procedure consists of the VMFASM EXEC procedure and two modules (VMFDATE and DMSUPD). The VMFASM EXEC procedure sets up for the assembly by calling DMSUPD to create the update control file. There is an entry in the VMCNTRL file for each update control and auxiliary update file. The VMCNTRL identifies the updates applied to the original assembler program and the date and time they were applied.

The Assembler language update procedure calls the VMFDATE program. The MACLIBs needed are then included in the VMCNTRL file.

The nucleus loader procedure consists of a program (VMFLOAD) and an EXEC procedure. Although the DMSUPD update program is not used, the control file that it creates may be used. The LOADER EXEC procedure lists the nucleus modules in the order they are to be loaded. The list includes the filename of each module and may optionally include the update level. If the update level is not specified, the control file created by DMSUPD is used to locate the highest level update available, and that level of the module is loaded.

When nucleus modules are updated and loaded, it is often necessary to create a new macro library. The level of macro library needed for each updated module is recorded in the VMCNTRL file created by the VMFDATE module. The VMFMAC EXEC procedure creates a new macro library. The VMFTXT EXEC procedure rebuilds a TXTLIB file. A member list EXEC file contains the filenames and optional filetypes of the members to be included. For those members that do not specify a filetype, a list of filetypes (provided in the CNTRL file) is searched. This search processing is consistent with the output file created by the VMFASM EXEC procedure using the same CNTRL file.

Directory

Four label directories are provided.

The label directory for the Assembler update function, including labels from:

- The VMFASM EXEC procedure.
- The DMSUPD update program.
- The VMFDATE control file program.

The label directory for the nucleus load program, VMFLOAD.

The label directory for the VMFMAC EXEC procedure, which creates and updates the macro library.

The label directory for the VMFTXT procedure which builds the text library.

Assemble Update Procedure

Label	Module or Procedure	Diagram	Description
-ASMP	VMFASM	8-3	Assumes default options for Assembler.
AUXFINT	DMSUPD	8-7	Closes the auxiliary file when it is completely processed.
AUXREAD	DMSUPD	8-7	Reads auxiliary file from the bottom up.
BADAUXC	DMSUPD	8-7	Processing when invalid card found in auxiliary file.
BADCTLC	DMSUPD	8-7	Abnormally terminates when an invalid control card is encountered.
-COMB	VMFASM	8-3	Saves the new text file, original ASSEMBLE file, and UPDTxxxx files.
CORBUST	DMSUPD	8-10	Insufficient storage to complete update.
CTLDONE	DMSUPD	8-7	Closes the control file once it is processed.
CTLGETM	DMSUPD	8-7	Searches for first control card.
CTLGOT1	DMSUPD	8-7	Checks that auxiliary file exists.
CTLIPTF	DMSUPD	8-7	Checks that PTF file exists.
CTLMULT	DMSUPD	8-5 8-7	Multiple update processing.
CTLOCUP	DMSUPD	8-7	Checks that update file exists.
CTLREAD	DMSUPD	8-7	Reads the control file from the bottom up.
CTLUMSG	DMSUPD	8-7	Updates the UPDATES file.
CTLUMSS	DMSUPD	8-7	Issues the short update message.
DELTINE	DMSUPD	8-9	Deletes cards from the source file.
DMSUPD	DMSUPD	8-6	Entry to update program.
-DTF	VMFASM	8-3	Stacks control file in printer.
ERMACS	DMSUPD	8-7	Processing when MACS card invalid or missing.
ERSC	DMSUPD	8-6	Processing when STK option specified without CTL option.
EXCESIV	DMSUPD	8-6	Error exit when too many parameters are specified.
-EXIT	VMFASM	8-3	Erases intermediate files and returns to CMS.
FCTDELT	DMSUPD	8-9	Checks the delete control card for validity.
FCTINST	DMSUPD	8-9 8-10	Checks the validity of the insert control card.
FCTREPL	DMSUPD	8-10	Checks the validity of the replace control card.
FCTRSEQ	DMSUPD	8-9	Checks the resequence control card.
-FUPD	VMFASM	8-2 8-3	Assembles the updated program.
IMPLICIT	DMSUPD	8-8	Update processing will be done using disk.
INSEQW	DMSUPD	8-10	Processing when sequence errors occur in input file.

Figure 8-2 (Part 1 of 2). The Assembler Update Procedure Label Directory

Label	Module or Procedure	Diagram	Description
INSLOOP	DMSUPD	8-10	Inserts cards from the source file.
INVCHAR	DMSUPD	8-9	Processing for invalid character in sequence field.
INVOPTN	DMSUPD	8-6	Error exit when an unrecognizable option is encountered.
INVUPCD	DMSUPD	8-10	Processing for invalid update file control card.
LOCTUPD	DMSUPD	8-5	Checks that a single update file exists.
NOERASE	DMSUPD	8-5	Checks that the control file exists.
NOFILE	DMSUPD	8-5	Processing when the source input file is not found.
NOFILEW	DMSUPD	8-7	Processing when PTF file not found.
NOFNAME	DMSUPD	8-6	Error exit when no operands were entered.
NOUPDATS	DMSUPD	8-5	Abnormally terminates when update file specified but not found.
OPTCONF	DMSUPD	8-6	Abnormally terminates when conflicting options specified.
OPTDUP	DMSUPD	8-6	Abnormally terminates when the same option is specified more than once.
PROCESS	DMSUPD	8-5	Checks if the update and source input files already exist.
RETRD	DMSUPD	8-11	Creates disk output file from the in-storage updated file.
RETR001	DMSUPD	8-11	Closes and renames the created output disk file.
RETURN	DMSUPD		Checks RETCODE for indication of warning messages.
RSEQDEF	DMSUPD	8-9	Sets the sequencing to 5 or 8 characters.
RSEQERR	DMSUPD	8-9	Issues DMSUPD184W message.
RSEQFIN	DMSUPD	8-9	Sets up for resequencing.
SINGUPD	DMSUPD	8-5 8-9	Applies a single update.
SMALLCOR	DMSUPD	8-8	Insufficient storage to begin update.
-STCTL	VMFASM	8-2	Checks for CNTRL file.
-STSYS	VMFASM	8-2	Checks for the ASSEMBLE file.
TEST	VMFDATE	8-4	Checks for the input file.
UPDREAD	DMSUPD	8-9	Reads control cards.
UPDSERR	DMSUPD	8-10	Issues DMSUPD186W message.
VMFDATE	VMFDATE	8-4	Creates the UPDATES file.
WOVF	DMSUPD	8-10	Issues DMSUPD176W message.
WRETURN	DMSUPD	8-5	Issues DMSUPD177I message.
XDELE	DMSUPD	8-9	Deletes line from storage.
XWRITE	DMSUPD	8-10	Inserts line into storage.
ZERSEQ	DMSUPD	8-9	Issues DMSUPD182W message.

Figure 8-2 (Part 2 of 2). The Assembler Update Procedure Label Directory

VMFLOAD Procedure

Label	Module or Procedure	Diagram	Description
BDCTR	VMFLOAD	8-12	Error exit when error occurs while reading control file.
DINITA	VMFLOAD	8-12	Reads the MACS record from control file.
DINITB	VMFLOAD	8-12	Punches text files.
DINITD	VMFLOAD	8-12	Punches the highest level update available.
ENDL	VMFLOAD	8-12	Closes punch and returns to CMS.
FNDM	VMFLOAD	8-12	Searches for file specified in control file.
NOCTR	VMFLOAD	8-12	Error exit when control file not found.
NOFILE	VMFLOAD	8-12	Skips the files that are not found.
NOLDL	VMFLOAD	8-12	Error exit when loadlist EXEC procedure is not found.
RDCTR	VMFLOAD	8-12	Reads the control file.
RETERR	VMFLOAD	8-12	Exits to CMS.
SRTXT	VMFLOAD	8-12	Punches the TEXT file if update level is not found.
VMFLOAD	VMFLOAD	8-12	Entry for load list program.

Figure 8-3. The VMFLOAD Program Label Directory

VMFMAC Procedure

Label	Module or Procedure	Diagram	Description
-AREAD	VMFMAC	8-13	Checks that each macro or copy file listed in the 'maclibname EXEC' file exists.
-ASGN	VMFMAC	8-13	Checks that the 'maclibname EXEC' file exists.
-ERR2	VMFMAC	8-13	Prints error message if entire update procedure is not successful.
-MACUP	VMFMAC	8-13	Updates the macro or copy files and puts in them the new macro library.
-RENEWCO	VMFMAC	8-13	Renames existing NEWMAC COPY and NEWMAC MACLIB files.
-STCTL	VMFMAC	8-13	Checks that the 'cntrlname CNTRL' file exists.
-STKL	VMFMAC	8-13	Prints the control file.
-UPDERR	VMFMAC	8-13	Prints error message if error occurs during updating.

Figure 8-4. The VMFMAC Procedure Label Directory

The VMFTXT Procedure Label Directory

Label	Module or Procedure	Diagram	Description
MAIN	VMFTXT	8-15	Checks that the invocation conditions are correct, that the 'txtlibname EXEC' file exists, and processes each entry in the 'txtlibname EXEC' file.
WITH-TYPE	VMFTXT	8-15	Checks that those files listed in the 'txtlibname EXEC' file with a filetype do exist.
FIND-TYPE	VMFTXT	8-15	Searches through the list of filetypes defined in the specified CNTRL file until a file with that filetype and given filename is found.
ADD-TXT	VMFTXT	8-15	Adds the files to the VMFTXT TXTLIB.
CNTRL	VMFTXT	8-15	Stacks the CNTRL file, reads the stack, verifies that only one MACS record exists and that it is the first record, and verifies that all entries have valid filetypes.
VERFN	VMFTXT	8-15	Checks that the filename and filetype parameters are valid.

Figure 8-5. THE VMFTXT Procedure Label Directory

Diagnostic Aids

The following figures list all the messages issued by the modules and EXEC procedures that create and update the VM/SP system. Figure 8-6 lists all the messages issued by the VMFASM EXEC procedure, Figure 8-7 lists the messages issued by the DMSUPD module, Figure 8-8 lists the messages issued by the VMFLOAD procedure, Figure 8-9 lists the messages issued by the VMFMAC procedure. The label of the issuing routine and the diagram (if any) describing that routine are included.

VMFASM Procedure

Label	Diagram	Message Text
-FUPD	8-3	***ERROR UPDATING filename***
-ASMP	8-3	ASMBLING filename (options...)
-DTF	8-3	***ERROR ASMBLING filename***
-DTF	8-3	***NO TEXT FOR filename***
-COMB	8-3	TEXT filename TXTxxxxx CREATED

Figure 8-6. VMFASM Messages

DMSUPD Program

Message Code	Label	Diagram	Return Code or Severity	Message Text
DMSUPD001E	NOFNAME	8-6	24	NO FILENAME SPECIFIED
DMSUPD002E	NOFILE	8-5	28	[INPUT|OVERLAY] {FILE[(S)]|DATASET} ['fn [ft[fm]]'] NOT FOUND
DMSUPD003E	INVOPTN	8-6	24	INVALID OPTION 'option'
DMSUPDOO7E	FMTERR		32	FILE 'fn ft fm' [IS] NOT FIXED, 80 CHAR. RECORDS
DMSUPD010W	INPFERR		12	PREMATURE EOF ON FILE 'fn ft fm' --SEQ NUMBER '.....' NOT FOUND
DMSUPD024E	PROCESS ERCMSUT	8-5	24	FILE 'fn ft fm' ALREADY EXISTS [SPECIFY 'REPLACE']
DMSUPD037E	PROCESS ERRW	8-5	36	[OUTPUT] DISK 'mode [cuu]' IS READ/ONLY
DMSUPD048E	BADMODE		24	INVALID MODE 'mode'
DMSUPD065E	OPTDUP	8-6	24	'option' OPTION SPECIFIED TWICE
DMSUPD066E	OPTCONF	8-6	24	'option' AND 'option' ARE CONFLICTING OPTIONS
DMSUPD069E	NOTACCER		32	DISK {'mode'/'cuu'/volid'} NOT ACCESSED
DMSUPD070E	EXCESIV	8-6	24	INVALID {PARAMETER 'param'/ARGUMENT' 'argument'}
DMSUPD104S	NPERR		100	ERROR 'nn' READING FILE 'fn ft fm' FROM DISK
DMSUPD105S	OUTERR		100	ERROR 'nn' WRITING FILE 'fn ft fm' ON DISK
DMSUPD174W	INSLOOP PASSW	8-10	8	SEQUENCE ERROR INTRODUCED IN OUTPUT FILE: 'xxx' TO 'xxx'
DMSUPD176W	WOVF	8-10	8	SEQUENCING OVERFLOW FOLLOWING SEQUENCE NUMBER 'xxx'
DMSUPD177I	WRETURN	8-5	-	WARNING MESSAGES ISSUED (SEVERITY = nn). ['REP' OPTION IGNORED]
DMSUPD178I	CTLUMSG	8-7	-	UPDATING fn ft fm Applying fn ft fm
DMSUPD179E	ERMACS	8-7	32	MISSING OR DUPLICATE 'MACS' CARD IN CONTROL FILE 'fn ft fm'
DMSUPD180W	NOFILEW	8-7	12	MISSING PTF FILE 'fn ft fm'
DMSUPD181E	NOUPDATS	8-5	40	NO UPDATE FILES WERE FOUND
DMSUPD182W	ZERSEQ	8-9	8	SEQUENCE INCREMENT IS ZERO
DMSUPD183E	BADCTLC BADAUXC	8-7	32	INVALID {CONTROL|AUX} FILE CONTROL CARD
DMSUPD184W	RSEQERR	8-9	12	'./S' NOT FIRST CARD IN UPDATE FILE-- IGNORED
DMSUPD185W	INVCHAR	8-9	12	INVALID CHARACTER IN SEQUENCE FIELD seqno
DMSUPD186W	UPDSERR	8-10	12	SEQUENCE NUMBER 'xxx' NOT FOUND
DMSUPD187E	ERSC	8-6	24	OPTION 'STK' INVALID WITHOUT 'CTL'
DMSUPD208W	UPDREAD INVUPCD	8-9 8-10	12	INVALID UPDATE FILE CONTROL CARD
DMSUPD210W	INSEQW	8-10	4	INPUT FILE SEQUENCE ERROR: 'xxx' TO 'xxx'
DMSUPD299E	CORBUST	8-10	40	INSUFFICIENT STORAGE TO COMPLETE UPDATE
DMSUPD300E	SMALLCOR	8-7	40	INSUFFICIENT STORAGE TO BEGIN UPDATE
DMSUPD304I	IMPLICIT	8-7	-	UPDATE PROCESSING WILL BE DONE USING DISK
DMSUPD361E				DISK mode IS NOT A CMS DISK

Figure 8-7. DMSUPD Messages

VMFLOAD Program

Label	Diagram	Message Text
NOFILE	8-12	filename filetype NOT FOUND
BDCTR	8-12	ERROR IN CONTROL FILE
NOCTR	8-12	NO CONTROL FILE
NOLDL	8-12	NO LOAD LIST
ENDL	8-12	SYSTEM LOAD DECK COMPLETE

Figure 8-8. VMFLOAD Messages

VMFMAC Procedure

Label	Diagram	Message Text
-ASGN	8-13	*** maclibname EXEC NOT FOUND ***
-STCTL	8-13	*** cntrlname CNTRL NOT FOUND ***
-AREAD	8-13	*** filename COPY OR MACRO NOT FOUND ***
-UPDERR	8-13	*** ERRORS UPDATING membername membertype *** membername membertype NOT INCLUDED IN MACLIB
-ERR2	8-13	DUE TO PREVIOUS ERRORS, THE RESULT OF THIS MACLIB BUILD IS CALLED 'NEWMAC MACLIB', libname MACLIB HAS NOT BEEN REPLACED

Figure 8-9. VMFMAC Messages

VMFTXT Program

Message Code	Label	Diagram	Message Text
DMSWTX001E	MAIN	8-15	NO FILENAME SPECIFIED
DMSWTX002E	MAIN	8-15	FILE fn ft fm NOT FOUND
DMSWTX006E	MAIN	8-15	NO READ/WRITE A DISK ACCESSED
DMSWTX024E	MAIN		FILE fn ft fm ALREADY EXISTS
DMSWTX026E	MAIN		INVALID PARAMETER parm FOR function FUNCTION
DMSWTX056E	MAIN	8-15	FILE fn ft fm CONTAINS INVALID RECORD FORMATS
DMSWTX062E	MAIN	8-15	INVALID CHARACTER char IN FILEID fn ft
DMSWTXT179E	CNTRL		MISSING OR DUPLICATE MACS CARD IN CONTROL FILE fn ft fm
DMSWTX183E	CNTRL		INVALID CONTROL FILE CARD
DMSWTX895I	MAIN	8-15	MEMBER fn ft ADDED
DMSWTX895I	ADD-TXT	8-15	MEMBER fn ft ADDED
DMSWTX896E	WITH-TYPE		FILE fn TEXT OR fn TXT* NOT FOUND
DMSWTX896E	FIND-TYPE	8-15	FILE fn ft fm NOT FOUND
DMSWTX897E	MAIN		DUE TO PREVIOUS ERRORS, THE RESULT OF THIS TXTLIB BUILD IS CALLED VMFTXT TXTLIB; YOUR fn TXTLIB HAS NOT BEEN REPLACED

Figure 8-10. VMFTXT Messages

VMFNLS Program

Message Code	Label	Diagram	Message Text
DMSWNL001E	FATAL		NO FILENAME SPECIFIED
DMSWNL002E	FATAL		FILE fn ft fm NOT FOUND
DMSWNL023E	FATAL		NO FILETYPE SPECIFIED
DMSWNL032E	FATAL		INVALID FILETYPE ft
DMSWNL122E	FATAL		RETURN CODE rc FROM command
DMSWNL328E	FATAL		CONTROL FILE NOT SPECIFIED
DMSWNL448E	FATAL		COUNTRY CODE code WAS NOT IN VMFNLS LANGLIST

Figure 8-11. VMFNLS Messages

DMKLD00E (Loader) Program

If the loader terminates, one of the following wait conditions is indicated in the instruction counter:

Code	Meaning
X'111111'	A program check occurred.
X'222222'	A unit check occurred while the bootstrap routine was reading in the loader.
X'999999'	An SVC was issued.
X'BBBBBB'	A machine check occurred.
X'CCCCCC'	An I/O error occurred on the card reader.
X'FFFFFF'	An I/O error occurred for the console (X'00' contains the message UNRECOVERABLE ERROR), or the control card for changing the default I/O addresses for the printer or terminal is invalid (X'00' contains the message BAD DEVICE CARD or INVALID DEVICE SPECIFIED).

Loader Wait State Codes

If the instruction counter contains X'999999', indicating an SVC wait state, examine the interruption code (the third and fourth bytes of the supervisor old PSW). The interruption codes (shown in hexadecimal) have the following meanings:

Code	Meaning
64	An error occurred during conversion of a value from hexadecimal to binary format.
65	There is no more free storage available for the loader.
66	A duplicate type 1 ESD (External Symbol Dictionary) entry has been encountered.
67	The "name" in the LDT (Loader Terminate) statement is undefined.
68	The control section named in the ICS (Include Control Section) statement was not found by end of file.
69	The loader attempted to add another entry to the reference table, which would have caused the table to overflow.
6A	The object modules being loaded are about to overlay the loader.

Code	Meaning
6B	The object modules being loaded are about to overlay an address between zero and 100.
6C	A permanent error occurred in the input device.
6D	The loader is trying to release storage that is not on a doubleword boundary.

For further explanations of these wait state conditions and the recommended operator action to correct them, see *VM/SP HPO System Messages and Codes.*

Chapter 9. The VM/SP HPO Starter System

Introduction

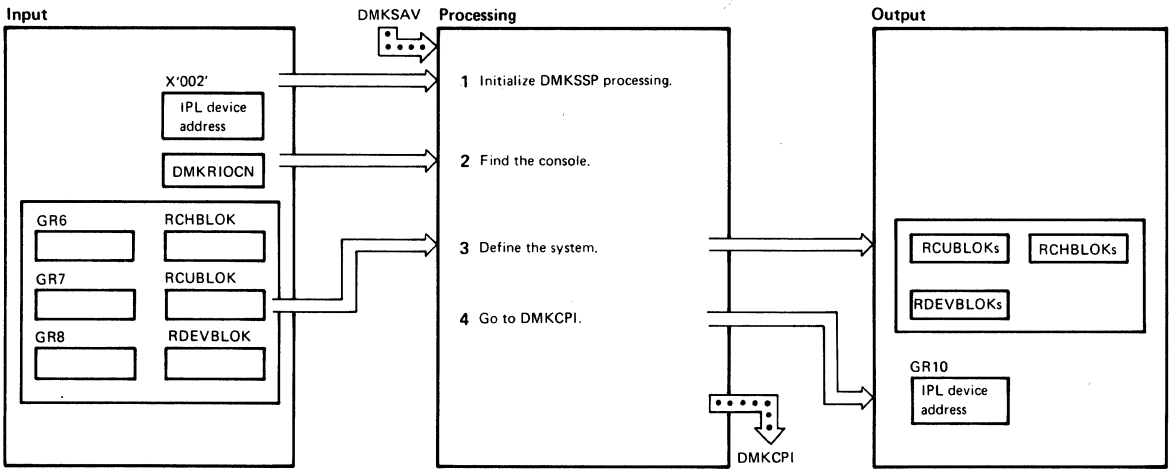
The Starter System Program (DMKSSP) redefines the real configuration according to the operator's specifications.

Normally, VM/SP HPO is loaded from disk; the DMKSAV module reads a copy of the CP nucleus into real storage and then calls DMKCPI to perform the initialization tasks (such as initializing storage, mounting devices, and so on). However, during system generation, the VM/SP HPO starter system is loaded from the starter system tape to disk using DDR. When VM/SP HPO is loaded from the starter system tape, the DMKSAV module reads a copy of the starter system nucleus into real storage and calls DMKSSP to give the operator the opportunity to redefine the devices necessary to continue with system generation. When DMKSSP is through with its processing, it calls DMKCPI to continue the initialization process.

DMKSSP is an interactive program. The operator must signal attention to define a console at an address other than 009 or 01F. Then, the operator responds to questions displayed at the terminal to redefine the printer, punch, reader, tape and disk devices.

Method of Operation

This section describes those functions that are performed by the DMKSSP program. There is only one method of operation diagram and that is Diagram 9-1.



Notes	Module	Label	Ref
1 Registers 11 and 12 are set up as base registers. The new I/O PSW, new machine check PSW, and new program check PSW are set up and all interrupts are disabled.	DMKSSP	DMKSSP01	
2 If the console address is valid, DMKSSP displays	DMKSSP		
VM/SP STARTER SYSTEM		HDRMSG	
***DO YOU WISH TO REDEFINE YOUR SYSTEM *** (YES,NO):		REDEFINE	
If the response is YES, proceed by redefining the system (see step 3). If the response is NO, DMKSSP processing is done. Proceed to step 4.			
3 First, all the control blocks and their pointers are cleared and the system residence device is set up.	DMKSSP	MAINLINE	
DMKSSP must find the console. If the console is not at 009 or 01F, DMKSSP enables for interrupts and waits until the operator signals attention to identify the console. The CPU model is checked and if it is valid, DMKSSP builds the real control blocks for the console, and displays		FINDCONS VLDCOND	
VM/SP STARTER SYSTEM		HDRMSG	
DMKSSP prompts the operator to reconfigure the system, DMKSSP displays		PRTLAB	
ENTER PRINTER ADDRESS (cuu): ENTER DEVICE TYPE (1403, 1443, 3211, 3203, 3262, 3289, 3800):			
and builds the printer real control blocks according to the operator's response.			
DMKSSP displays		PCHLAB	
ENTER DEVICE ADDRESS (cuu): ENTER DEVICE TYPE (2540P, 3525):			
and builds the punch real control blocks according to the operator's response.			

Notes	Module	Label	Ref
and builds the reader real control blocks according to the operator's response.			
DMKSSP displays		PIDLAB	
ENTER ADDRESS WHERE PID TAPE IS MOUNTED (cuu): ENTER DEVICE TYPE (2401, 2415, 2420, 3420, 3430, 8809):			
and builds the tape real control blocks according to the operator's response.			
DMKSSP displays		BKUPLAB	
ENTER ADDRESS WHERE SCRATCH TAPE IS MOUNTED (cuu): ENTER DEVICE TYPE (2401, 2415, 2420, 3420, 3430, 8809):			
and builds the tape real control blocks according to the operator's response.			
DMKSSP displays		SYSLAB	
ENTER DEVICE ADDRESS WHERE SYSTEM RESIDENCE WILL BE BUILT (cuu): ENTER DEVICE TYPE (2319, 3330, 3340, 2305, FB-512, 3380):			
and builds the disk real control blocks according to the operator's response.			
DMKSSP then asks the operator to verify the configuration by displaying ***SYSTEM DEFINITION COMPLETED*** cuu PRINTER cuu PUNCH cuu READER cuu PID TAPE cuu SCRATCH TAPE cuu NEW SYSTEM RESIDENCE cuu SCRATCH PACK ARE THE ABOVE ENTRIES CORRECT (YES, NO):		WORKLAB	
If the operator responds NO, the entire system definition process is repeated.			
4 Control is transferred to DMKCPI with the address of the IPL device in general register 10.	DMKSSP	XPRINT	

Diagram 9-1. DMKSSP—The Starter System

Program Organization

This section describes the organization of the DMKSSP module.

DMKSSP

The Starter System Program that allows the operator to redefine the minimum devices necessary to generate the CP system.

Attributes

Nonreentrant, nonresident, entered via IPL.

Entry Point

DMKSSP001

Entry Conditions

DMKSAV gives control to DMKSSP01. Location X'002' must contain the address of the IPL device.

Exit Conditions

DMKSSP gives control to DMKCPINT to initialize the remainder of the system. Register 10 must contain the IPL device address.

Register Usage

R1: Parameter register
R2: Parameter register
R5: General BAL register
R6: Address of RCHBLOK
R7: Address of RCUBLOK
R8: Address of RDEVBLOK
R11: Base register 2
R12: Base register 1

External References

DMKRIODV — Anchor to the first real device block
DMKRIOCU — Anchor to the first real control unit block
DMKRIOCH — Anchor to the first real channel block
DMKRIOCN — Address of the system console device
DMKRIOPR — Address of the system printer device
DMKRIOPU — Address of the system punch device
DMKRIORD — Address of the system reader device
DMKSYSNU — Disk address on the nucleus
DMKRIO — Address of real I/O control blocks

Call to Other Routines

DMKCVTHB – To convert the device address to binary
DMKCVTBH – To convert the device address to printable hexadecimal characters
DMKCPINT – To continue system initialization

Data Areas

RCHBLOK, RCUBLOK, RDEVBLOK, PSA

Directory

Figure 9-1 is an alphabetic list of the major labels in the Starter System Program. The associated method of operation diagram (if any) is indicated and a brief description of the operation performed at the point in the program associated with each label is included.

Label	Diagram	Description
ATTNHAND	9-1	Enables system for I/O interrupts.
BKUPLAB	9-1	Builds real control blocks for scratch tape.
DMKSSP01	9-1	Starter system entry point called by DMKSAV.
CONINT	9-1	Identifies the system console.
GRAPHID	9-1	Handles the I/O for display terminals.
MSGHAND	9-1	Displays starter system header message.
MAINLINE	9-1	Builds all the real control blocks necessary.
PCHLAB	9-1	Builds the real control blocks for the punch.
PIDLAB	9-1	Builds the real control blocks for the tape drive containing the PID (Program Information Department) distribution tape.
PRTLAB	9-1	Builds the real control blocks for the printer.
RDRLAB	9-1	Builds the real control blocks for the reader.
READADDR	9-1	Initiates writes to and reads from the console to determine the device address.
READTYPE	9-1	Initiates writes to and reads from the console to determine the device type.
REAWRITE	9-1	Writes to and reads from the console. The REAWRITE routine is called by both the READADDR and READTYPE routines.
REDEFINE	9-1	Asks the operator if he wants to redefine the system.
SCAN	9-1	Finds or builds the necessary real control blocks.
STARTIO	9-1	Issues the Start I/O (SIO).
SYSLAB	9-1	Builds the real control blocks for the disk that contains the system residence volume.
WORKLAB	9-1	Asks the operator if the configuration just defined is the one he wants.
XFRINIT	9-1	Transfers control to DMKCPI.

Figure 9-1. The Starter System (DMKSSP) Label Directory

Diagnostic Aids

Figure 9-2 lists the messages issued by the Starter System Program. The associated program label and method of operation diagram are included in the list.

Label	Diagram	Message Text
PRTADDR	9-1	ENTER PRINTER ADDRESS (ccuu):
PRTCLS	9-1	ENTER DEVICE TYPE (1403, 1443, 3211, 3203, 3800, 3289E, 3262):
PCHADDR	9-1	ENTER PUNCH ADDRESS (ccuu):
PCHCLS	9-1	ENTER DEVICE TYPE (2340P, 3525):
DRDADDR	9-1	ENTER READER ADDRESS (ccuu):
RDRCLS	9-1	ENTER DEVICE TYPE (2540R, 2501, 3505):
PIDADDR	9-1	ENTER ADDRESS WHERE FIRST TAPE IS MOUNTED (ccuu):
PIDCLS	9-1	ENTER DEVICE TYPE (3420, 2415, 2420, 2401):
BKUPADDR	9-1	ENTER ADDRESS OF A SECOND TAPE DRIVE (ccuu):
BKUPCLS	9-1	ENTER DEVICE TYPE (3420, 2415, 2420, 2401):
SYSADDR	9-1	ENTER DEVICE ADDRESS OF WORK PACK (ccuu):
SYSCLS SYSDEV	9-1	ENTER DEVICE TYPE (3330, 3350, 3380):
WORKADDR	9-1	ENTER ADDRESS WHERE EXTRA WORK PACE IS MOUNTED (ccuu):
WORKCLS WORKDEV	9-1 9-1	ENTER DEVICE TYPE (3330, 3350, 3380):
GRAFADR	9-1	ENTER ADDRESS OF A GRAPHIC DEVICE (ccuu):
GRFCLS	9-1	ENTER DEVICE TYPE (3277, 3279, 3066):
DEFINE	9-1	***DO YOU WISH TO RE-DEFINE YOUR SYSTEM*** (YES,NO):
DEVUSED	9-1	***ERROR*** DEVICE HAS BEEN ALREADY ALLOCATED
COMPLINE	9-1	***SYSTEM DEFINITION COMPLETED***
LASTLINE	9-1	ARE THE ABOVE ENTRIES CORRECT (YES,NO):

Figure 9-2. The Starter System (DMKSSP) Messages

Chapter 10. The 3704/3705 Service Programs

Introduction

There are four CMS commands and two CP commands specifically for generating and manipulating the 3704/3705 control program. The CMS commands are needed to generate and save a copy of the 3704/3705 control program. The CP commands allow you to operate and manipulate the 3704/3705 in a manner similar to the way other CP commands let you operate your other virtual machine devices.

The CMS commands that help you generate a 3704/3705 control program are: ASM3705, GEN3705, LKED, and SAVENCP. The ASM3705 command is an interface between CMS and the NCP/VS Release 2 and 3 Assembler (IFKASM) or the NCP/VS Release 4 Assembler (CWAX00). It accepts source statement files as input, checks that the input file exists and that the options specified are valid, calls IFKASM or CWAX00 to perform the assembly, and produces an object deck and program listing as output. The ASM3705 command produces the stage 1 output for the 3704/3705 control program generation process.

The GEN3705 command accepts the file produced in stage 1, creates a unique assembler file for each job step in the input file, creates several unique files containing the linkage editor statements necessary to build the load module file, and builds an EXEC macro file of the CMS commands necessary to assemble and load the 3704/3705 control program. If SAVE was specified on the command line, it saves a copy of the control program in page-format on a CP-owned volume.

The LKED command is an interface between CMS and the OS/VS1 linkage editor. The GEN3705 command processor embeds the LKED commands in the EXEC macro file it produces. The LKED command processor interprets the CMS command lines, defines the necessary files, and links to the OS/VS linkage editor. Two permanent files are produced: the 'filename LOADLIB' file, which contains the load modules, and the 'filename LKEDIT' file, which contains the printed output.

The SAVENCP command builds the parameter list (CCPARM) and calls DMKSNC via Diagnose instruction X'50' to write a core image copy of the 3704/3705 control program to a CP-owned system volume. This copy of the control program is loaded each time the 3704/3705 is loaded.

The CP commands that help you to control the operation of the 3704/3705 are NCPDUMP and NETWORK. The NCPDUMP command processor performs several different tasks. It:

- Erases a specific CP or CMS 3704/3705 dump file
- Formats the 3704/3705 dump
- Prints the 3704/3705 dump file
- Assigns an identifier to the 3704/3705 dump file
- Creates the CMS 3704/3705 dump file.

The NETWORK command processor provides the support for the 3704/3705 that several CP commands (ENABLE, DISABLE, QUERY, DISPLAY, VARY, HALT, TRACE, and SHUTDOWN) provide for other devices. In addition, the NETWORK command has options that load a named 3704/3705 control program into 3704/3705 storage and dump the contents of that storage.

These commands are discussed in detail in other publications. For more information about the ASM3705, GEN3705, LKED, and SAVENCP commands and a complete description of the generation process, see the *VM/SP HPO Installation Guide.* For more information about the NCPDUMP and NETWORK commands, see the *Virtual Machine Diagnosis Guide.*

Method of Operation

This section describes the CMS modules that provide the commands to generate the 3704/3705 control programs. Diagrams describe the functions performed by each of the command processors. Figure 10-1 shows the relationships between these diagrams.

Diagram 10-1 describes the SAVENCP command, which saves an image of the 3704/3705 control program so that it can later be loaded. Diagram 10-2 shows how CCPARM is built.

Diagrams 10-3, 10-4, and 10-5 describe the GEN3705 command, which generates a series of commands to assemble, link edit, and load the 3704/3705 control program.

Diagrams 10-6 and 10-7 describe the ASM3705 command, which is an interface between CMS and the NCP/VS Assembler (IFKASM or CWAX00).

Diagram 10-8 describes the LKED command, which is an interface between CMS and the OS/VS1 Linkage Editor.

Diagram 10-9 describes the NCPDUMP command, which prints a dump of the 3704/3705 storage.

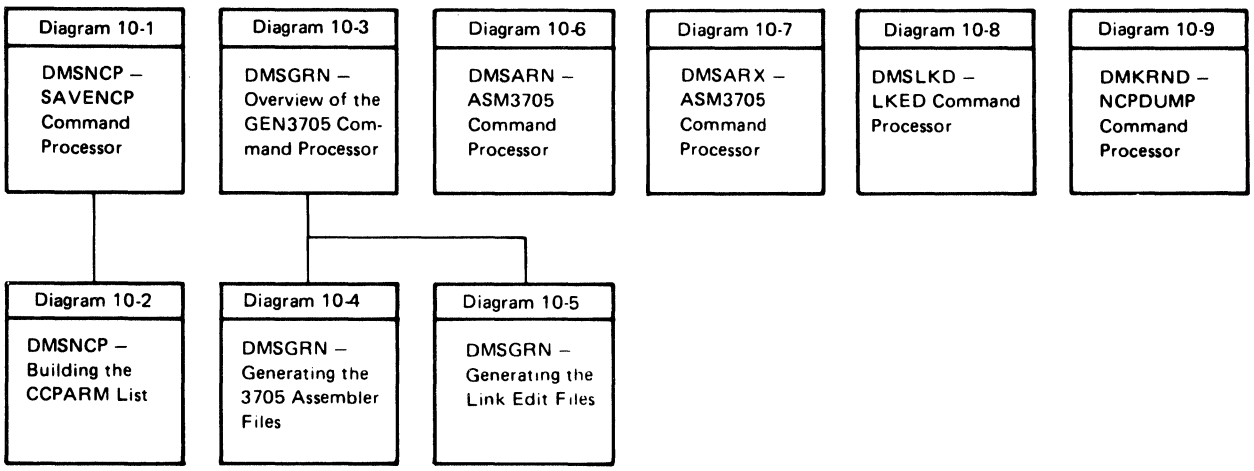
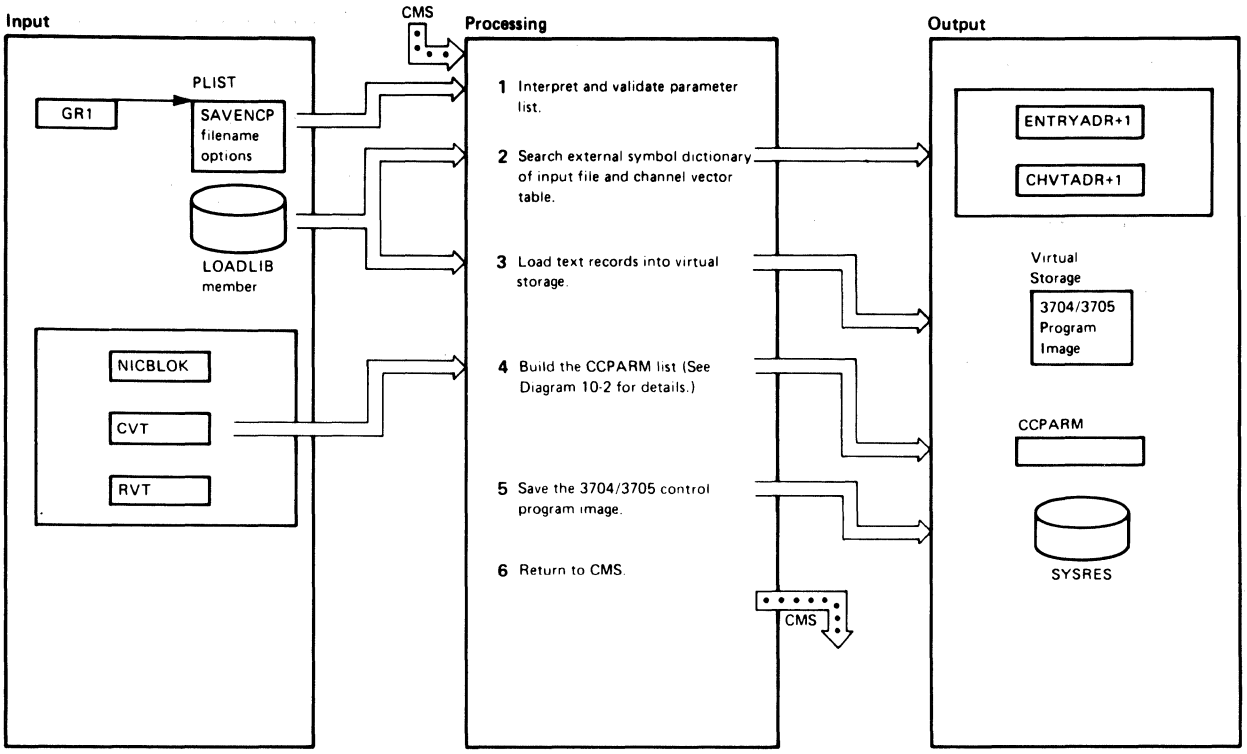


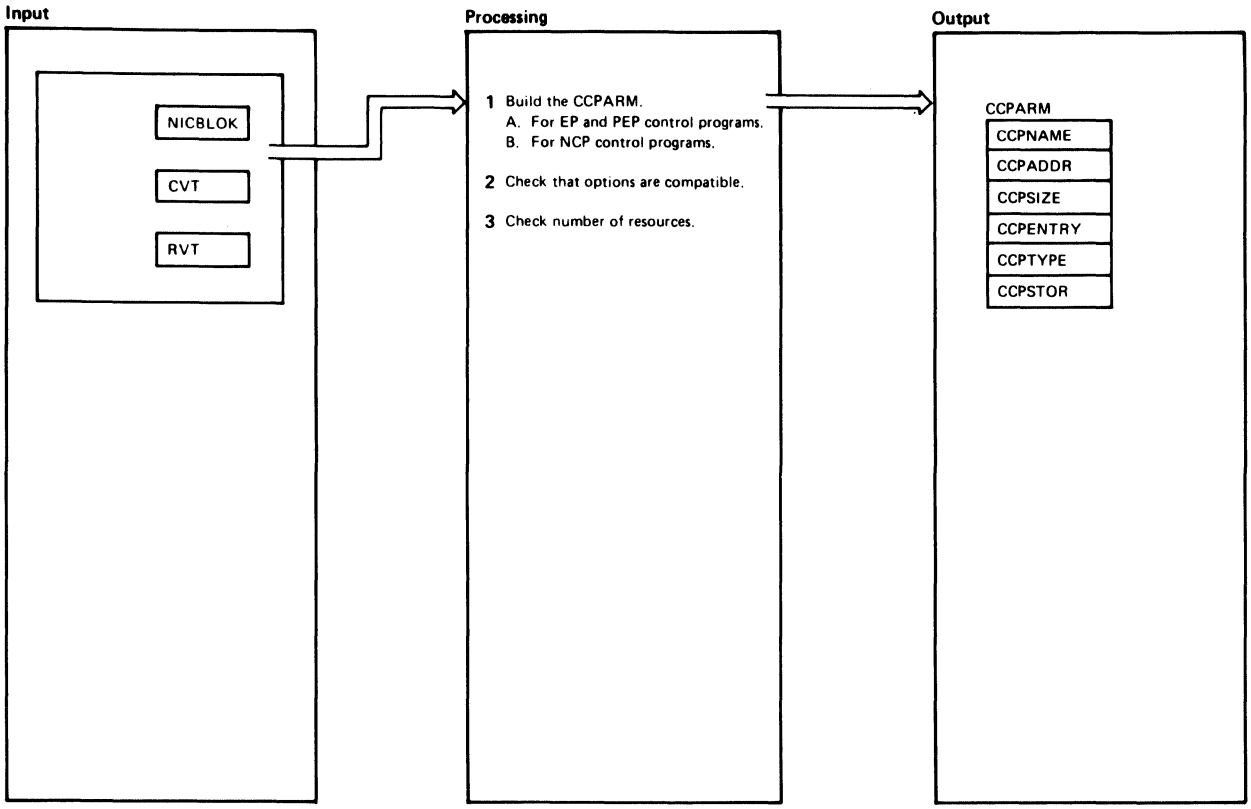
Figure 10-1. Key to the 3704/3705 Service Programs Method of Operation Diagrams



Notes	Module	Label	Ref
1 The filename must be specified. If a library name or a member name is not specified, the input filename is used. If the 3704/3705 control program load module entry point is not specified, CXFINIT is assumed. An error in the parameter list results in one of the following messages DMSNCP001E NO FILENAME SPECIFIED DMSNCP002E FILE 'fn ft fm' NOT FOUND DMSNCP003E INVALID OPTION 'option' being issued and control being returned to CMS with return code 24 or 28. If no errors are encountered, the input file is opened and a search is made for the member. When the member is found, it is read. If the member is not found, the message DMSNCP013E MEMBER xxxxxxxx NOT FOUND IN LIBRARY is issued and control returns to CSM with a return code of 4.	DMSNCP	SAVENCP ENDPARMS DOSTATE	
2 The entry point for NCP or PEP is CXFINIT. The entry point for EP is CYASTART. For either EP or PEP, the channel vector table, CYACHVT CYECHVT1, or CYECHVT2 must also be found. The entry point address and channel vector table address are saved.	DMSNCP	CESDENT CESDCHVT	

Notes	Module	Label	Ref
3 The text records are moved from the input buffer into the proper position in the core image buffer. If the entry point symbol has not been resolved when the first text record is encountered, the message DMSNCP021E ENTRY POINT xxxxxxxx NOT FOUND is issued and control returns to CMS with a return code of 40. Premature end of file or invalid control records cause the messages DMSNCP056E FILE 'fn ft' CONTAINS INVALID RECORD FORMATS DMSNCP109E VIRTUAL STORAGE CAPACITY EXCEEDED to be issued and control to be returned to CMS.	DMSNCP	CONTROL ERR21 ERR66	
4 When the core image buffer is loaded, the input file is closed. The Communication Control Parameter list (CCPARM) is built from the information in the core image buffer.	DMSNCP	CLOSE	
5 The size of the read buffer is stored in register 1 and the DIAGNOSE instruction with code X'50' is issued to save a copy of the 3704 3705 control program.	DMSNCP	SAVECCP	
6 The return code from the DIAGNOSE instruction is passed to CMS and control returns to CMS.	DMSNCP	EXIT	

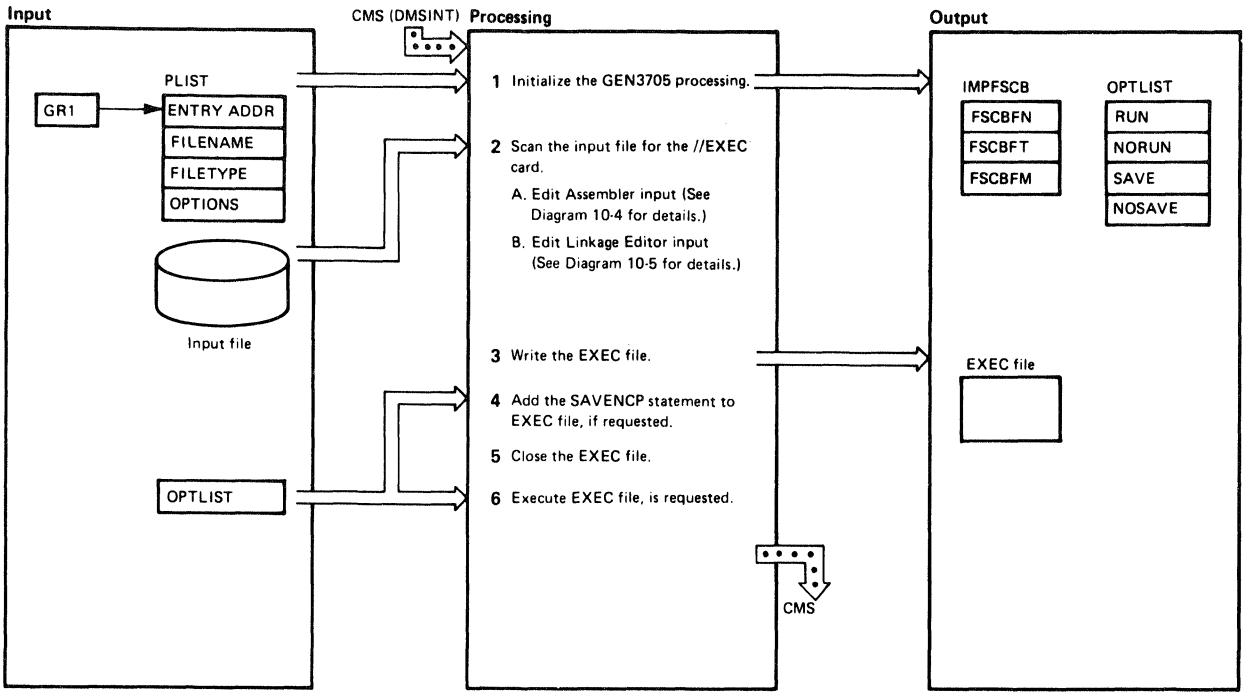
Diagram 10-1. DMSNCP—SAVENCP Command Processor



Notes	Module	Label	Ref
1	DMSNCP		
A. For EP and PEP control programs, additional fields are updated (CCPRSTYP, CCPRSTAT, CCPRSTEP, CCPPSIZE). A channel vector table must exist for EP and PEP control programs. If the CVT does not exist, the message DMSNCP025E INVALID DATA IN 370X PROGRAM is issued and control returns to CMS with return code 16.		SCANCEP	
B. Additional fields in the CCPARM block are updated for NCP and PEP control programs (CCPCAONE, CCPHBFSZ, CCPHBFNO, CCPPADO, CCPPADI, CCPMAXID, CCPRESID, CCPRSTYP, CCPRSTAT CCPRSTEP).		SCANNCP	
2 A check is made that the options specified are compatible. If they are not, the message DMSNCP099W GENERATION PARAMETERS INCOMPATIBLE WITH VM/SP is issued and processing continues.	DMSNCP	CHEKVMV	
3 If there are more than 4086 resources or if the first resource is not a 3704/3705, the message	DMSNCP		

Notes	Module	Label	Ref
DMSNCP025E INVALID DATA IN 370X PROGRAM is issued and control returns to CMS with a return code of 16.			

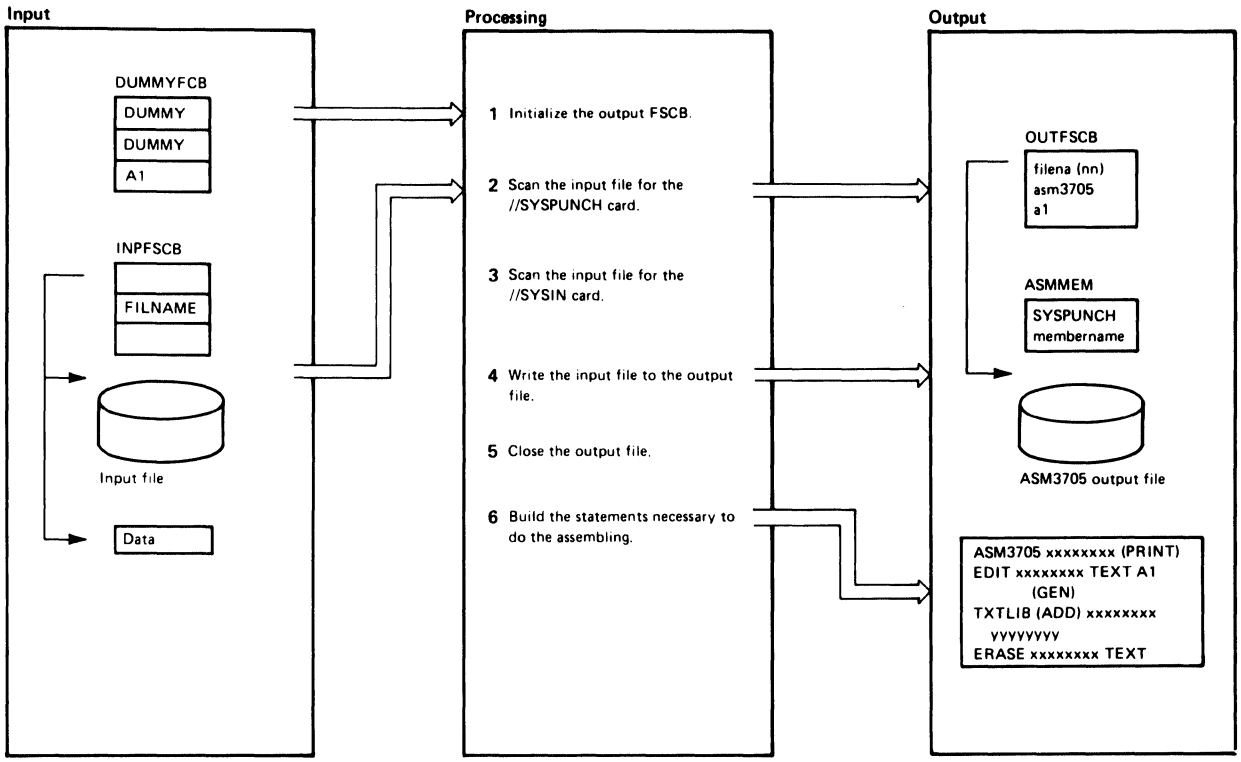
Diagram 10-2. DMSNCP—Building the CCPARM List



Notes	Module	Label	Ref
1 The input file name, type, and optionally the mode are put into INPFSCB. The filename or the first 6 characters of the name, whichever is the least, is saved for naming the assembler and linkage editor output files.	DMSGRN	START	
The input options are scanned and the appropriate options are set on. Invalid opitons cause the message DMSGRN003E INVALID OPTION xxxxxxxx to be issued.		OPTIONS1	
The FSSTATE macro is issued to see if the file exists. Either of the following messages is issued in case of an error DMSGRN048E INVALID MODE xxx DMSGRN002E FILE xxxxxxxx NOT FOUND		OPTEND	
2 The FSCBRD routine is used to read the input file. The EDITIN routine scans for a //EXEC card containing PGM IFKASM or PGM=IEWL. Control cards are scanned until a valid EXEC card is found. If *, //, or /* do not appear as the first characters of the input record or if an invalid //EXEC card is read, the message DMSGRN078E INVALID CARD IN INPUT FILE 'xxxxxxxxxxxxxx' is displayed.	DMSGRN	PRIMEDIT	

Notes	Module	Label	Ref
The IFKASM routine processes the assembler input and the IEWL routine processes the linkage editor input. After the input is processed, DMSGRN continues by scanning the input file for another //EXEC card.		FINDASM FINDIEWL	
3 The EXEC statements that were generated as a result of the assembler and linkage editor input are written to an EXEC file.	DMSGRN	STACK30	
4 The CLOSTACK routine is called to add SAVECP filename (ENTRY entryname to the end of the EXEC file, if SAVE was specified on the GEN3705 command.	DMSGRN	PROCEND2 STACK30	
5 The EXEC macro file is closed by branching and linking to the PROCEND routine.	DMSGRN	PROCEND1	
6 If RUN was specified, the command EXEC ncpname is stacked in the reader.	DMSGRN	PROCEND1	
Control is returned to CMS.		RETURN1	

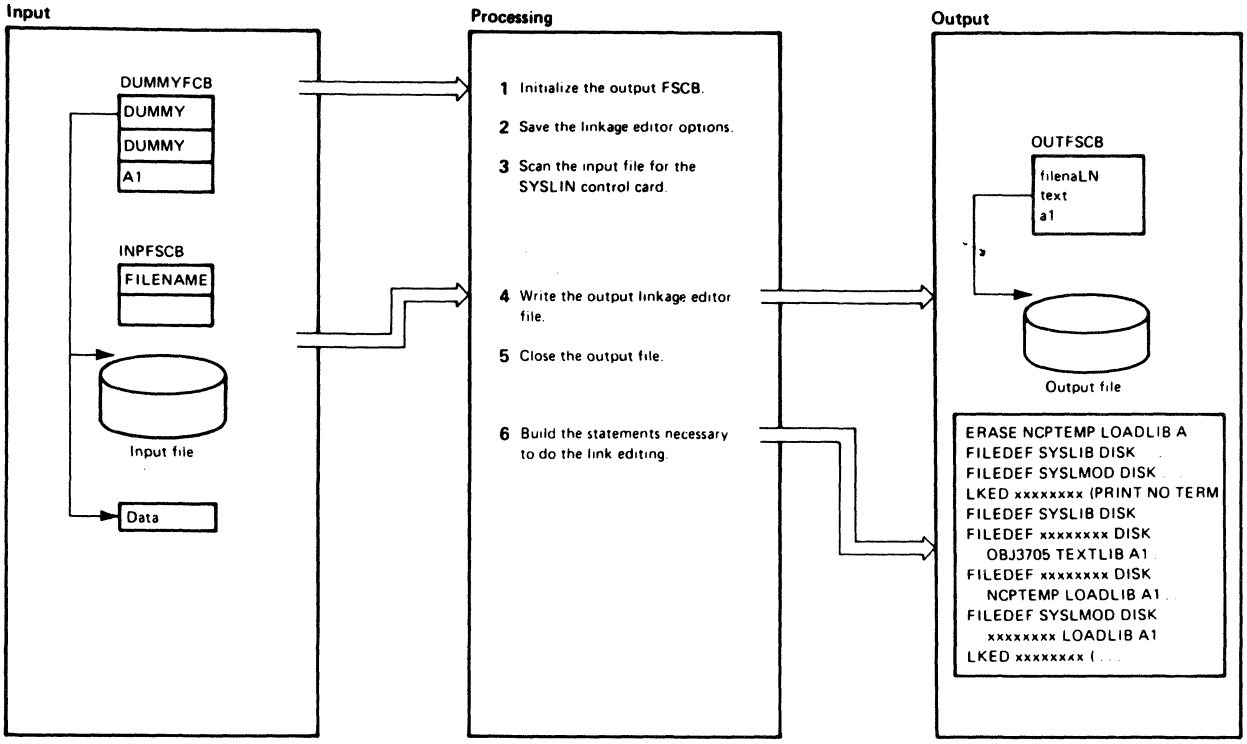
Diagram 10-3. DMSGRN—Overview of the GEN3705 Command Processor



Notes	Module	Label	Ref
1 The filetype in the dummy FSCB is initialized to ASM3705. Each ASM3705 file has a filename consisting of the first 6 characters of the filename (or the entire filename if it is 6 characters or less) concatenated with a number. The FSCBWT routine uses the dummy FSCB to initialize the OUTFSCB.	DMSGRN	IFKASM	
2 The input file is scanned for a SYSPUNCH or SYSPUNCH continuation card. If found, it is scanned for the DSN= or DSNAME=keyword. The DSNEDIT routine then saves the membername of the data set in the current SYSPUNCH membername savearea.	DMSGRN IFKASM34	IFKASM10	
3 The input file is scanned for the SYSIN card. All cards scanned preceding the SYSIN card must have * or // in the first positions of the card. Otherwise DMSGRN078E INVALID CARD IN INPUT FILE 'xxxxxxxxxxxxxxx' is issued.	DMSGRN	IFKASM40	
4 The FSCBRD routine reads all the input and the FSCBWT routine writes it to the output file.	DMSGRN	IFKASMA0	
5 The output file is closed by branching and linking to the FSCBCLOS routine. Close errors are ignored.	DMSGRN	IFKASMK0	

Notes	Module	Label	Ref
6 The ASMFIRST bit in the PROCSW1 byte is tested. If the bit is on, the GEN parameter in the TXTLIB command is changed to ADD. Otherwise, the bit is turned on.	DMSGRN	ASMSTAK	
The name of the output assembler file is moved into the ASM3705 and EDIT commands. The FSCB base address is changed and the name of the input file is put into the TXTLIB command.		ASMSTAK2	
The SYSPUNCH membername is then moved to the TXTLIB command.		ASMSTAK4	
The number of commands and the address of the first command in the stack are loaded from STACKASM into registers 1 and 2 respectively.		ASMSTAK6	

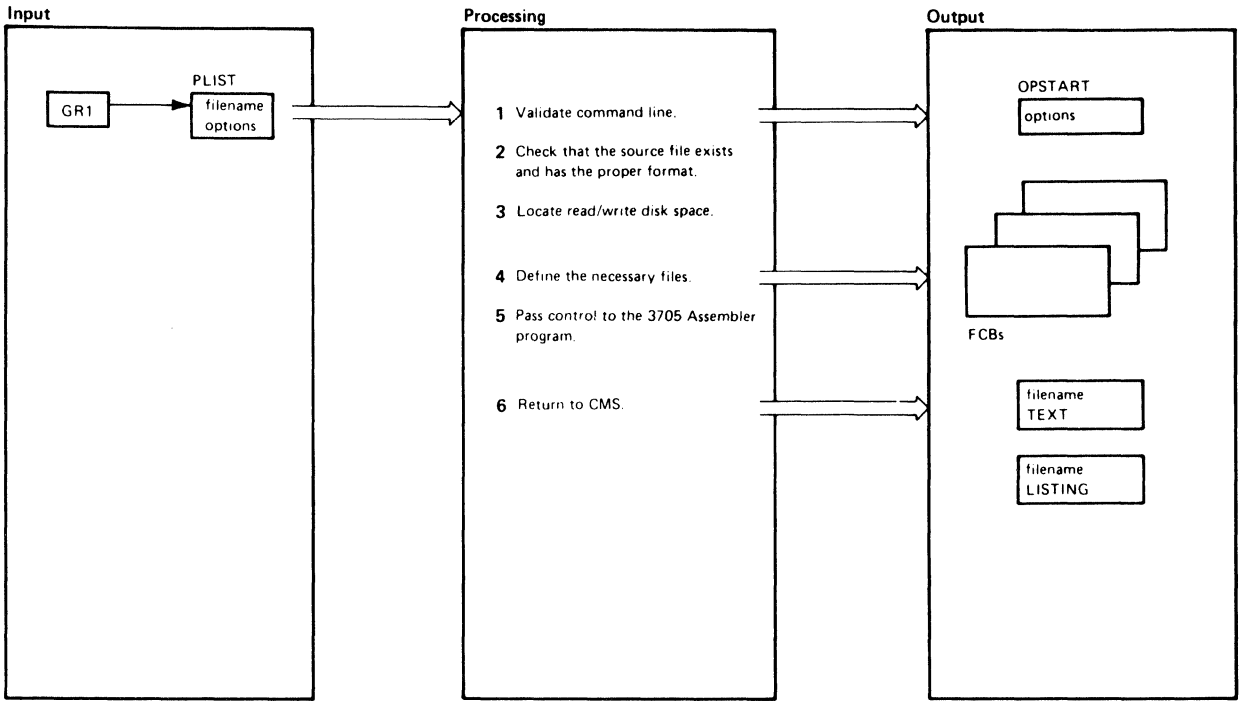
Diagram 10-4. DMSGRN–Generating the 3705 Assembler Files



Notes	Module	Label	Ref
1 The filetype in the dummy FSCB is initialized to TEXT. Each linkage editor TEXT file has a filename consisting of the first 6 characters of the filename (or the entire filename if it is 6 characters or less) concatenated with L and a number.	DMSGRN	IEWL	
2 The //EXEC card is edited for the keyword PARM=. The linkage editor options are moved to the option field of the LKED command. EXEC continuation cards are ignored.	DMSGRN	IEWLJCLA	
3 The input file is scanned for the SYSLIN card. All cards scanned preceding the SYSLIN card must have * or // in the first positions. Otherwise, the error message DMSGRN078E INALID CARD IN INPUT FILE 'xxxxxxxxxxxxxxx' is issued.	DMSGRN	IEWLJCL2	
4 The FSCBRD routine reads the input file and the FSCBWT routine writes it to the output file.	DMSGRN	IEWLSN10	
		WRTSIN	
The EDITIN routine scans for the keyword ENTRY. If the keyword ENTRY is found, the IEWLENT routine moves the entry name to the SAVENCP statement.		IEWLENT	
5 The output file is closed by branching and linking to the FSCBCLOS routine.	DMSGRN	IEWLSEOF	
Close errors are ignored.		FSCBCLOS	

Notes	Module	Label	Ref
6 The LKDFIRST bit in the PROCSW1 byte is tested. If it is off, it is set on and the filename of the input file is moved into the FILEDEF and LKED commands. Also, the command count and address from STAKLKD1 are loaded into registers 1 and 2.	DMSGRN	LKDSTACK	
If the LKDFIRST bit is on, the command count and address from STACKLKD2 are loaded into registers 1 and 2		LKDSTAK1	

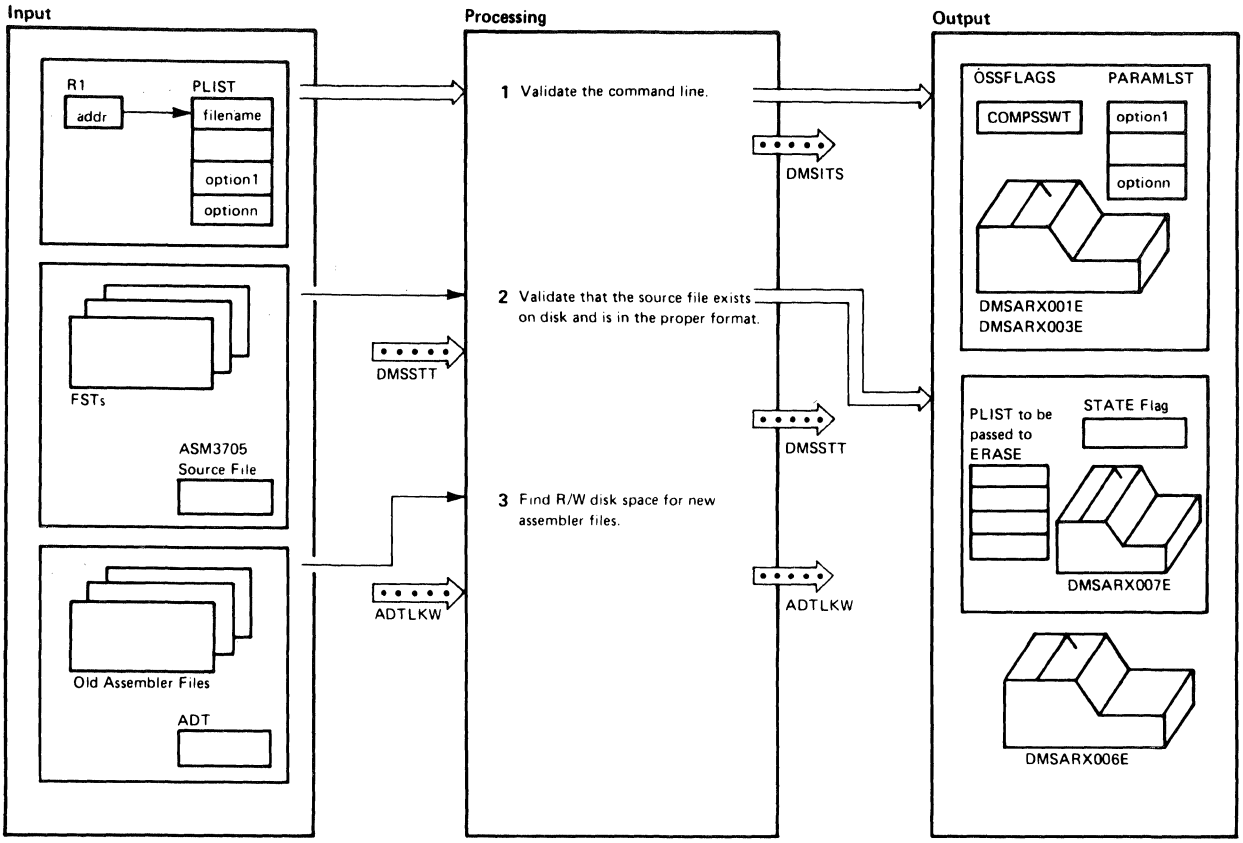
Diagram 10-5. DMSGRN—Generating the Link Edit Files



Notes	Module	Label	Ref
1 A filename must be specified. If it is not, the message DMSARN001E NO FILENAME SPECIFIED is issued and processing terminates.	DMSARN	DMSARN	
The COMPSWT bit is set on in OSSFLAGS to indicate the 3705 assembler is running. The option list to be passed to the 3705 assembler is built.		SQUEEZE	
If Batch is running, the message ASSEMBLING filename A1 is displayed and steps 2 and 3 are skipped.		SUIT15	
2 The STATE macro is issued to check that the input file exists and has fixed 80-character records. If the record format is wrong, the message DMSARN007E FILE filename IS NOT FIXED, 80 CHAR. RECORDS is issued and processing terminates.	DMSARN	SUIT25	
3 If the input file resides on a read/write disk, that disk is used to contain the text and listing files that are generated. If the input disk is an extension of a read/write disk, the parent disk is used. Otherwise, the A disk is used.	DMSARN	SUIT17	

Notes	Module	Label	Ref
4 All the old text, listing, and utility files for the current file are erased. Free storage is initialized and enough storage to contain the longest assemble path is obtained via a GETMAIN call	DMSARN	CONTINUE	
FILEDEFs are issued for SYSUT1, SYSUT2, SYSUT3, SYSIN, TEXT, SYSPUNCH (if the DECK option was specified), SYSPRINT (if the NOPRINT option was not specified), LISTING, and CMSLIB		NOERASE	
5 Control is passed to IFKASM	DMSARN	LIST2	
6 If the return code is not zero, one of the following messages is issued DMSARN004W WARNING MESSAGE ISSUED DMSARN008W ERROR MESSAGES ISSUED DMSARN012W SEVERE ERROR MESSAGES ISSUED DMSARN016W TERMINAL ERROR MESSAGES ISSUED	DMSARN	RETURN	
The output files are closed and the utility files SYSUT1, SYSUT2, and SYSUT3 are erased. All FCBs are cleared, OSSFLAGS is reset, and control returns to CMS.		SUIT19	

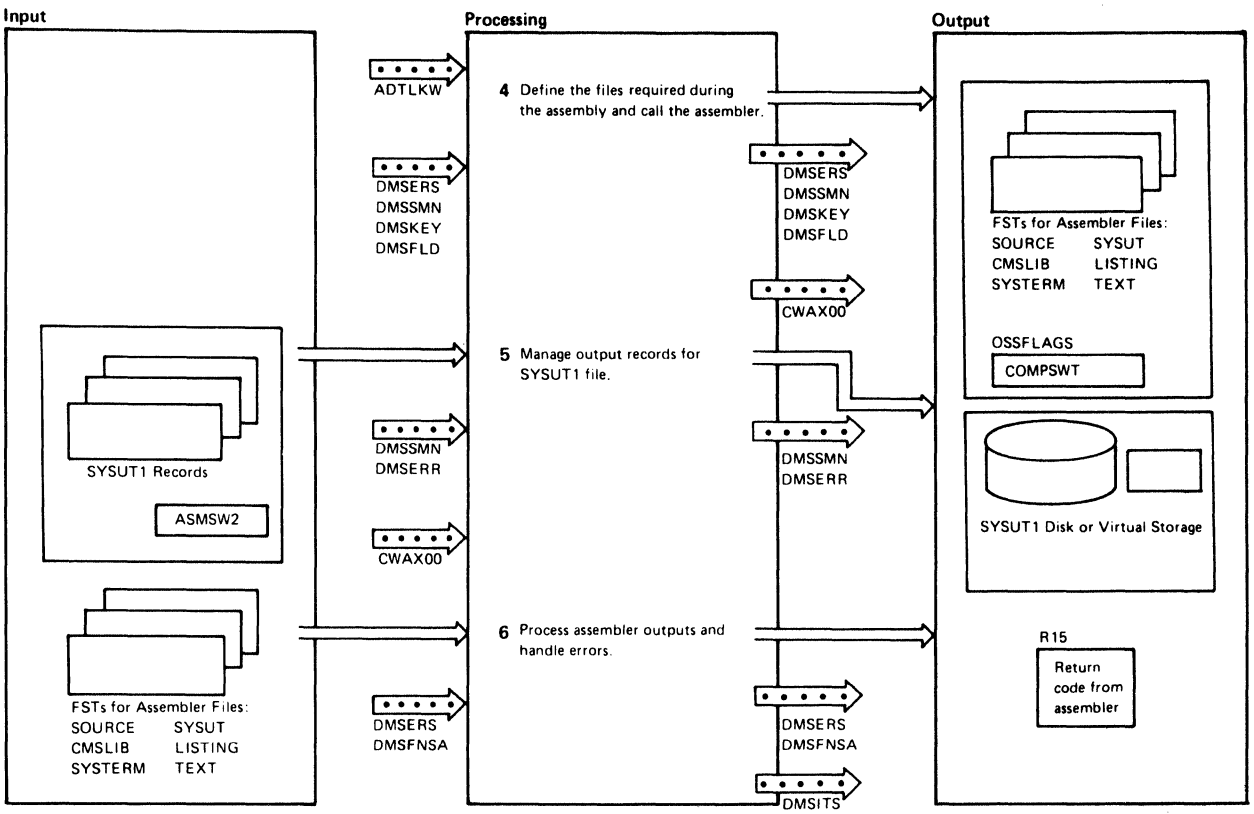
Diagram 10-6. DMSARN—ASM3705 Command Processor (for the NCP/VS Release 2 and 3 Assembler)



Notes	Module	Label	Ref
1 Validate the command line by ensuring that a filename has been specified and creating an assembler option list. If the filename is not specified, the message DMSARX001E NO FILENAME SPECIFIED is issued. The option list is built by scanning the command line, checking the options specified, and placing the valid entries in the PARAMLST table. If an invalid option is specified, the message DMSARX003E INVALID OPTION 'option' is issued and processing terminates.	DMSARX	OPTSCN	
2 Verify that the source file exists by issuing a STATE command (module DMSSTT). If the file exists but is not in proper format (80-character records), the message DMSARX007E FILE 'fn ASM3705' IS NOT FIXED, 80-CHAR. RECORDS is issued and processing terminates. If the file is in proper format, processing continues at step 3.	DMSARX	STATASM	

Notes	Module	Label	Ref
3 New files to be used during assembler processing (TEXT, LISTING, and SYSUT) can be obtained from three sources. If the input file resides on a R/W disk, that disk is used to contain the TEXT and LISTING files generated during the assembly. If the input file resides on an extension of the R/W disk, the parent disk is used. If neither of the above disks is a R/W disk, the user's A-disk is used. If no R/W disk can be obtained, the message DMSARX006E NO READ/WRITE DISK ACCESSED is issued and control returns to CMS via DMSITS.	DMSARX	FINDRW	

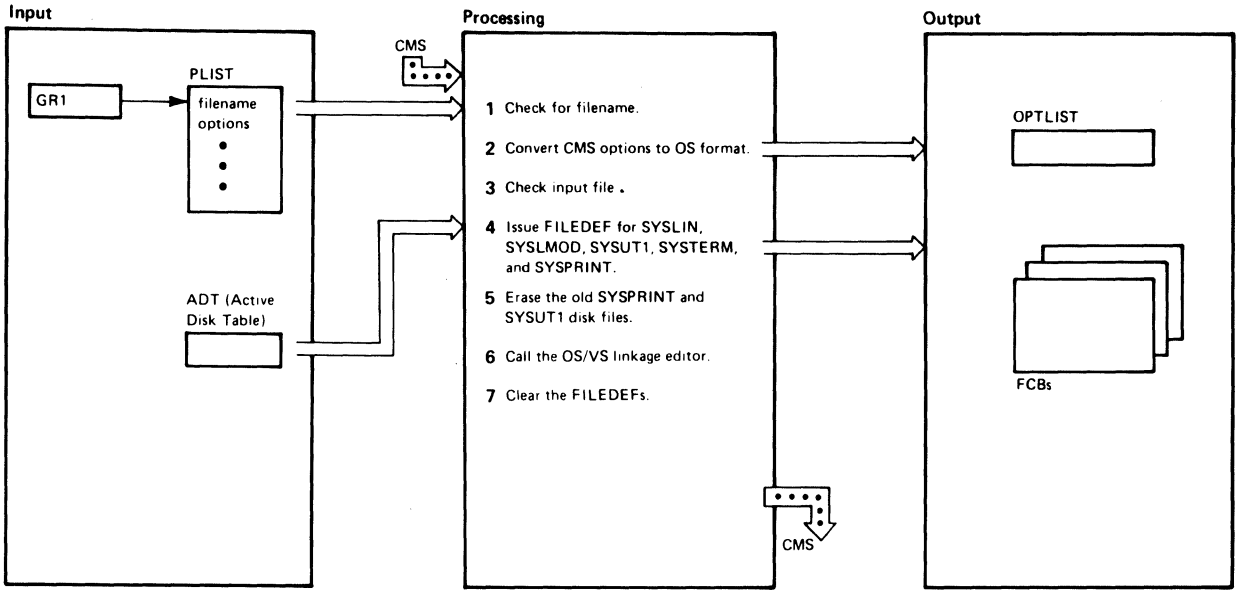
Diagram 10-7. DMSARX—ASM3705 Command Processor (for the NCP/VS Release 4 Assembler) (Part 1 of 2)



Notes	Module	Label	Ref
4 DMSERS is called to erase the old TEXT, LISTING, and SYSUT files associated with the new input file. DMSSMN (GETMAIN) is called to obtain enough storage to contain the SYSUT1 work file.	DMSARX	ERASE	
When disk space is obtained for the required assembler files and for the files CMS needs (SYSTERM and CMSLIB), FILEDEF commands are issued to convert all the files to CMS format. The assembler is then called and begins processing.		FILEDEF LOADASM	
5 If possible, all SYSUT1 records are kept in virtual storage during an assembly. However, when virtual storage is exhausted, records are written to disk.	DMSARX	ASMPROC	
If the records must be written to disk, they are formatted to fit DASD requirements and moved to disk a record at a time.		SYSWTX	
6 All SYSUT files used during the assembly are erased via a call to DMSERS. DMSFNSA is called to close all files and DMSFLD is called to clear all FILEDEFs not defined with the PERM option. COMPSWT in OSSFLAGS is turned off to indicate that the assembler is no longer processing, the auxiliary directory list is released, and control returns to CMS via DMSITS.	DMSARX	ERASUTS RETURN	

Notes	Module	Label	Ref

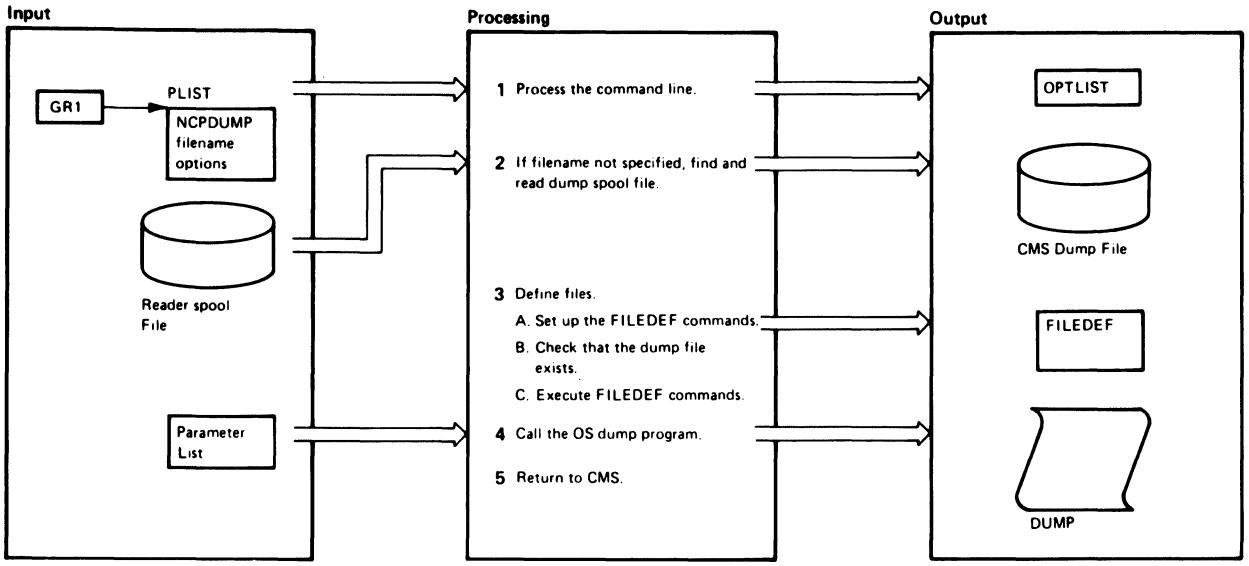
Diagram 10-7. DMSARX–ASM3705 Command Processor (for the NCP/VS Release 4 Assembler) (Part 2 of 2)



Notes	Module	Label	Ref
1 The first operand on the LKED command must be the filename. If it is not, the message DMSLKD001E NO FILENAME SPECIFIED is displayed. The filename specified is used as the default FILEDEF filename.	DMSLKD	DMSLKD	
2 If anything other than options follows the filename, the message DMSLKD070E INVALID PARAMETER 'parameter' is issued. Flags are set to reflect the following options – PRINT, NOPRINT, DISK, SIZE, NAME, TERM, NOTERM, and LIBE. If they are specified, membername and libraryname are moved into the FILEDEF commands. If NAME or LIBE is specified without a corresponding name, the message DMSLKD005E NO 'option' SPECIFIED is issued.	DMSLKD	OUTLOOP2	
3 The STATE macro is issued to check that the input file exists. If it does not, the message DMSLKD002E FILE 'fn ft' NOT FOUND is issued. If the input file does not contain fixed 80-character records, the message DMSLKD007E FILE 'fn ft' IS NOT FIXED, 80 CHAR. RECORDS is issued.	DMSLKD	OUTLOOP2	
4 The CMS file definition function is called to create a file control block for each of the linkage editor DDNAMEs: SYSLIN, SYSLMOD, SYSUT1, SYSTERM, SYSPRINT. Standard file definitions are performed unless otherwise specified on the command line.	DMSLKD	PRTDEF	

Notes	Module	Label	Ref
If no read/write disk is accessed for the SYSUT1 file, the message DMSLKD006E NO READ/WRITE DISK ACCESSED is issued.			
5 The CMS erase function is called to delete 2 disk files: 'fn SYSUT1' and 'fn LKEDIT' (fn = the input filename).	DMSLKD	PRTDEF	
6 Control is passed to the OS/VS1 linkage editor root phase (HEWLFROU) with the specified parameters and the default member name.	DMSLKD	CALL	
7 The command FILEDEF * CLEAR is issued to cancel all the file control blocks.	DMSLKD	CALL	
If the return code from the linkage editor is not zero, one of the following messages is displayed. DMSLKD004W WARNING ERROR MESSAGES ISSUED DMSLKD008W ERROR MESSAGES ISSUED DMSLKD012W SEVERE ERROR MESSAGES ISSUED DMSLKD016W TERMINAL ERROR MESSAGES ISSUED		PROCERR	
Control then returns to CMS, with the return code in register 15.		EXIT	

Diagram 10-8. DMSLKD—LKED Command Processor



Notes	Module	Label	Ref
1 If the second parameter in the input line starts with DUMP, the name of the CMS file is saved in the output FSCB. The appropriate options are marked in the OPTLIST. If there are no options specified, FORMAT, no MNEMONIC, and no ERASE are assumed. If an invalid option is specified, the following message is generated DMKRND863E INVALID PARAMETER 'xxxxxxxx' and control returns to CMS with a return code of 24.	DMKRND	NCPDUMP TESTOPT	
2 If the name of a CMS dump file was not specified, DMKRND assumes the dump file is in the reader. The filename of the output file is set to DUMP00 through DUMP09 and the STATE macro is issued until a dump file is found. If an available name is not found, the following message is generated. DMKRND851I TEN DUMP FILES ALREADY EXIST and control returns to CMS with a return code of 22.	DMKRND	LOOKLOOP	
The reader is spooled class E and the spool file is read via a DIAGNOSE instruction. The records are deblocked and written to the CMS dump file. The read/write loop continues until the real spool file DIAGNOSE instruction returns a nonzero return code. When the end of file is reached, the message 'DUMPnn NCPDUMP' FILE CREATED is issued, the spool file is closed, and processing continues. If the reader was empty or if a read error occurs, an error message is issued. DMKRND853I NO DUMP FILES EXIST DMKRND850I UNABLE TO READ DUMP FROM READER		READNXT DUMPWRT	

Notes	Module	Label	Ref
3 A. The name of the CMS dump file is put in the SYSUT2 and SYSIN FILEDEFs and in the control statement skeleton for the IPLDUMP processor.	DMKRND	STRTDUMP	
B. The STATE macro is issued to check that the CMS dump file exists. If an error is returned, the following message is generated DMKRND861E FILE 'DUMPnn NCPDUMP' NOT FOUND The SYSIN record is created, using the specified user options, any old SYSIN file is erased, and the new SYSIN file is written to the DUMPnn SYSIN file. If the record cannot be written, the message DMKRND870I UNABLE TO CREATE CONTROL FILE FOR IPLDUMP is issued and control returns to CMS.			
C. The following commands are issued to simulate an OS interface. FILEDEF SYSUT2 DISK DUMPnn NCPDUMP A1 (XTENT 513 NOCHANGE FILEDEF SYSIN DISK DUMPnn SYSIN A1 FILEDEF SYSPRINT PRINTER		LINKDMP	
4 DMKRND loads register 1 with the address of a dummy parameter list and links to IFLDUMP. If the return code from IFLDUMP is not zero, it is passed to CMS.	DMKRND		
5 If the return code from IFLDUMP is zero and ERASE has been requested, the DUMPnn file is erased, and the following message is generated 'DUMPnn NCPDUMP' FILE ERASED	DMKRND		

Diagram 10-9. DMKRND–NCPDUMP Command Processor

Program Organization

This section describes the following 3704/3705 command processing modules:

- DMKRND–NCPDUMP command processor
- DMSARN–ASM3705 command processor (for NCP/VS Release 2 and 3 Assembler)
- DMSARX–ASM3705 command processor (for NCP/VS Release 4 Assembler)
- DMSGRN–GEN3705 command processor
- DMSLKD–LKED command processor
- DMSNCP–SAVENCP command processor.

DMKRND

The interface to the OS/360 3705 dump program.

Entry Point

DMKRND

Attributes

Runs in a CMS virtual machine

Entry Conditions

R1: Address of parameter list
R13: Address of savearea
R14: Return address
R15: CSECT base register

Register Usage

R0-10: Work registers
R11: Address of FSCBDSECT
R12: CSECT base register
R13: Address of savearea
R14: Linkage register
R15: Return code

Call to Other Routines

IFLDUMP To format and print the dump

External References

None

Data Areas

FSCB

Exit Conditions

R12: CSECT base address
R13: Address of input savearea
R14: Return address
R15: Return code

DMSARN

The interface between CMS and the 3704/3705 Assembler (IFKASM).

Entry Points

DMSARN — To process the ASM3705 command.

ASMHAND — To handle any I/O activity pertaining to the SYSUT2 file during the assembly.

Attributes

Disk resident

Entry Conditions

At DMSARN

R1: Address of the parameter list
R14: Return address
R15: Address of the entry point

At ASMHAND

R1: Address of the DECB
R2: Address of the DCB
R8: Address of the OPSECT
R11: Address of the FCBSECT
R14: Return address
R15: Address of the entry point

Register Usage

R0-1: Work registers
R3: Base register
R4-5: Work registers
R6: Return address to caller
R7-9: Work registers
R10: Constant 8
R12-13: Work registers
R14: Linkage register
R15: Error code

Calls to Other Routines

DMSERSA — To erase old files
DMSSMNE — To initialize storage pointers
DMSSTTA — To locate the file
IFKASM — To assemble the 3704/3705 control program

External References

FREEMAIN — To return free storage
GETMAIN — To obtain free storage
NUCON — The nucleus constant area
TYPE — To send messages to the terminal

Data Areas

None

Exit Conditions

Contents of register 15 indicate results of processing.

Return Code	Meaning
0	No errors
4	Minor errors detected during assembly, successful program execution is probable
8	Errors detected during assembly, unsuccessful program execution is possible
12	Serious errors detected during assembly, unsuccessful execution is probable
16	Critical errors detected during assembly, unsuccessful execution is probable
20	Catastrophic errors detected during assembly, partial or complete assembly canceled
24	Invalid option, no filename
28	File not found
32	Invalid record length for ASM3705 file
36	No read/write disks accessed

DMSARX

The interface between the ASM3705 command and the 3704/3705 Assembler (CWAX00).

Entry Points

DMSARX
ASMPROC — SYSUT1 processing routine
TERMPROC — Terminal output processing routine

Attributes

Executes in user area

Entry Conditions

R1: Address of the parameter list
R14: Return address
R15: Address of the entry point (DMSARX)

Register Usage

R0 NUCON addressability
R1 Address of all PLISTs
R2 Work register
R3 Work register
R4 GETMAIN/FREEMAIN amount
R5 Work register
R6 GETMAIN/FREEMAIN address
R7 ASMPROC address
R8 Work register
R9 Work register
R10 Linkage register
R11 FCB address during ASMPROC
R12 Base register
R13 Save area address
R14 Return register from calls
R15 Assembler root address and return error code

Calls to Other Routines

DMSCRD — Read SYSPARM from console
DMSCWR — Display SYSPARM message to console
DMSFLD — FILEDEF all assembler files
DMSFNS — Close all assembler files
DMSKEY — Control nucleus protect key
DMSERR — Display all error messages
DMSERS — Erase old assembler files
DMSSLN — Load the assembler phases
DMSSMN — Control storage pointers (GETMAIN/FREEMAIN)
DMSSTT — Verify disk file existence
DMSLADAD — SET/RESET the FST chain for auxiliary directory
CWAX00 — 3705 assembler (XF) root segment

External References

ADT
CMSCB
DMSARD
FSTB
IO
NUCON

Data Areas

DDNAME — Names of CMS ddnames for assembler
OPTLIST — Option list passed to the assembler
OPDEF — (Macro label) names and abbreviations of all options
PARAMLST — Parameter list for assembler
UTENTRY — In-core SYSUT1 record area
UTHEAD — Header area for in-core records
OPTAB$ — List of pointers to option table entries
SAVEAREA — SAVEAREA

Exit Conditions

NORMAL
GPR15=0 No error

ERROR
GPR15=24 Invalid option, no filename specified
GPR15=28 File not found
GPR15=32 File not fixed, 80 char. records
GPR15=36 No read/write disks accessed
GPR15=40 Fileid conflict, device invalid for input

Return Code	Meaning
0	No errors
4	Minor errors detected during assembly, successful program execution is probable
8	Errors detected during assembly, unsuccessful program execution is possible
12	Serious errors detected during assembly, unsuccessful execution is probable
16	Critical errors detected during assembly, unsuccessful execution is probable
20	Catastrophic errors detected during assembly, partial or complete assembly canceled
24	Invalid option, no filename
28	File not found
32	Invalid record length for ASM3705 file
36	No read/write disks accessed

DMSGRN

Edits the Stage 2 input for the 3704/3705 control program generation, builds the 3704/3705 assembler files and linkage editor text files, and builds an EXEC macro file.

Entry Point

DMSGRN

Attributes

Runs in a CMS virtual machine

Entry Conditions

R1: Address of the input parameter list
R13: Address of the savearea
R14: Return address
R15: CSECT base address

Register Usage

R0-10: Work registers
R11: Base register 2
R12: Base register 1
R13: Address of the savearea
R14: Linkage register
R15: Return code

Call to Other Routines

None

External References

None

Data Areas

FSCB

Exit Conditions

R12: Base address
R13: Address of input savearea
R14: Return address
R15: Return code

DMSLKD

The interface to the OS/VS1 Linkage Editor.

Entry Point

DMKSLKD

Attributes

Reusable, disk resident

Entry Conditions

R1: Address of input parameter list

Register Usage

R0-11: Work registers
R12: Base register
R13: Address of savearea
R14-15: Work registers

Calls to Other Routines

DMSSTT — To get a copy of an FST
DMSERS — To delete a file from disk
DMSLADW — To find a read/write disk
DMSFLD — To establish file definitions for OS simulation
HEWLFROU — To link edit text files

External References

NUCON — The nucleus constant area
ADTSECT — The active disk table
FSTSECT — The file status table

Data Areas

ADT — (Active Disk Table)

Exit Conditions

Contents of register 15 indicate results of processing

Return Code	Meaning
0-16	Linkage editor return codes
20	Invalid file ID character
24	No filename specified, missing operand on LIBE or NAME option, or invalid parameter
28	File not found
32	File not fixed 80-byte records
36	No read/write disk accessed or disk not accessed

DMSNCP

Reads a 3705 control program module (EP or NCP) in OS load module format and writes a page-format core-image copy on the system volume.

Entry Point

SAVENCP

Attributes

Serially reusable, executes in a CMS virtual machine

Entry Conditions

R1: Address of the input parameter list

Register Usage

R0: Work register

R1: Address of parameter list and word register

R2: Pointer to input record and work register

R3: Length of input record and work register

R4-6: Work registers

R10: Address of the input file DCB during the read, then the address of the control program core image

R11: Address of the CCPARM parameter list

R12: Base register

R13: Address of the savearea

R14: Linkage register

R14: Linkage and work register

Calls to Other Routines

DMKSNC via Diagnose Code X'50' to write the core image of the 3704/3705 control program and parameters on disk

External References

None

Data Areas

CCPARM

Exit Conditions

R15: Return code

Directory

This section contains two types of directories:

- **Module Directory** (Figure 10-2) is a list of the CP and CMS modules that process the commands that generate the 3704/3705 control program and process the 3704/3705 storage dumps.
- **Label Directories** (Figure 10-3 through Figure 10-8) list the major labels in each of the command processors. In addition to the label, the module (if more than one is involved), associated method of operation diagram, and a brief description are included in the list.

Module	Description
DMKRND	NCPDUMP command processor.
DMSARN	ASM3705 command processor.
DMSARX	ASM3705 command processor.
DMSGRN	GEN3705 command processor.
DMSLKD	LKED command processor.
DMSNCP	SAVENCP command processor.

Figure 10-2. Module Directory for 3704/3705 Command Processors

The NCPDUMP Command Processor (DMKRND)

Label	Diagram	Description
DUMPWRT	10-9	Writes the output file.
LINKDMP	10-9	Links to the OS dump service program, IFLDUMP.
LOOKLOOP	10-9	Checks the reader for a valid CMS dump file.
NCPDUMP	10-9	Starts processing the NCPDUMP command.
READNXT	10-9	Reads the dump spool file.
STRTDUMP	10-9	Builds the control file for the IFLDUMP processing routine.
TESTOPT	10-9	Processes the options on the NCPDUMP command line.

Figure 10-3. The NCPDUMP Command Processor (DMKRND) Label Directory

The ASM3705 Command Processor (DMSARN)

Label	Diagram	Description
CONTINUE	10-6	Erases old files and gets enough storage for the assembler to execute in.
DMSARN	10-6	Entry point for the ASM3705 command processor.
LIST2	10-6	Calls the 3705 Assembler (IFKASM).
NOERASE	10-6	Issues FILEDEFs for the necessary assembler files.
RETURN	10-6	Returns control to CMS.
SQUEEZE	10-6	Checks that the input file exists.
SUIT15	10-6	If running in a batch machine, sends ASSEMBLING filename A1 message.
SUIT17	10-6	Finds a read/write disk for writing text and listing files.
SUIT19	10-6	Closes the output files and erases the utility files.
SUIT25	10-6	Checks the format of the input file.

Figure 10-4. The ASM3705 Command Processor (DMSARN) Label Directory

The ASM3705 Command Processor (DMSARX)

Label	Diagram	Description
ERASE	10-7	Erases old files.
DMSARX	10-7	Entry point for the ASM3705 command processor.
FILEDEF	10-7	Issues FILEDEFs for the necessary assembler files.
FINDRW	10-7	Finds a read/write disk for writing text and listing files.
LOADASM	10-7	Load the 370X Assembler root.
OPTSCN	10-7	Validates command line.
RETURN	10-7	Returns control to CMS.
VERIFY	10-7	Checks that the input file exists.

Figure 10-5. The ASM3705 Command Processor (DMSARX) Label Directory

The GEN3705 Command Processor (DMSGRN)

Label	Diagram	Description
ASMSTAK	10-4	Stacks the required 3705 Assembler commands in the Stage 2 EXEC macro file.
ASMSTAK2	10-4	Puts the name of the output assembler file in the ASM3705 and EDIT commands.
ASMSTAK4	10-4	Puts the SYSPUNCH membername in the TXTLIB command.
ASMSTAK6	10-4	Puts the number of commands and the address of the first command into registers 1 and 2.
CLOSTACK		Builds the SAVENCP command.
EDITIN		Edits the input records for keywords.
FINDASM	10-3	Checks for assembler input.
FINDIEWL	10-3	Checks for linkage editor input.
FSCBCLOS	10-4	Closes the output file.
FSCBRD		Reads the input file.
FSCBWT		Writes the output file.
GENMSG		Generates error messages.
IEWL	10-5	Main processing routine for generating linkage editor commands.
IEWLENT	10-5	Scans for the keyword ENTRY.
IEWLJCLA	10-5	Edits the //EXEC statement.
IEWLJCL2	10-5	Scans for the //SYSLIN statement.
IEWLSEOF	10-5	Branches and links to FSCBCLOS to close the linkage editor output file.
IEWLSIN		Processes SYSLIN information.
IEWLSN10	10-5	Branches and links to FSCBRD to read the linkage editor input file.
IFKASM	10-4	Main processing routine for generating 3705 assembler files.
IFKASMA0	10-4	Branches and links to the FSCBRD and FSCBWT routines to read the input file and write the output file.
IFKASMK0	10-4	Branches and links to the FSCBCLOS routine to close the output assembler files.
IFKASM10	10-4	Scans for the SYSPUNCH statement.

Figure 10-6 (Part 1 of 2). The GEN3705 Command Processor (DMSGRN) Label Directory

Label	Diagram	Description
IFKASM34	10-4	Scans for the DSN= or DSNAME= keyword on the SYSPUNCH statement.
IFKASM40	10-4	Scans for the SYSIN statement.
LKDSTACK	10-4	Builds the LKED commands and the FILEDEF for their file.
LKDSTAK1	10-5	Loads registers 1 and 2 with the number of commands and the address of the first linkage editor command.
OPTEND	10-3	Checks that the input file exists.
OPTIONS1	10-3	Scans the input options.
PRIMEDIT	10-3	Scans for a valid //EXEC statement.
PROCEND1	10-3	Closes the EXEC file.
PROCEND2	10-3	Adds the SAVENCP command to the EXEC macro file.
PROCWT		Writes commands to the Stage 2 EXEC processor file.
RETURN1	10-3	Returns control to CMS.
STACK30	10-3	Writes the linkage editor and assembler statements to the EXEC macro file.
START	10-3	Starts the GEN3705 command processing.
WRTSIN	10-4	Branches and links to the FSCBWT routine to write the linkage editor output file.

Figure 10-6 (Part 2 of 2). The GEN3705 Command Processor (DMSGRN) Label Directory

The LKED Command Processor (DMSLKD)

Label	Diagram	Description
CALL	10-8	Calls the OS/VS1 Linkage Editor (HEWLFROU).
DMSLKD	10-8	Entry point for the LKED command processor.
EXIT	10-8	Returns control to CMS.
OUTLOOP2	10-8	Processes the command options.
PROCERR	10-8	Processes the error messages.
PRTDEF	10-8	Sets up the file definition for the printer.

Figure 10-7. The LKED Command Processor (DMSLKD) Label Directory

The SAVENCP Command Processor (DMSNCP)

Label	Diagram	Description
CESDCHVT	10-1	Finds the channel vector table.
CESDENT	10-1	Saves the entry point.
CHEKVMV	10-2	Checks that the specified options are compatible.
CLOSE	10-1	Closes the input file.
CONTROL	10-1	Moves the text records from the input buffer to the core image buffer.
ENDPARMS	10-1	Opens the input file and searches for the member.
ERR21	10-1	Checks for the entry point record.
ERR66	10-1	Checks for premature end of file or invalid control records.
EXIT	10-1	Returns control to CMS.
SAVECCP	10-1	Issues the Diagnose X'50' instruction to have DMKSNC do the actual saving.
SAVENCP	10-1	Entry point for the SAVENCP command processor.
SCANCEP	10-2	Updates the CCPARM parameter list for EP and PEP control programs.
SCANDEV		Scans for devices.
SCANLINE		Scans for teleprocessing lines.
SCANNCP	10-2	Updates the CCPARM parameter list for NCP and PEP control programs.

Figure 10-8. The SAVENCP Command Processor (DMSNCP) Label Directory

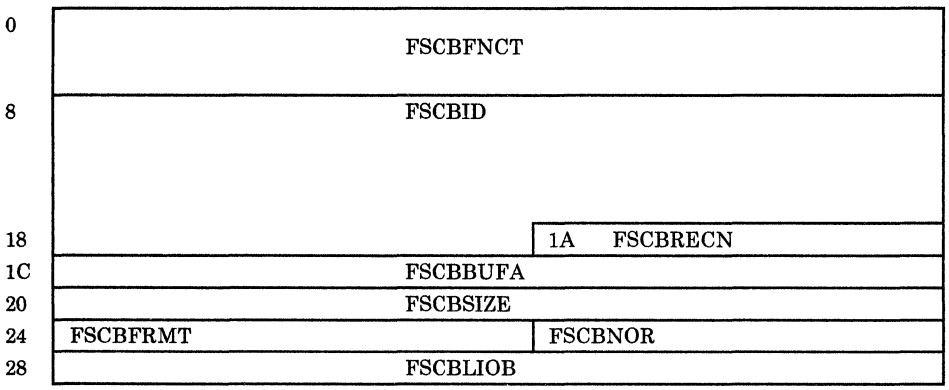
Data Areas

The following data areas are used by the 3704/3705 command processor modules:

- Active Disk Table (ADT)
- Communications Controllers Parameter List (CCPARM)
- File System Control Block (FSCB)
- Input/Output Block (IOBLOK)
- Network Interface Control Block (NICBLOK)
- Real Device Block (RDEVBLOK)
- Spool File Block (SFBLOK)
- Virtual Machine Block (VMBLOK).

All the above data areas except the FSCB are described in the *VM/SP HPO Data Areas and Control Block Logic*. The FSCB is described in Figure 10-9.

File System Control Block



Displacement Hex	Dec	Field Name			Description
0	0	FSCBFNCT	DS	CL8	Control field for I/O function
8	8	FSCBID	DS	OCL18	File Identifier
8	8	FSCBFN	DS	CL8	Filename
10	16	FSCBFT	DS	CL8	Filetype
18	24	FSCBFM	DS	CL2	Filemode
1A	26	FSCBRECN	DS	H	Relative record number
1C	28	FSCBBUFA	DS	A	Buffer address
20	32	FSCBSIZE	DS	F	Buffer size
24	36	FSCBFRMT	DS	CL2	File format
26	38	FSCBNOR	DS	0H	Number of records to be read
28	40	FSCBLIOB	DS	A	

Figure 10-9. File System Control Block (FSCB)

Diagnostic Aids

The following figures list the messages and abnormal termination codes issued by the CMS 3704/3705 command processors.

Figure 10-10 lists the messages issued by the NCPDUMP command processor (DMKRND).

Figure 10-11 and Figure 10-12 list the messages issued by the ASM3705 command processor (DMSARN and DMSARX).

Figure 10-13 lists the messages issued by the GEN3705 command processor (DMSGRN).

Figure 10-14 lists the messages issued by the LKED command processor (DMSLKD).

Figure 10-15 lists the messages issued by the SAVENCP command processor (DMSNCP).

The NCPDUMP Command Processor (DMKRND)

Message Code	Label	Diagram	Message Text
DMKRND850I	DUMPWRT	10-9	UNABLE TO READ DUMP FROM READER (Return Code = 21)
DMKRND851I	LOOKLOOP	10-9	TEN DUMP FILES ALREADY EXIST (Return Code = 22)
DMKRND852I			FATAL I/O ERROR WRITING DUMP
DMKRND853I	DUMPWRT	10-9	NO DUMP FILES EXIST (Return Code = 23)
DMKRND861E	STRTDUMP	10-9	DUMP FILE filename NOT FOUND (Return Code = 28)
DMKRND863E	TESTOPT	10-9	INVALID PARAMETER - parameter (Return Code = 24)
DMKRND870I	STRTDUMP	10-9	UNABLE TO CREATE CONTROL FILE FOR IPLDUMP (Return Code = 16)
	DUMPWRT	10-9	'DUMPnn NCPDUMP' FILE CREATED
	LINKDMP	10-9	'DUMPnn NCPDUMP' FILE ERASED

Figure 10-10. The NCPDUMP Command Processor (DMKRND) Error Messages

The ASM3705 Command Processor (DMSARN)

Message Code	Label	Diagram	Message Text
DMSARN001E	DMSARN	10-6	NO FILENAME SPECIFIED
DMSARN002E			[INPUT/OVERLAY] {FILE[(s)]/DATA SET} ['fn[fm]]' NOT FOUND
DMSARN003E			INVALID OPTION 'option'
DMSARN004W	RETURN	10-6	WARNING MESSAGES ISSUED
DMSARN006E			NO READ/WRITE ['A'] DISK ACCESSED [FOR 'fn ft']
DMSARN007E	SUIT25	10-6	FILE 'fn ft fm' [IS] NOT FIXED, 80-CHAR. RECORDS
DMSARN008W	RETURN	10-6	ERROR MESSAGES ISSUED
DMSARN012W	RETURN	10-6	SEVERE ERROR MESSAGES ISSUED
DMSARN016W	RETURN	10-6	TERMINAL ERROR MESSAGES ISSUED

Figure 10-11. The ASM3705 Command Processor (DMSARN) Error Messages

The ASM3705 Command Processor (DMSARX)

Message Code	Label	Diagram	Message Text
DMSARX001E	OPTSCN	10-7	NO FILENAME SPECIFIED
DMSARX002E	NEWFILE	10-7	[INPUT/OVERLAY] {FILE[(s)]/DATA SET} ['fn[fm]]' NOT FOUND
DMSARX003E	OPTSCN	10-7	INVALID OPTION 'option'
DMSARX007E	FINDRW	10-6	NO READ/WRITE ['A'] DISK ACCESSED [FOR 'fn ft']
DMSARX007E	STATASM	10-7	FILE 'fn ft fm' [IS] NOT FIXED, 80-CHAR. RECORDS
DMSARX038E	DOFDEF	10-7	FILEID CONFLICT FOR DDNAME 'ASM3705'
DMSARX052E	MOVEKEY	10-7	MORE THAN 100 CHARS OF OPTIONS SPECIFIED
DMSARX070E	DMSARX	10-7	INVALID {PARAMETER 'parameter'/ARGUMENT 'argument'}
DMSARX074E	DMSARX	10-7	ERROR [RE]SETTING AUXILIARY DIRECTORY
DMSARX075E	NOTDSK	10-7	DEVICE 'device' INVALID FOR {INPUT/OUTPUT}

Figure 10-12. The ASM3705 Command Processor (DMSARX) Error Messages

The GEN3705 Command Processor (DMSGRN)

Message Code	Label	Diagram	Message Text
DMSGRN002E	OPTEND	10-3	[INPUT/OVERLAY] {FILE[(S)]/DATA SET} ['fn[fm]]' NOT FOUND
DMSGRN003E	OPTIONS1	10-3	INVALID OPTION 'option'
DMSGRN007E			FILE 'fn ft fm' [IS] NOT FIXED, 80 CHAR. RECORDS
DMSGRN048E	OPTEND	10-3	INVALID MODE 'mode'
DMSGRN054E			INCOMPLETE FILEID SPECIFIED
DMSGRN078E	PRIMEDIT	10-3	INVALID CARD IN READER {DECK/FILE'cardimage'}
	IFKMAS40	10-4	'xxx... x'
	IEWLJCL2	10-5	

Figure 10-13. The GEN3705 Command Processor (DMSGRN) Error Messages

The LKED Command Processor (DMSLKD)

Message Code	Label	Diagram	Message Text
DMSLKD001E	DMSLKD	10-8	NO FILENAME SPECIFIED
DMSLKD002E	OUTLOOP2	10-8	[INPUT/OVERLAY] {FILE[(s)]/DATA SET} ['fn[fm]]' NOT FOUND
DMSLKD004W	PROCERR	10-8	WARNING MESSAGES ISSUED
DMSLKD005E	OUTLOOP2	10-8	NO 'option' SPECIFIED
DMSLKD006E	PRTDEF	10-8	NO READ/WRITE ['A'] DISK ACCESSED [FOR 'fn ft']
DMSLKD008E	OUTLOOP2	10-8	FILE 'fn ft fm' [IS] NOT FIXED, 80 CHAR. RECORDS
DMSLKD008W	PROCERR	10-8	ERROR MESSAGES ISSUED
DMSLKD012W	PROCERR	10-8	SEVERE ERROR MESSAGES ISSUED
DMSLKD016W	PROCERR	10-8	TERMINAL ERROR MESSAGES ISSUED
DMSLKD080E			INVALID PARAMETER 'parameter'/ARGUMENT 'argument'

Figure 10-14. The LKED Command Processor (DMSLKD) Error Messages

The SAVENCP Command Processor (DMSNCP)

Message Code	Label	Diagram	Message Text
DMSNCP001E	SAVENCP	10-1	NO FILENAME SPECIFIED (Return Code = 24)
DMSNCP002E	ENDPARMS	10-1	[INPUT/OVERLAY] {FILE[(s)]/DATA SET} ['fn[fm]]' NOT FOUND (Return Code = 28)
DMSNCP003E	SAVENCP TESTOP		INVALID OPTION 'option' (Return Code = 24)
DMSNCP013E	DMS0001A	10-1	MEMBER 'name' NOT FOUND IN LIBRARY ['fn ft fm'/'libname'] (Return Code = 4)
DMSNCP021E	CONTROL	10-1	ENTRY POINT 'name' NOT FOUND (Return Code = 40)
DMSNCP025E	SCANCEP SCANNCP	10-2	INVALID DATA IN 370X CONTROL PROGRAM (Return Code = 16)
DMSNCP045E	CLOSE		UNSUPPORTED 370X CONTROL PROGRAM TYPE (Return Code = 16)
DMSNCP056E	NOTLAST	10-1	FILE 'fn ft fm' CONTAINS INVALID {NAME/ALIAS/ENTRY/ESD} RECORD FORMATS
	CLOSE ERR66		(Return Code = 32)
DMSNCP099W	CHEKVMV	10-2	GENERATION PARAMETERS INCOMPATIBLE WITH VM/SP (Return Code = 99)
DMSNCP109S	CONTROL NOTLAST	10-1	VIRTUAL STORAGE CAPACITY EXCEEDED (Return Code = 104)

Figure 10-15. The SAVENCP Command Processor (DMSNCP) Error Messages

Chapter 11. The ZAP Service Program

Introduction

The ZAP service program (DMKZAP) executes under the control of CMS via the ZAP and ZAPTEXT commands. The ZAP command performs functions for LOADLIB, TXTLIB, and MODULE files residing on direct access storage devices (DASDs). The ZAPTEXT command performs functions for individual text files and internally invokes ZAP. For a complete description of the ZAP and ZAPTEXT commands, see the *VM/SP HPO Operator's Guide.*

The functions that ZAP and ZAPTEXT can perform are:

- Dump
- Verify
- Replace

In addition, ZAPTEXT can also perform the EXPAND function.

DUMP

The dump function reads all or part of a specified CSECT, or an entire member or module, formats the dump, and prints it at the system printer (133-character lines, each containing 32 bytes in hexadecimal, plus the translation) or displays it at the terminal (80-character lines, each containing 16 bytes in hexadecimal, plus the translation). If more than one CSECT is dumped, the CSECT name appears before each dump.

VERIFY

The verify function compares specified data with the data at a specified address in a CSECT. If the data is the same, a replace operation (if one is specified) is permitted; otherwise, an error message is issued.

REPLACE

The replace function replaces data at a specified address in a CSECT with the data specified in a control record. The changed record is then written back to the file.

EXPAND

The ZAPTEXT service program uses the EXPAND command to add space to a program in object deck form. The ZAP service program, however, ignores the EXPAND command.

Method of Operation

The method of operation diagrams describe the execution of the ZAP program and show the processing associated with:

- Verifying and replacing data in a CSECT
- Dumping a CSECT, member, or module.

The relationship of the method of operation diagrams is shown in Figure 11-1.

Diagram 11-1 describes the execution of the ZAP program.

Diagram 11-2 shows the ZAP command and control record processing.

Diagram 11-3 describes the processing of the DUMP function.

Diagrams 11-4 and 11-5 describe the processing for modifying data in a CSECT.

Diagrams 11-6 and 11-7 describe how the proper CSECT is located for dumping or modifying.

Diagram 11-8 shows how a file is read for dumping or modifying.

Diagram 11-9 describes how a dump is printed.

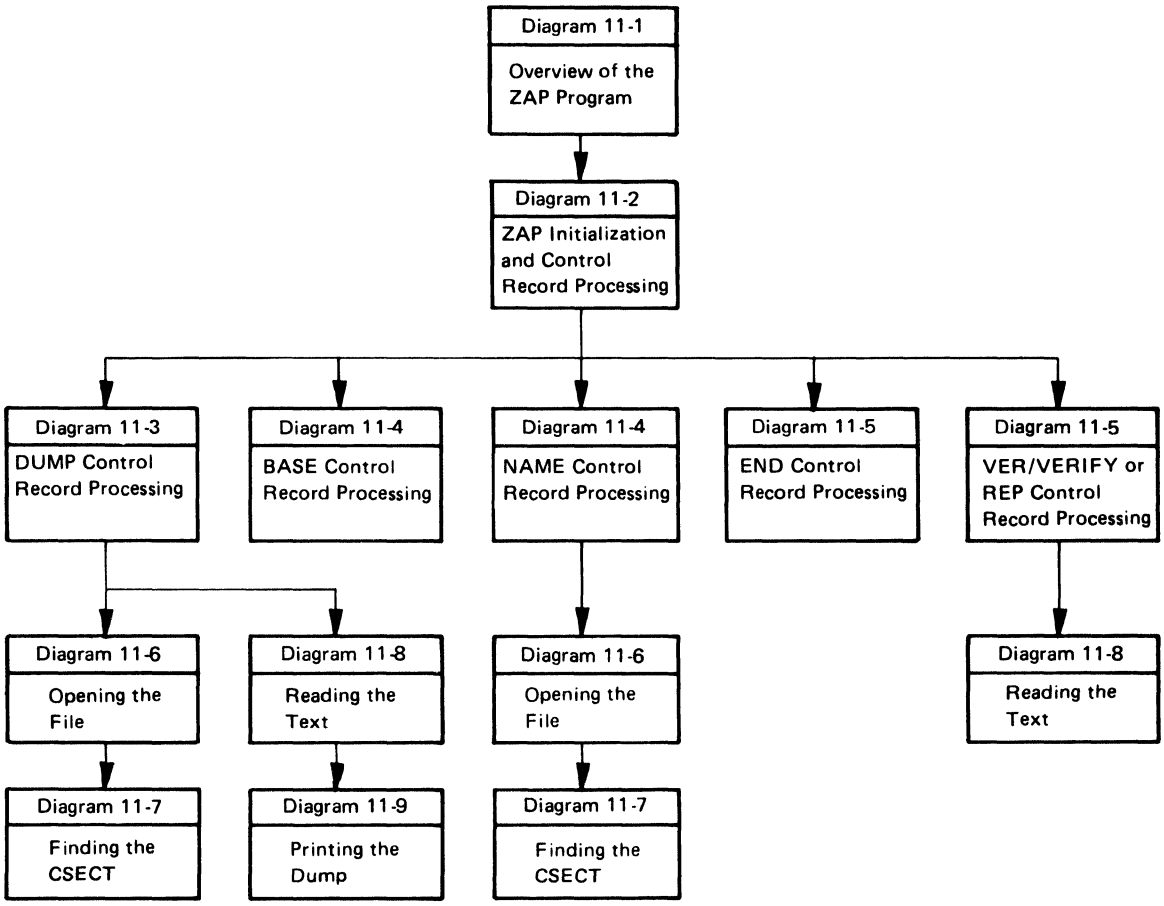
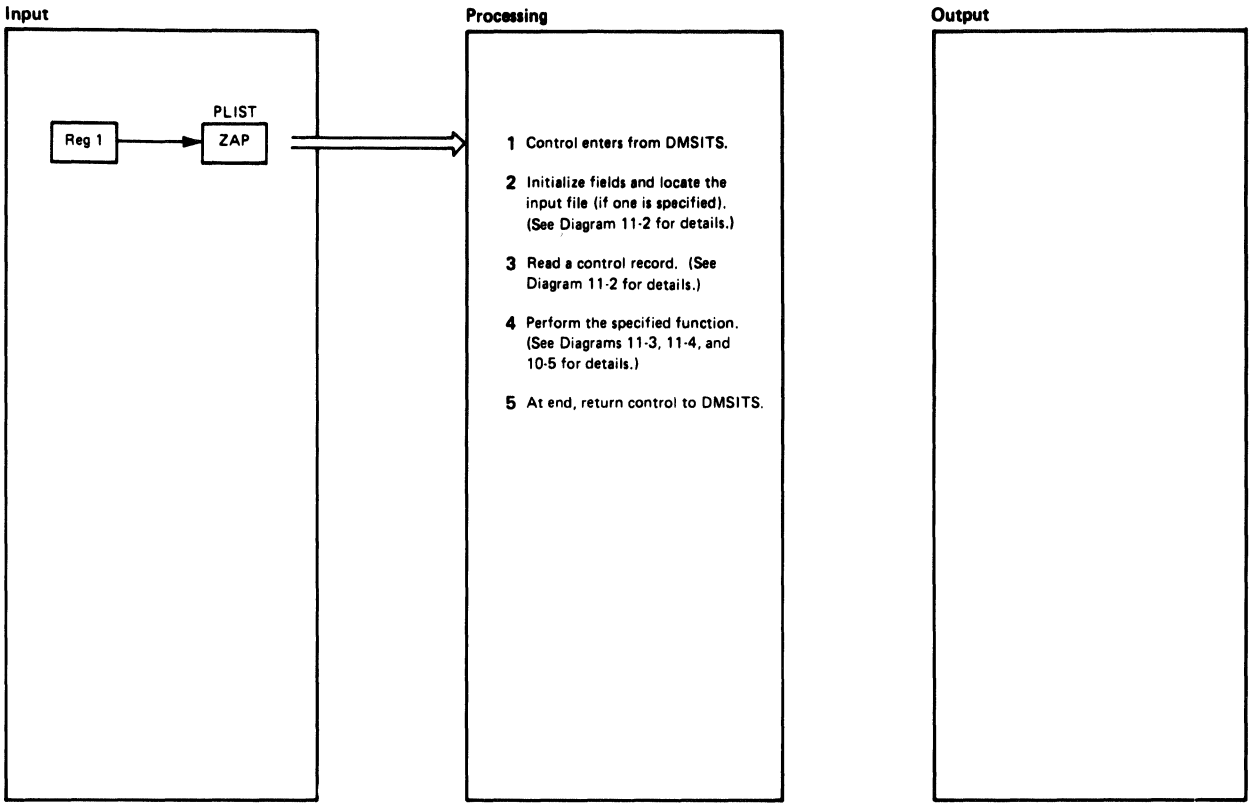


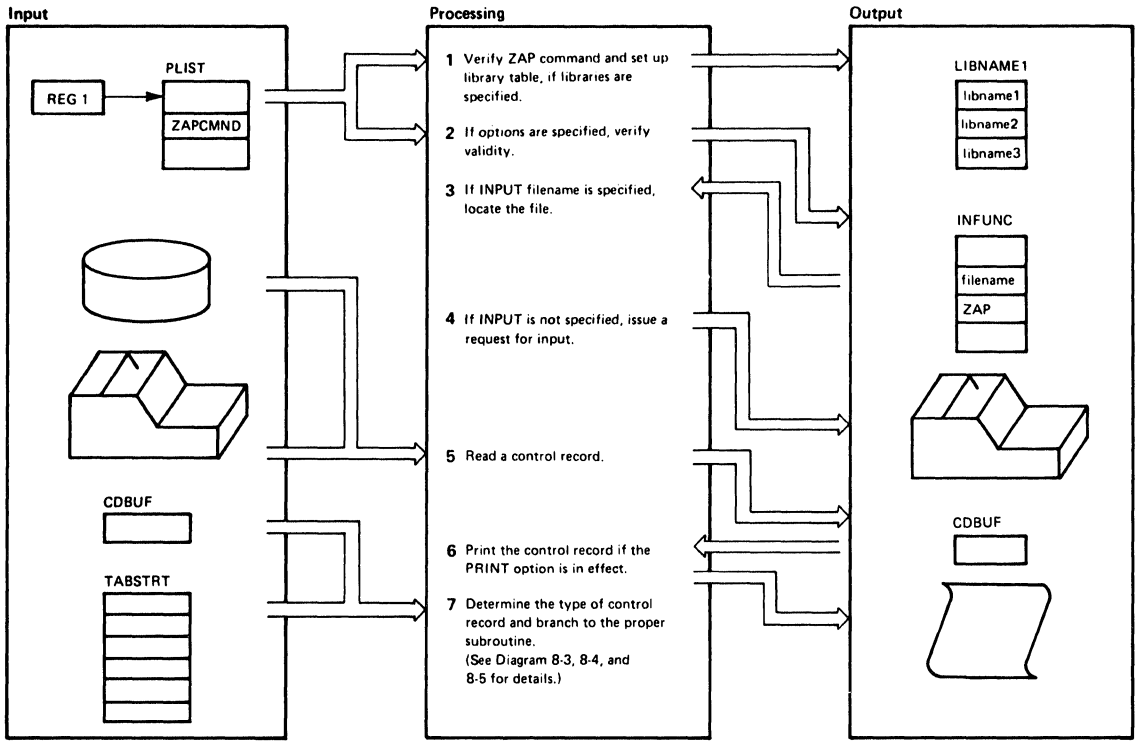
Figure 11-1. Key to the ZAP Program Method of Operation Diagrams



Notes	Module	Label	Ref
1 Control enters DMSZAP from DMSITS. Register 1 points to a PLIST that contains the type of file to be operated on, libraries to be used if applicable, and controls for input and output operations.	DMSZAP	DMSZAP	
2 Initialize fields and pointers and verify input and output options. Locate the input file if an input file is specified. Otherwise, request input from the terminal.	DMSZAP	SCANLINE INITOPEN FDEFINP	
3 Read a control record. Find the routine needed to perform the function specified by searching a table of control record keywords.	DMSZAP	READINP	
4 Perform the specified function. At its end, return control to READINP to read another control record.	DMSZAP		
5 When the END control record is read, return control to DMSITS.	DMSZAP		

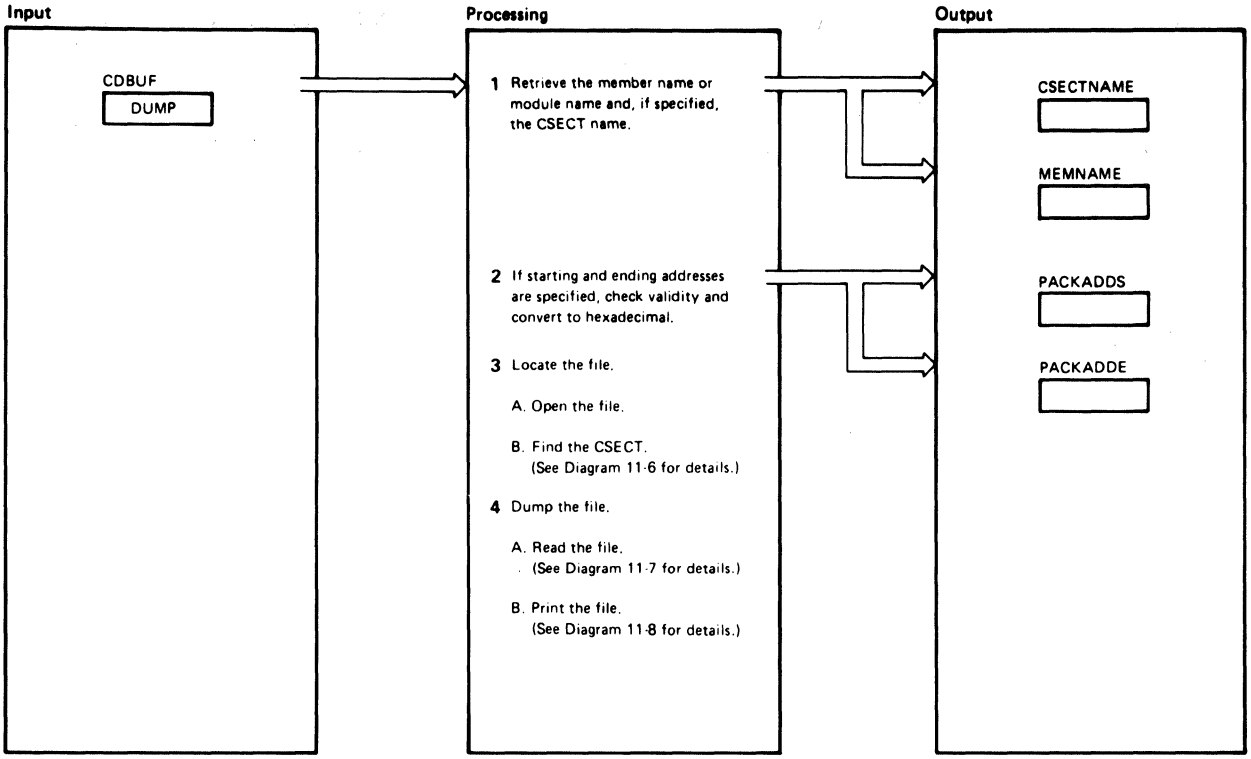
Notes	Module	Label	Ref

Diagram 11-1. Overview of the ZAP Program



Notes	Module	Label	Ref
1 Verify the operands in the ZAP command. If TXTLIB or LOADLIB is specified, move the library names (up to three) into LIBNAME1. If no library name was specified, issue the message: DMSZAP001E NO FILENAME SPECIFIED Other messages that may be issued if the command line is in error are: DMSZAP014E INVALID FUNCTION 'function' DMSZAP047E NO FUNCTION SPECIFIED DMSZAP070E INVALID PARAMETER 'param'	DMSZAP	SCANLINE STLIB	
2 If options are specified, check for validity. If mutually exclusive options or invalid options are specified, issue the message: DMSZAP003E INVALID OPTION 'option'	DMSZAP	CHKOPT	
3 If INPUT filename is specified, move filename into INFUNC. Issue STATE to locate the file. If this file cannot be found, the message: DMSZAP002E FILE 'fn ft' NOT FOUND	DMSZAP	INPTOPT FDEFINP	
4 If INPUT is not specified, display ENTER: to request ZAP control records to be entered from the terminal.	DMSZAP	READINP RDCARD	
5 Read the control record either from the terminal (RDCARD routine) or from the specified INPUT file (RDCARD2 routine). Save the control record in CDBUF.		RDCARD2	
6 Print the control record on the SYSOUT printer if the PRINT option is in effect.	DMSZAP	WRCARD	
7 Check the control record for a valid keyword. If the statement is blank, or the first word is EXPAND, or the first character is an asterisk, return control to READINP (step 4). Otherwise, compare the keyword to keyword tables whose formats are: bytes 1-8 keyword bytes 9-12 keyword routine Valid keywords and the diagrams in which their routines are described are: Keyword / Diagram DUMP 8-3 NAME 8-4 BASE 8-4 VER 8-5 VERIFY 8-5 REP 8-5 END 8-5 If a match is found, go to the appropriate routine. If no match is found, issue the message: DMSZAP201W INVALID CONTROL RECORD OR NO GO SWITCH SET and return control to READINP (step 4).	DMSZAP	SCANKEY1 TABLOOK NAMFOUND INVEREP	

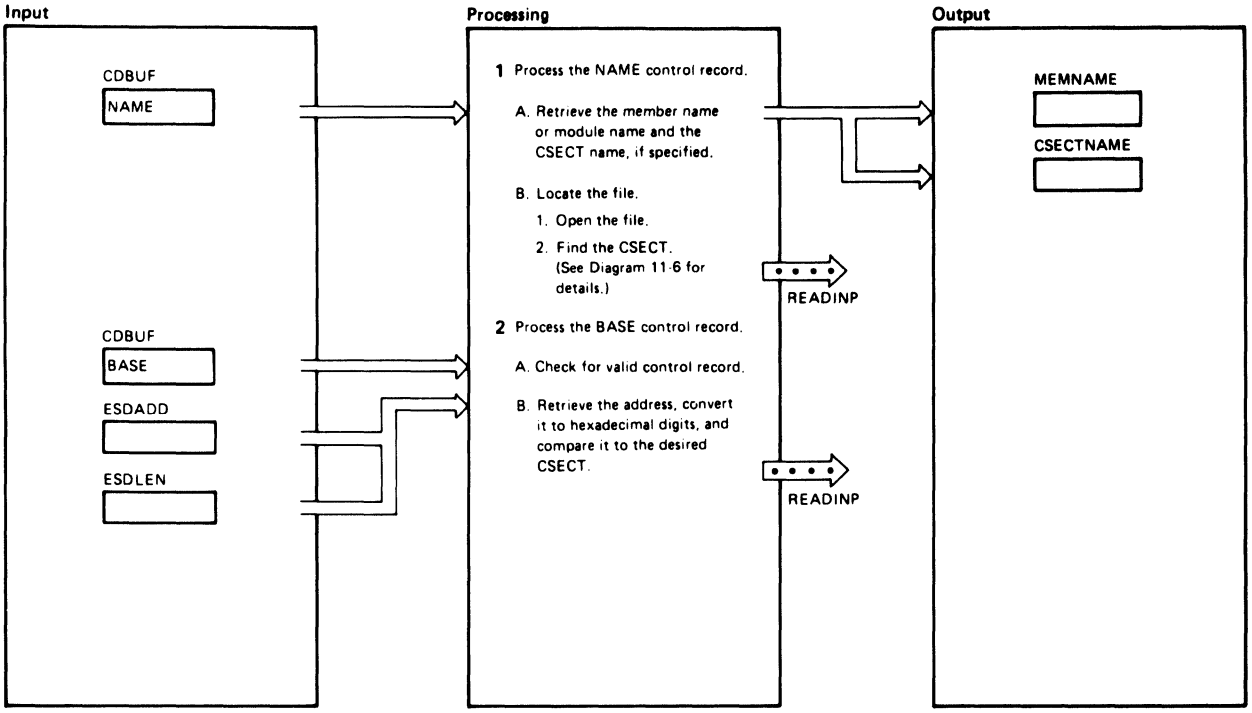
Diagram 11-2. ZAP Initialization and Control Record Processing



Notes	Module	Label	Ref
1 Retrieve the member name or module name, if specified, from the control record. If an error is encountered, issue the message DMSZAP201W INVALID CONTROL RECORD OR NO GO SWITCH SET Continue by reading another control record.	DMSZAP	DUMPREC DUMPERR	
2 If starting and ending addresses are specified, retrieve them from the control record, check them for validity, and convert them into hexadecimal digits. If either of the addresses is not an even number of digits, issue the message DMSZAP203W – ERROR – ODD NUMBER OF DIGITS – SET NO GO SWITCH and continue by reading another control record.	DMSZAP	DMPNTALL SCANKEY1 DECODE1 PACKVAL INVEREP2	
3 Go to the open routine (PREOPLIB) to locate the member or module and the CSECT desired.	DMSZAP	DMPCSECT PREOPLIB	
4 Use the starting and ending addresses of the CSECT to determine the length of the dump if not otherwise specified. Go to the read text routine to read the file (RDTXT) and then to the print dump routine (PRTDUMP).	DMSZAP	STSTART GORDTXT RDTXT PRTDUMP	

Notes	Module	Label	Ref
If all CSECTs are requested, return control to step 3. When the request is satisfied, read another control record (see Diagram 11-2, Step 4).		READINP	

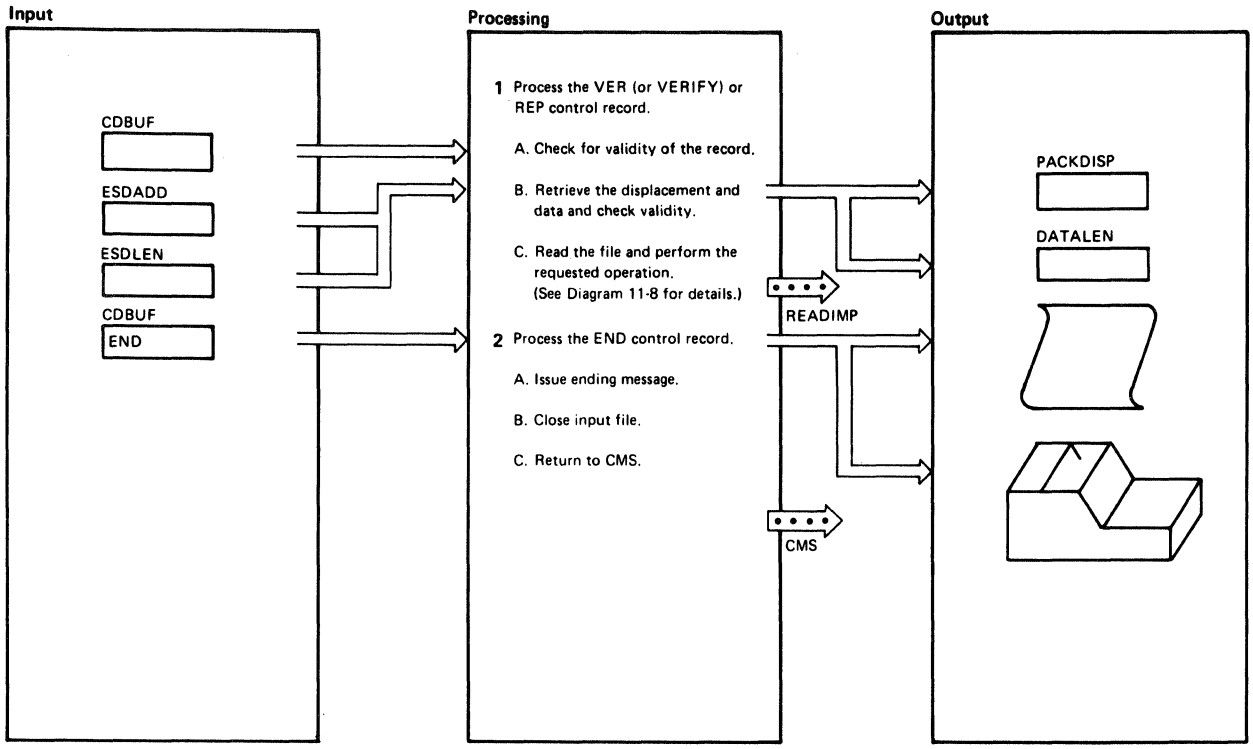
Diagram 11-3. DUMP Control Record Processing



Notes	Module	Label	Ref
1			
A. Retrieve the member name or module name and the CSECT name, if specified, and check for errors. If errors are found, issue the message DMSZAP190W INVALID CONTROL RECORD OR NO GO SWITCH SET Continue by reading another control record.	DMSZAP	NAMEREC INVEREP	
B. If no errors are found, open the specified file and locate the desired CSECT. Continue by reading another control record.	DMSZAP	NOCSECT1 PREOPLIB READINP	
2			
A. Check that the NAME control record has been entered. If not, issue the message DMSZAP190W INVALID CONTROL RECORD OR NO GO SWITCH SET Continue by reading another control record.	DMSZAP	BASEREC INVEREP	
B. Retrieve the BASE address, check it for accuracy, and convert it to hexadecimal.	DMSZAP	CKBASE DECODE1 PACKVAL	
If the address is not an even number of digits, issue the message DMSZAP192W ERROR – ODD NUMBER OF DIGITS – SET NO GO SWITCH and continue by reading another control record.		INVEREP2	

Notes	Module	Label	Ref
If the file is a MODULE file created with the NOMAP option, accept the BASE address and continue by reading another control record.			
If the file is a LOADLIB or TXTLIB file, or a MODULE file not created with the NOMAP option, compare the BASE address to the CSECT address. If there is a match, continue by reading another control record.		CKBASE1	
If the CSECT address is not equal to the BASE address, issue the message DMSZAP195W BASE VALUE INVALID – SET NO GO SWITCH Continue by reading another control record.		INVEREP2	

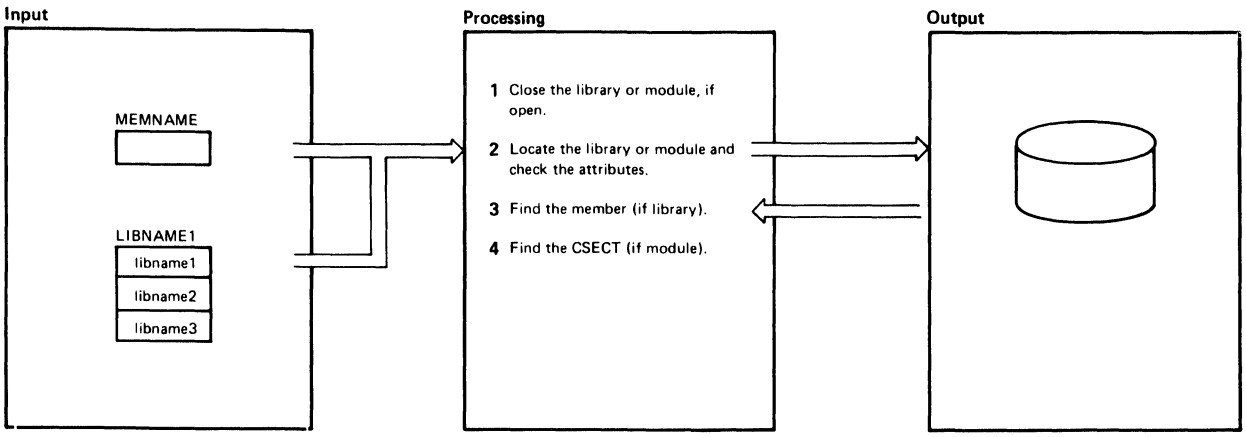
Diagram 11-4. NAME and BASE Control Record Processing



Notes	Module	Label	Ref
1			
A. If a NAME control record has not been entered or was invalid, issue the message DMSZAP190W INVALID CONTROL RECORD OR NO GO SWITCH SET and return control to READINP to read another control record. Ignore all VER or REP control records until the next NAME control record is encountered.	DMSZAP	GOODTHRE INVEREP	
If this is a REP control record and the NO GO switch is on, issue the message DMSZAP193W PRECEDING CONTROL RECORD FLUSHED and return control to READINP to read another control record.		INVEREP2	
B. Check the displacement for validity and convert into hexadecimal digits.	DMSZAP	GOOK SCANKEY1 DECODE1 PACKVAL	
Retrieve the data field, remove commas from the field, and check that the data are an even number of bytes. If not, issue the message DMSZAP192W ERROR – ODD NUMBER OF DIGITS – SET NO GO SWITCH	DMSZAP	SCANKEY1 CKCOMMA2 CKCOMMA3 INVEREP2	

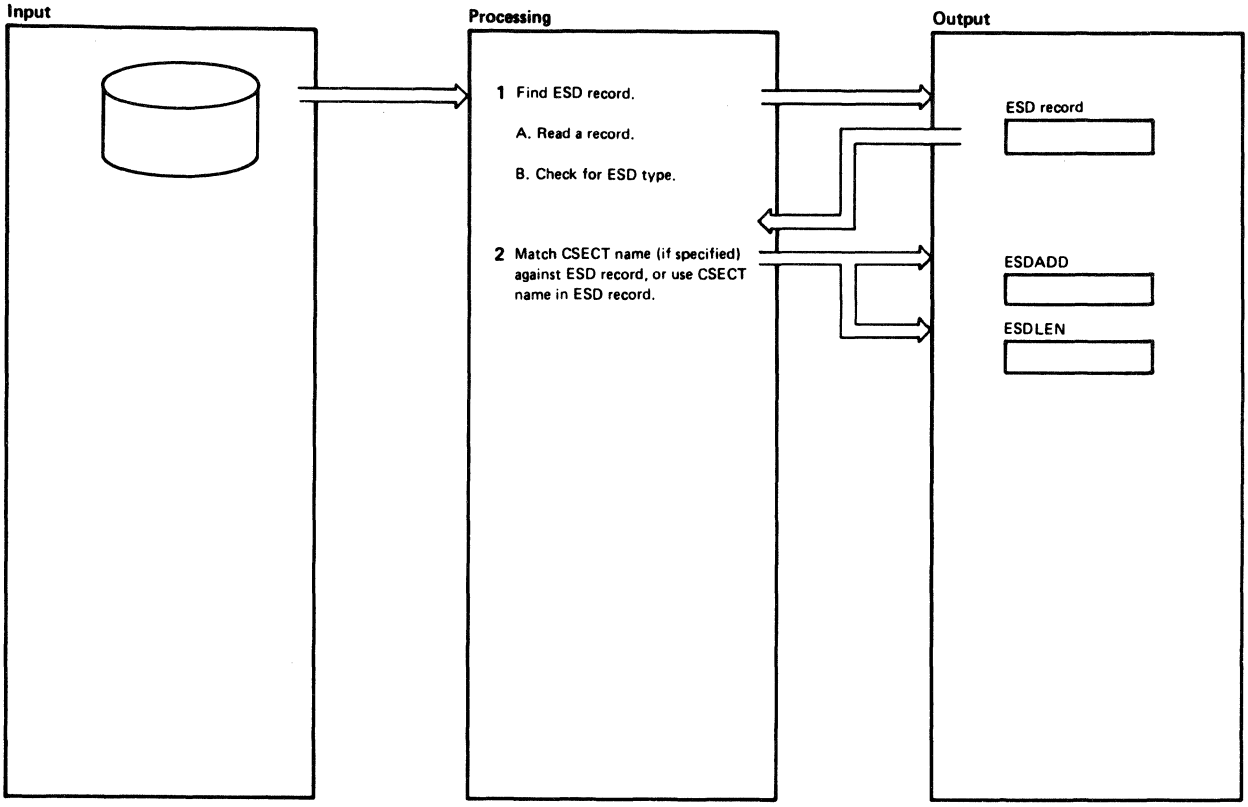
Notes	Module	Label	Ref
and return control to READINP to read another control record.			
Convert the data to hexadecimal and add the BASE value to the displacement. Check that the displacement plus the data length will fit within the CSECT. If not, issue the message. DMSZAP191W PATCH OVERLAPS – SET NO GO SWITCH and return control to READINP.	DMSZAP	EQLNTH PACKDAT INVEREP2	
C. Go to the RDTXT routine to perform the operation, then return control to READINP.	DMSZAP	GOVER RDTXT	
2			
A. Issue the message DMSZAP750I ZAP PROCESSING COMPLETE	DMSZAP	COMEND INVEREP4	
B. Close the INPUT file, if it is open, and free buffer space.	DMSZAP	CLOSEINP CLRSPCE	
C. Return to CMS.	DMSZAP	NOMORE	

Diagram 11-5. VER/VERIFY or REP and END Control Record Processing



Notes	Module	Label	Ref
1 Close input module and library files, if open.	DMSZAP	PREOPLIB CLOSELIB	
2 If MODULE was specified, locate the module name and search for the module. If the module is found, check the attributes and if they are valid, go to Step 4. Otherwise, issue one of these error messages:	DMSZAP	STFDEF	
DMSZAP210E FILE 'fn ft' IS ON A READ/ONLY DISK		LIBRO	
DMSZAP208E FILE 'fn ft' IS NOT VARIABLE RECORD FORMAT		LIBNTV	
If the module cannot be found, issue the message		PREOPLB3	
DMSZAP002W FILE 'fn ft' NOT FOUND		PREOPLB5	
and read another control record. Ignore all control records until the next NAME, DUMP, or END control record.		INVEREP2	
If LOADLIB or TXTLIB was specified, locate the first library name and search for the member. If the member is found, check the attributes and if they are invalid, issue one of these messages:		STFDEF	
DMSZAP210E FILE 'fn ft' IS ON A READ/ONLY DISK		LIBRO	
DMSZAP208E FILE 'fn ft' IS NOT VARIABLE RECORD FORMAT		LIBNTV	
DMSZAP007E FILE 'fn ft' IS NOT FIXED, 80 CHAR, RECORDS		FILENTF	
Otherwise, go to Step 3 after issuing the message		MEMFND	
DMSZAP751I MEMBER FOUND IN LIBRARY 'fn'			
If the library cannot be found, issue the message		PREOPLB3	
DMSZAP002W FILE 'fn ft' NOT FOUND		INVEREP2	
and locate the next library name and execute Step 2 again. If none of the libraries specified can be found, issue the message		LIBNTFD1	
DMSZAP002E FILE 'fn ft' NOT FOUND			
and terminate processing.		NOMORE	
3 When a library is found, read the first record. If the header record or the pointer to the directory is invalid, issue the message	DMSZAP	OPENFILE PREOPLB4	
DMSZAP056E FILE 'fn ft' CONTAINS INVALID RECORD FORMATS		INVFORM	
Otherwise, locate the directory record and search for the member name. If the file is a CMS-only (not OS) TXTLIB file and the member name cannot be found, search for the CSECT name. If a member name or CSECT name is found, go to the READCESD routine to find a CSECT record.		READLIB CHKMEM CHKCSECT	
4 If the file is a MODULE, compute the length of the module and its starting and ending addresses. Determine if a map is present and, if not, that no CSECT name was specified, then exit. If a CSECT name was specified, issue the message	DMSZAP	CHKLDTBL NOTABLE INVEREP2 CHKLDCST	
DMSZAP246W NO LOADER TABLE PRESENT FOR MODULE 'fn' SET NO GO SWITCH			
then exit. If a module map is present, locate the map record and read it. If the map record cannot be found, issue the message			
DMSZAP056E FILE 'fn ft' CONTAINS INVALID RECORD FORMATS		INVFORM	
Otherwise, locate the CSECT specified or the first CSECT in the map, and determine its length, and return control to caller. If the CSECT specified cannot be found, issue the message		LDRLOOP FINDCLNTH INVEREP2	
DMSZAP194W CSECT NOT FOUND IN 'fn ft' – SET NO GO SWITCH			
and read another control record.			

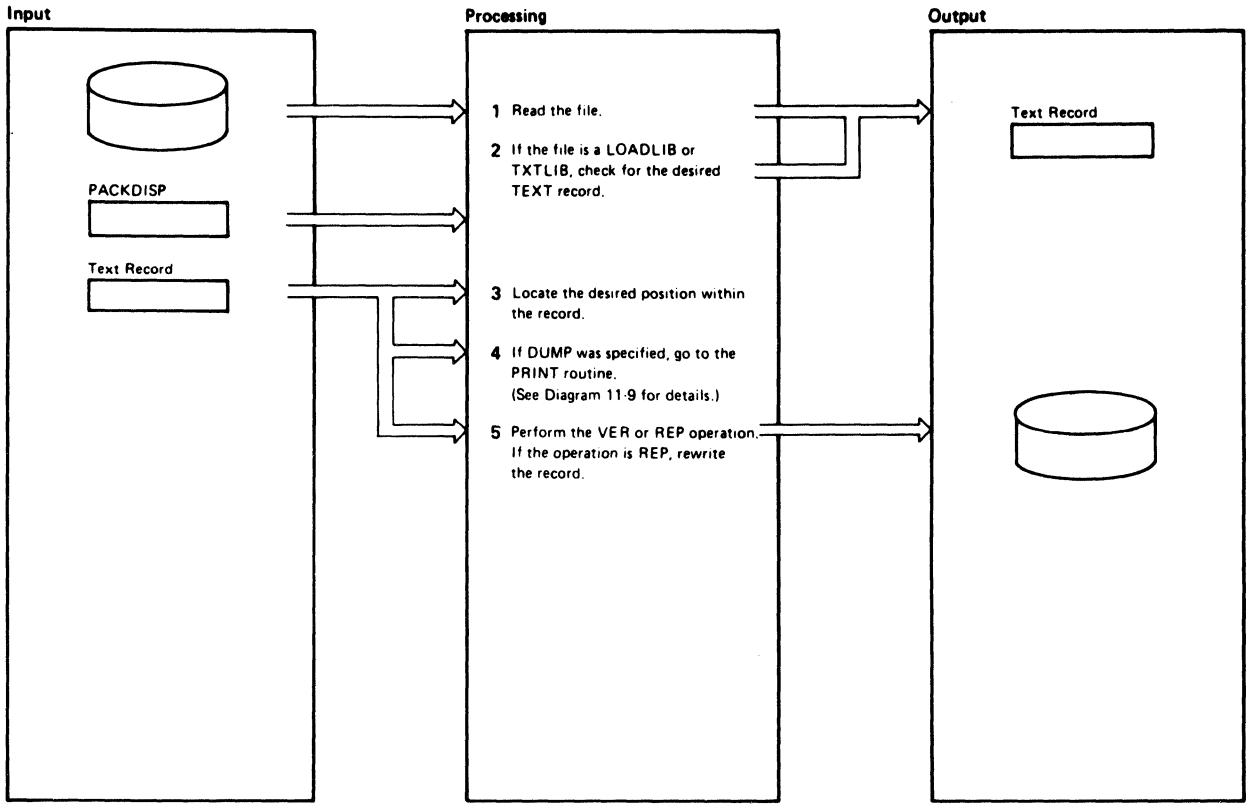
Diagram 11-6. Opening the File



Notes	Module	Label	Ref
1 Read a LOADLIB or TXTLIB member record. Check to see if it is an ESD-type record. If not, re-execute Step 1.	DMSZAP	READCESD TXTESD RDLIB	
2 If a CSECT name was specified in the NAME or DUMP control record, compare it with the CSECT name(s) in the ESD record(s).		SEARCHSD	
If there is a match, save the starting address and length. If there is no match, issue the message DMSZAP194W CSECT NOT FOUND IN 'fn ft' – SET NO GO SWITCH		CSECTFND NOCESD2	
If no CSECT name was specified in the NAME or DUMP control record, use the first CSECT named in an ESD record. If ALL was specified in a DUMP control record, use the next CSECT name encountered in an ESD record.			
Control then returns to caller.		MEMEND	

Notes	Module	Label	Ref

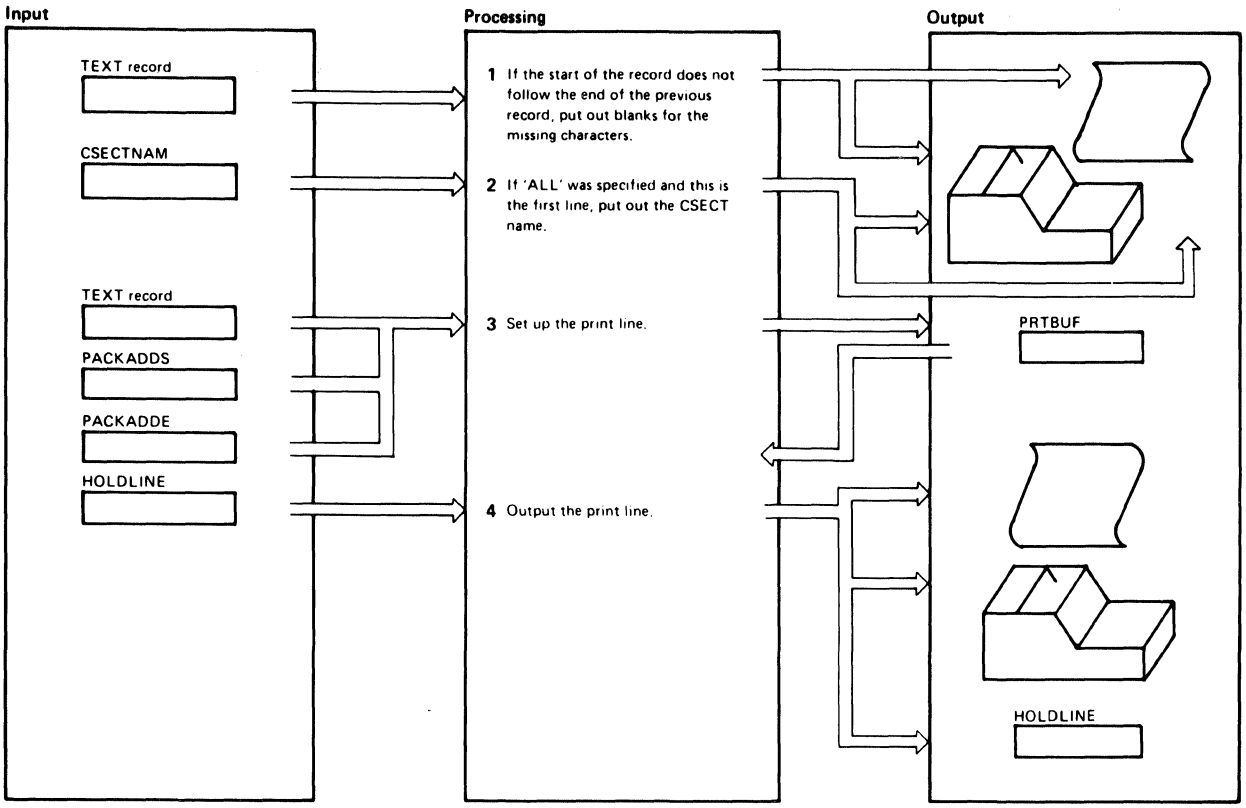
Diagram 11-7. Finding the CSECT



Notes	Module	Label	Ref
1 Read the next record of the file. If the file is a module, go to step 3.	DMSZAP	RDTXT	
2 If the file is a TXTLIB, check for the desired record. If not, repeat step 1. Otherwise, check for valid characters. If there are no valid characters (that is, if the area is a Define Storage area), and the operation is VER or REP, issue the message DMSZAP248W INVALID VER/REP DISP – SET NO GO SWITCH If there are no valid characters, but the operation is DUMP, determine the length of the gap and handle it as a TEXT record.	DMSZAP	RDTXTLIB RDTXTFND	
If the file is a LOADLIB, check for the desired record. When it is found, check for valid characters as with a TXTLIB and, if valid, read the next record for the actual text.	DMSZAP	RDLDLIB RDLDCHK	
3 Determine the position within the record.	DMSZAP	CHKVER	
4 If the operation desired is DUMP, go to the PRINT routine to print out lines.	DMSZAP	VERCHK PRTDUMP	
5 If the operation is REP, replace each byte read with the data supplied in the REP control record. When the end of the record is reached or the REP operation is completed, rewrite the record.	DMSZAP	VERLOOP VERIFY1 VERIFY2 WRLIB	

Notes	Module	Label	Ref
If the operation is VER, compare each byte read with the data in the VER control record. If they do not agree, issue the message DMSZAP200W VERIFY REJECT -SET NO GO SWITCH		VERLOOP VERIFY1	
If another record is required, go to step 1. Otherwise, control return to caller.		RDTXEND	

Diagram 11-8. Reading the Text



Notes	Module	Label	Ref
1 If the start of a new record does not match the end of the previous record, or the requested start of the dump is not found, insert blanks in the output record to represent the bytes not in the file.	DMSZAP	PRTDUMP SETBLANK	
2 If 'ALL' was specified and this is the first line of the CSECT, output the CSECT name.	DMSZAP	NEWLIN PRTHDR	
3 If a line has been started, finish the line. If not, set up the new line. Determine the address of the new line, check that the line does not exceed the requested end of the dump, and move characters from the record into the line. If the line does exceed the requested end of the dump or the record is exhausted, fill the output line as much as possible, convert its characters for printing, and save the pointers. Return control to caller.	DMSZAP	FINLINE NOFSTLN SETADD SETHXLN SETHXLNA CHARCONV PRTRDNXT	
4 When the line is ready for printing, convert the non-printing characters to periods, and compare the line to the previous line. If there is a match, save the address of the current line. If there is no match, and addresses have been saved, print the message LINES xxx TO xxx SAME AS ABOVE. Otherwise, print the line and save it in HOLDINE.	DMSZAP	CHARCONV PRTLINE CHKDUP NOTDUP PRTLIN2	

Notes	Module	Label	Ref

Diagram 11-9. Printing the Dump

Program Organization

This section contains a program description of the DMSZAP module.

DMSZAP

The ZAP service program.

Entry Point

DMSZAP — via the command ZAP.

Attributes

Reusable, not disk resident.

Entry Conditions

R1: Address of the input parameter list
R15: Address of the entry point

Register Usage

R1: Address of the input parameter list
R2-8: Work registers
R9: Base registers
R10: Link register
R11-12: Base registers
R13: Address of the save area
R14: Return address
R15: Return code

Calls to Other Routines

DMSBRD — To read input disk files.
DMSBWR — To write output disk files as a result of REP operation.
DMSERR — To handle calls from DMSERR and LINEDIT macros.
DMSFNS — To close input and output files.
DMSPRT — To handle PRINT command.
DMSSMN — To handle OS GETMAIN and FREEMAIN macros.
DMSSTT — To provide a copy of an FST.
DMSSVT — To process OS macros.

External References

None.

Data Areas

File Status Table

Exit Conditions

R15: Return code

Directory

Figure 11-2 is an alphabetical list of the major labels of the ZAP program. The associated method of operation diagrams are indicated and a brief description of the operation performed at the point in the program associated with each label is included.

Label	Diagram	Description
BASEREC	11-4	Processes a BASE control record. Scans for displacement.
CHKLDTBL	11-6	Locates a CSECT (for a module file) if a name is given.
CHKMEM	11-6	Checks for a member, or, if a CMS TXTLIB, for a CSECT.
CLOSELIB	11-6	Finishes the specified library or module.
CLOSINP	11-5	Closes the input file.
CLRSPCE	11-5	FREEMAINs buffer space.
CONEND	11-5	Processes an END control record.
CONSOPT	11-2	Sets the TERM option.
DECODE1	11-4 11-5	Checks that a field is less than six digits.
DECODE2	11-4 11-5	Checks that a field is an even number of digits.
DMSZAP	11-1	Saves the input registers and sets addressability.
DOWTO		Does a write-to-operator for messages when in terminal mode.
DUMPREC	11-3	Gets the location of the dump and prints it.
FDEFINP	11-2	FILEDEFs the input DCB and opens it.
FINDMEM	11-6	Locates the beginning of a member.
FNDCLNTH	11-6	Locates the boundary of a CSECT.
INITOPEN	11-1	Opens input (if specified) and output (printer) files.
INPTOPT	11-2	Sets the INPUT option.
INVEREP	11-2	Processes the error message for an invalid control record and closes the SYSLIB file.
NAMEREC	11-4	Processes a NAME control record. Scans for the member name and CSECT name.
NAMFOUND	11-2	Branches to the appropriate routine when a keyword is found in the table.
NEWLIN	11-9	Prints full lines.

Figure 11-2 (Part 1 of 2). The ZAP Program Label Directory

Label	Diagram	Description
NOMORE	11-5	Gets the error code and prior save area address, restores the registers, and returns to DMSITS.
NOPRTOPT	11-2	Sets the NOPRINT option.
OPENFILE	11-6	Opens a library.
PREOPLB1	11-6	Gets the first library name address.
PREOPLB4	11-6	Reads a ZAP file and locates a member (CSECT for a MODULE file if a name was given).
PREOPLIB	11-6	Opens ZAP files and looks for the library name, if given.
PRINTOPT	11-2	Sets the PRINT option.
PRTCARD		Prints a card image.
PRTDUMP	11-9	Prints the requested dump.
PRTHDR	11-9	Prints the name of the CSECT being dumped.
PRTLINE	11-9	Prints a dump line.
RDCARD	11-2	Requests input from the terminal.
RDCARD2	11-2	Reads an input control record file.
RDLDLIB	11-8	Analyzes LOADLIB records.
RDLIB	11-7	Reads the specified library or module.
RDTXT	11-8	Reads a library searching for the record to be verified or replaced.
RDTXTLIB	11-8	Analyzes TXTLIB records.
READCESD	11-7	Reads a CESD record of a member.
READINP	11-2	Reads a control record from the input file. Writes the control record to the output (SYSPRINT) file. Scans the first keyword from the control record.
SCANKEY1	11-2	Scans control records.
SCANLINE	11-2	Checks the command line for validity.
SEARCHSD	11-7	Searches a CESD record for an ESD entry with a CSECT name.
SETBLANK	11-9	Spaces over a DS area.
STFDEF	11-6	Issues a STATE for a library file, checks that the disk is in Read/Write mode.
TABLOOK	11-2	Look for a keyword in the table.
TXTESD	11-7	Finds a TXTLIB CSECT.
WRCARD	11-2	Writes a control record and messages to SYSPRINT file.
WRLIB	11-8	Updates the specified library or module.

Figure 11-2 (Part 2 of 2). The ZAP Program Label Directory

Data Areas

The File Status Table is used by the DMSZAP module:

4 Bytes

0	Filename		
8	Filetype		
16	DATE LAST WRITTEN (Note 1)		
20	Write Pointer Relative Record Number	22	Read Pointer Relative Record Number
24	Filemode	26	Number of Records in File
28	Disk Address of First Chain Link	30 Fixed Variable (Note 2)	31 Flag Byte (Note 3)
32	Record Length (F) Maximum Record Length (V)		
36	Number of 800-Byte Data Blocks		Year (Note 4)

Notes:

1. *Date last written is in packed decimal format MM DD HH MM; for example, 02 20 14 07 represents February 20, 2:07 p.m.*
2. *F = Fixed-length records. V = Variable-length records.*
3. *Flag Byte = 0.*
4. *Year is in character form; for example, '72' for 1972.*

Figure 11-3. File Status Table Entry

Diagnostic Aids

The ZAP Command Processor (DMSZAP)

Message Code	Label	Diagram	Message Text
DMSZAP001E	SCANLINE	11-2	NO FILENAME SPECIFIED
DMSZAP002W	PREOPLB5	11-6	[INPUT/OVERLAY] {FILE[(s)]/DATA SET} ['fn[ft[fm]]'] NOT FOUND
DMSZAP002E	FDEFINP PREOPLB3	11-2 11-6	FILE 'fn ft fm' NOT FOUND
DMSZAP003E	SCANLINE	11-2	INVALID OPTION 'option'
DMSZAP007E	FDEFINP STFDEF	11-6	FILE 'fn ft fm' [IS] NOT FIXED, 80 CHAR. RECORDS
DMSZAP014E	SCANLINE	11-2	INVALID KEYWORD 'function'
DMSZAP047E	SCANLINE	11-2	NO FUNCTION SPECIFIED
DMSZAP056E	PREOPLB4	11-6	FILE 'fn ft[fm]' CONTAINS INVALID {NAME/ALIAS/ENTRY/ESD} RECORD FORMATS
DMSZAP070E	SCANLINE	11-2	INVALID {PARAMETER 'parameter'/ARGUMENT 'argument'}
DMSZAP104S	PREOPLB4 CHKLDTBL RDCARD2 RDLIB	11-6 11-6 11-2 11-6	ERROR 'nn' READING FILE 'fn ft fm' FROM DISK
DMSZAP190W	INVEREP	11-2 11-3 11-4 11-5	INVALID CONTROL RECORD OR NO GO SWITCH SET
DMSZAP191W	DUMPREC GOODTHRE	11-5	PATCH OVERLAPS - SET NO GO SWITCH
DMSZAP192W	DECODE1 GOODTHRE	11-3 11-5 11-4	ERROR - ODD NUMBER OF DIGITS - SET NO GO SWITCH
DMSZAP193W	GOODTHRE	11-5	PRECEDING CONTROL RECORD FLUSHED
DMSZAP194W	OPENFILE READCESD	11-6 11-7	CSECT NOT FOUND IN {member'membername'/MODULE 'modulename'} -SET NO GO SWITCH
DMSZAP195W			BASE VALUE INVALID-SET NO GO SWITCH
DMSZAP200W	VERIFY1	11-8	VERIFY REJECT - SET NO GO SWITCH
DMSZAP208E	STFDEF	11-6	FILE 'fn ft' IS NOT VARIABLE RECORD FORMAT
DMSZAP210E	STFDEF	11-6	FILE 'fn ft' IS ON A READ/ONLY DISK
DMSZAP213W	BASEREC	11-4	BASE VALID INVALID - SET NO GO SWITCH

Figure 11-4 (Part 1 of 2). ZAP Command Processor (DMSZAP) Messages

Message Code	Label	Diagram	Message Text
DMSZAP245S	WRCARD	11-2	ERROR 'nnn' ON PRINTER
DMSZAP246W	CHKLDTBL	11-6	NO LOADER TABLE PRESENT FOR MODULE 'fn' - SET NO GO SWITCH
DMSZAP247W	PREOPLB3	11-6	MEMBER 'name' NOT FOUND - SET NO GO SWITCH
DMSZAP248W	RDTXTLIB RDLDLIB	11-8 11-7	INVALID VER/REP DISP - SET NO GO SWITCH
DMSZAP249I			DUMMY LOG ENTRY IN FILE 'fn ZAPLOG fm'
DMSZAP750I	CONEND	11-5	ZAP PROCESSING COMPLETE
DMSZAP751I	OPENFILE	11-6	MEMBER 'name' FOUND IN LIBRARY 'libname'

Figure 11-4 (Part 2 of 2). ZAP Command Processor (DMSZAP) Messages

Chapter 12. DMSIFC and DMSREA—EREP/Error Recording Interface

Introduction

The method of editing error records accumulated on the VM/SP HPO error recording area or stored on other devices makes use of the OS/VS EREP Edit and Print programs. To use these programs from a virtual machine environment requires the use of the DMSIFC module which is called by DMSITS when the CPEREP (EXEC) command is processed.

DMSIFC loads DMSREA and several modules of OS/VS EREP into main storage and then passes control to OS/VS EREP.

Prior to passing control to EREP, DMSIFC does the following:

- Issues FILEDEFs for files needed by OS/VS EREP
- Reads control parameters from the user and puts them into an OS-compatible parameter (PARM) list format to be passed to OS/VS EREP
- Creates a SYSIN file of control parameters from the control parameters that have been entered
- Uses the HNDSVC macro instruction to prepare for trapping the EXCPs (SVC 0) that OS/VS EREP will issue when it attempts to read records from the SY1.LOGREC data set.

Note: HNDSVC is also used to prepare to trap BLDLs (SVC 18) that OS/VS EREP will issue.

The several modules of OS/VS EREP that must be loaded by DMSIFC are those that contain VCONs or that are needed in the process of resolving VCONs. DMSIFC invokes the CMS INCLUDE command dynamically to load these OS/VS EREP modules from CPEREP's two TXTLIB files. Other modules of OS/VS EREP that do not contain VCONs are loaded later (from the two TXTLIB files) by OS/VS EREP itself as they are needed.

DMSIFC passes control to OS/VS EREP by executing an OS LINK (to EREP's IFCEREP1 module, which has already been loaded). The OS-compatible parameter list built by DMSIFC is passed to IFCEREP1 at this time and OS/VS EREP begins to execute.

EREP issues set EXCPs for I/O to the OS SYS1.LOGREC data, which are intercepted by CMS. CMS transfers control back to DMSIFC, which simulates the EXCPs so that they appear to access a SYS1.LOGREC data set. This simulation results in calls to DMSREA to supply records contained in the VM/SP HPO error recording area.

EREP issues BLDLs (SVC 18) to determine whether or not EREP modules needed for certain error records are present in the TXTLIBs. The standard CMS simulation of OS BLDL does not include the JOBLIB/STEPLIB form of BLDL which EREP uses here. Therefore, these BLDLs are intercepted and are simulated by DMSIFC.

When EREP is finished executing, it exits (returns to DMSIFC which invoked it). Before returning to CMS, DMSIFC does some cleaning up. Temporary files are erased and FILEDEFs issued by DMSIFC are cleared with the following exceptions: the EREPPT, ACCIN, TOURIST and ACCDEV FILEDEFs are not cleared because they may have been entered by the user or by DMSIFC but DMSIFC has no way of knowing which. Since they should not be cleared if they were entered by the user, DMSIFC never clears them.

In order to make use of the CPEREP command, both of the following publications are required. The first publication provides general information on the use of the command and detailed information on command operands applicable to VM/SP HPO. The second publication provides detailed information on the operands that are common to both VM/SP HPO and OS/VS.

VM/SP HPO Release 4.2 OLTSEP and Error Recording Guide, Order No. ST00-1901

OS/VS Environment Recording Editing and Printing (EREP) Program, Order No. GC28-0772.

Program logic information describing OS/VS EREP is contained in:

OS/VS Environment Recording Editing and Printing (EREP) Program Logic, Order No. SY28-0773.

Method of Operation

This section describes the interface between CMS (the Conversational Monitor System) and the OS/VS EREP program. Diagrams 12-1 and 12-2 describe the functions of the interface modules and serve as a guide to the program listings. The labels shown indicate the closest, nonmacro expansion label to the function being documented. These diagrams are not terribly detailed, therefore, some functions are not shown. Use the Directory and Program Organization section to find the labels in the program listings for any routines that are not shown in the Method of Operation section. Figure 12-1 shows the relationship of these diagrams.

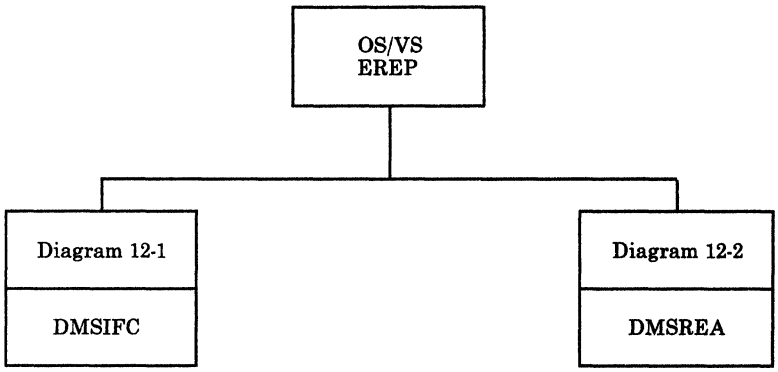
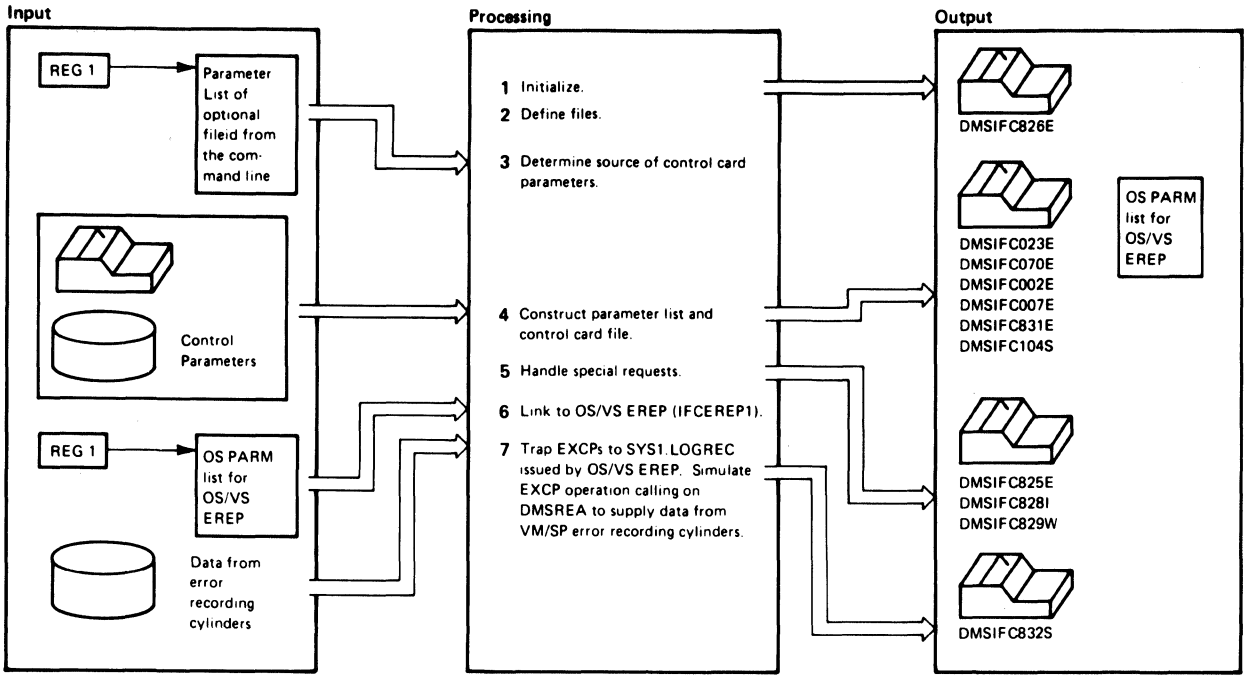


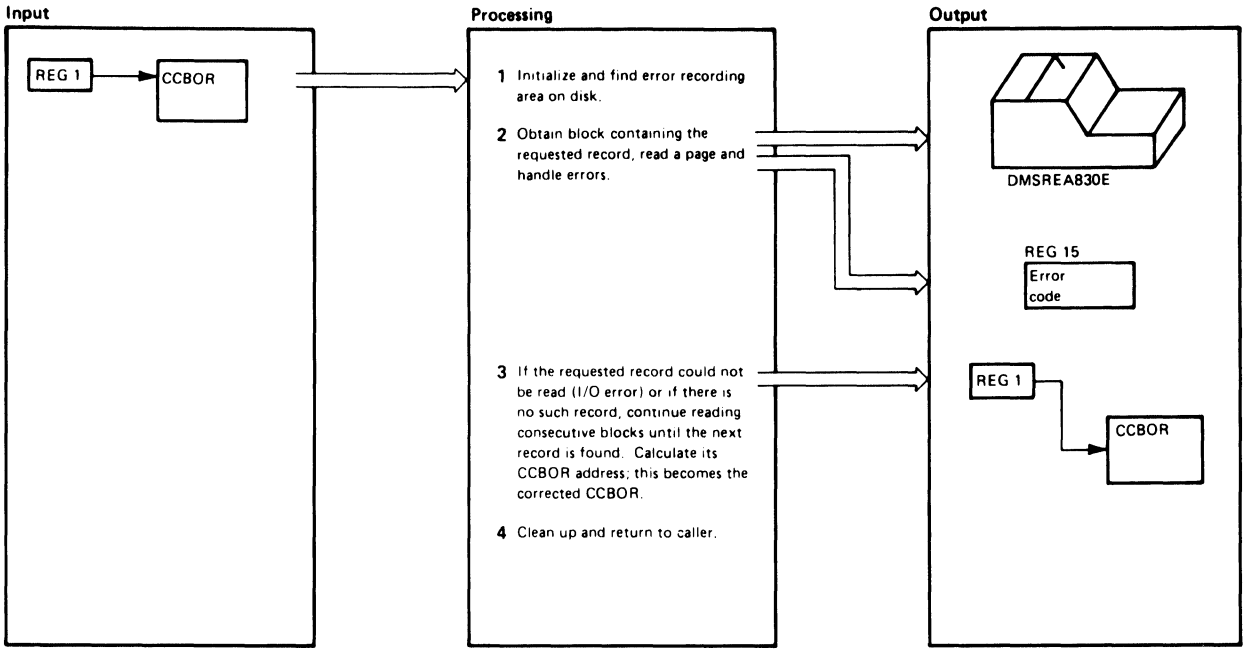
Figure 12-1. Key to EREP/Error Recording Interface Method of Operation Diagrams



Notes	Module	Label	Ref
1 The initialization procedures include: • Standard linkage and addressability functions. • Loading and resolving VCONs in OS/VS EREP decks. • Loading DMSREA. • Turning off flag in CMS nucleus to cause OS simulation. • Setting COMPSWT in CMS nucleus to load LINK and LOAD macros to be entered in TEXT files. • Establishing handling of SVC 76, SVC 18, and SVC 0.	DMSIFC	DMSIFC	Diagram 12-1
2 Invoke FILEDEF to define: • Printer file (EREPPT). • SYSIN file (SYSIN). • Dummy file for SYS1.LOGREC (SERLOG). • Error file (TOURIST). • Work file (DIRECTWK). • Accumulation tape file (ACCDEV). • History input tape (ACCIN).		NORWDISK RDYACC RDYHIST OPER12	
3 Determine where control parameters are to be taken from (Control file or terminal).		HAVETYPE NOEXTRA BADATTR GOODATTR	
4 Set up to read parameters. Obtain storage for OS PARM list to be passed to EREP. Read control parameters, generating the OS PARM list and a SYSIN file as output. Call subroutine to read control parameters. Handle errors.		PARMWORK RDERR1 PLISTBLD	Diagram 12-1
5 If CLEAR is specified with other parameters, type an error message. If CLEAR is specified properly, call subroutine to erase error records from the VM/SP error recording cylinders. Subroutines handle each parameter information.		WANTCLR CLEARRTN	

Notes	Module	Label	Ref
CLEARF parameter (determines validity by examining processor identity. If not 3031, 3032, or 3033 processor reject command but if valid, erase error records from the error recording cylinders then initialize SRF frames to the beginning of the error recording cylinders.)		HCLEARF	
• CLEAR parameter.		HCLEAR	
• TERMINAL parameter (stops reading from control file on disk and goes to terminal to read additional control parameters).		HTERM	
• SHARE control statement.		HSHARE	
• ACC parameter.		HACC	
• HIST parameter.		HHIST	
• MERGE parameter.		HMERGE	
• THRESHOLD parameter.		HTHRES	
• ZERO parameter.		HZERO	
• DASDID control statement.		HDASDID	
• LIMIT control statement.		HLINIT	
• CONTROLLER control statment		HCONTROL	
6 Load the address of the word that points to the OS PARM list built for OS/VS EREP and LINK to IFCEREP1.			
7 EXCP SVCs from EREP are intercepted and simulated so they appear to access a SYS1.LOGREC data set. Simulation causes calls to DMSREA for VM/SP error records. BLDL SVCs from EREP are also trapped and simulated by DMSIFC.		DMSIFC0 DMSIFC18	

Diagram 12-1. DMSIFC



Notes	Module	Label	Ref
1 The initialization procedures include: • Saving registers. • Setting return code to zero. • Issuing DIAGNOSE X'2C' to locate beginning of error recording area and number of cylinders. • Setting and checking "first time" switch. • Checking CCBOR address passed for validity. *Note:* A CCBOR disk address is a disk addressing format devised solely for use in CPEREP and resembles the commonly used CCHHR disk address. In a CCBOR address the fields have the following meaning: Field Meaning CC Relative cylinder within the VM/SP error recording area, for example: CC = X'0000' for the first cylinder of the error recording area. = X'0001' for the second cylinder of the error recording area. B The number of the desired 4K block within the cylinder. The first 4K block in a cylinder is X'01'. 0 Zero. R The number of the desired record within the 4K block. The first record in a block is X'01'.	DMSREA	DMSREA FIRSTSW OPER4	Diagram 12-2

Notes	Module	Label	Ref
2 DMSREA converts the CCBOR address to a VM/SP Control Program Internal Format address and issues a DIAGNOSE X'30' to read the block into the buffer. If the requested block is found, return to caller. If specified cylinder is outside error recording area, sets error code in register 15 for invalid cylinder. If end of cylinder and no more cylinders are available, sets register 0 or zero, indicating end-of-file to caller; otherwise, advance to next cylinder. If an I/O error occurs so that the block could not be read, issue message DMSREA830E. 3 If requested record was not found, read next block and return first record from this block. If block is empty or unreadable, continue reading blocks until a record is found or until end-of-file is reached. Use CCBOR address of the record found as the corrected CCBOR value to be returned to the caller. Make register 1 point to this CCBOR address. Note: The CCBOR record addresses are passed back to OS/VS EREP (as a result of the EXCP simulation) as if they were CCHHR addresses. EREP never notices the difference and, as a result, EREP uses CCBOR addresses in all its I/O operations to the SYS1.LOGREC data set. 4 Restore registers (except output parameter registers) and return to caller.		OPER5 OPER7 OPER16 OPER17 OPER7 OPER9 OPER10 OPER15	

Diagram 12-2. DMSREA

Program Organization

This section includes program descriptions of modules DMSIFC and DMSREA.

DMSIFC

Allows virtual users to edit and print VM/SP HPO error recordings under CMS via the OS/VS EREP Edit and Print Program (IFCEREP1).

Entry Point

DMSIFC

Routines Called

IFCEREP1 — via LINK to edit and print VM/SP HPO error recording area.
DMSREA — via BALR to read a specified record from the VM/SP HPO error recording area.
DMSLAD — via BALR to determine which read/write disk has the most space.
DMKIOG — via DIAGNOSE to clear requested recording area.
STATE/STATEW — via SVC to perform CMS functions.
ERASE — via SVC to perform CMS functions.
INCLUDE — via SVC to perform CMS functions.

Attributes

Nonreusable, CMS User Area, and called by CMS.

Registers at Entry

R1: CMS parameter list address
R13: Save area address
R14: Return address

Registers at Exit

R0-R14: Restored
R15: One of the following return codes:

Return Code	Meaning
12	CLEAR specified with other parameters.
24	An invalid parameter or no filetype was specified.
28	The file was not found.
32	The file was not a fixed-length format.
56	GLOBAL command was not issued for CPEREP's TXTLIBs.

Return Code	Meaning
60	An I/O error caused one or more of the 4K blocks of error records to be skipped.
62	More than the maximum number of characters in options specified.
88	Attempt to set to zero was suppressed. Requires privilege class F.
100	Error reading file from disk.

Register Usage

R0-R1: Parameter registers
R2-R9: Scratch
R10-R11: Spares, not used
R12: Base register
R14-R15: Link registers

External References

CURRSAVE — Contains address of the current system save area when control is received to handle an SVC as requested by the HNDSVC macro.

OSSFLAGS — OS simulation flags in the NUCON area.

DOSFLAGS — DOS simulation flags in the NUCON area.

AADTLKW — Contains address of routine that determines which read/write disk has the most space. (In the NUCON.)

TXTLIBS — Indicates whether or not any TXTLIBs have been globaled. (In the NUCON.)

TXTDIRC — Indicates whether or not any TXTLIBs have been globaled. (In the NUCON; points to the first directory in the chain of global TXTLIB directories.)

The functions performed by DMSIFC can be summarized as follows:

1. Performs standard linkage and addressability functions.

2. Invokes CMS LOAD function to load and resolve VCONs in about a dozen EREP object decks.

   Note: All other EREP object decks are brought into storage later, as needed, by OS LOAD and LINK macros issued by OS/VS EREP.

3. Invokes STRINIT function. Indicates that area above presently loaded programs is the beginning of free storage.

4. Turns off the DOSSVC flag in the CMS nucleus so that OS simulation can be used. Sets COMPSWT in CMS nucleus so that OS LOAD and LINK macros bring in TEXT files rather than module files. Invokes OS LOAD to load DMSREA into storage and saves its address so it can be called later during the EXCP simulation.

5. Establishes handling of SVC 76 (error log), SVC 18 (BLDL), and SVC 0 (EXCP).

6. Invokes FILEDEF function to define:

   - Printer file for EREP
   - SYSIN file to be created for EREP
   - Dummy file for EREP to open and close as SYS1.LOGREC
   - "TOURIST" error file to the terminal
   - DIRECTWK work file on disk.

7. Gets the command line arguments and determines if a control file is provided for input. If so, sets up to read parameters from the control file, otherwise, sets up to read parameters from the terminal.

8. Issues a DMSFREE macro to get storage for building OS parameter list to be passed to EREP.

9. Gets input parameters (from control file or terminal) and constructs equivalent OS/VS EREP parameter list and SYSIN control card file.

10. If CLEAR was specified, and it was not the only parameter specified, types an error message to the terminal and does housekeeping and exits to CMS.

11. If CLEAR was specified correctly, calls a subroutine to issue the DIAGNOSE that clears the appropriate records from the VM/SP HPO error area, then does housekeeping and exits to CMS. If CLEARF was specified, read CPU and director frames from SRF device and write on error area.

12. Invokes FILEDEF to define the accumulation tape file if requested. Issues the tape control macros necessary to position tape for subsequent write operations.

13. Invokes FILEDEF to define history input tape if requested and makes sure that it is rewound.

14. Links to OS/VS EREP (IFCEREP1).

15. Simulates BLDL SVCs issued from OS/VS EREP. Simulates EXEC SVCs issued from OS/VS EREP so they will appear to access a SYS1.LOGREC data set. EXCP simulation will result in calls to DMSREA to get records from VM/SP HPO error recording area.

16. Eventually OS/VS EREP is done and control returns from that LINK done above.

17. Housekeeps all indicators and switches, frees any storage obtained for the OS parameter list area, clears handling of SVC 0, SVC 18, and SVC 76; and clears any FILEDEFs that were set up by CPEREP.

18. Exits to CMS.

DMSREA

Reads a specified logical record from the error recording area and returns it to the caller.

Entry Point

DMSREA

Routines Called

DIAGNOSE X'2C' to find the beginning of the recording area on the system disk, and the size of the error recording area.

DIAGNOSE X'30' to read a page size record from the error recording area.

DMSERR via macro SVC to write error messages to the console.

Attributes

Nonreusable, CMS User Area, enter via CALL.

Registers at Entry

R1: Address of CCB0R DASD record address
R13: Save area address
R14: Return address

Register at Exit

R0: Nonzero: address of variable-length record being returned.
The first 4 bytes are the record descriptor word containing the record length.
Zero: end-of-file; no record was at or beyond the entered address.
R1: Address of CCB0R DASD record address (sometimes corrected).
R13: Save area address.
R15: One of the following return codes:

Return Code	Meaning
00	Nothing unusual.
04	Empty 4K block skipped.
08	Invalid CC value in CCB0R address that was entered.
60	I/O error accompanied by message DMSIFC830E.

Register Usage

R0-R9: Scratch
R10-R11: Spares, not used
R12: Base
R13: Save area address
R14-R15: Scratch

External References

None.

The functions performed by DMSREA can be summarized as follows:

1. Issues the DIAGNOSE command to find the beginning of the VM/SP HPO error recording area and the size of the area.

2. Reads a requested record from the VM/SP HPO error recording area.

3. Returns the next logical record to the caller when the requested record does not exist or cannot be read and revises the caller's specified CCB0R address accordingly.

4. Handles errors.

Directory

Figure 12-2 is an alphabetical list of the major labels of modules DMSIFC and DMSREA. The associated method of operation diagrams are indicated and a brief description of the operation performed at the point in the program associated with each label is included.

Label	Diagram	Description
BADATTR	12-1	Handles file not fixed.
CLEARRTN	12-1	Logically erases VM/SP HPO error recording area.
DMSIFC0	12-1	Handles trapped EXCPs issued by EREP.
DMSIFC18	12-1	Handles trapped OS BLDL macros issued by EREP.
OPER7	12-2	Issues I/O error reading records message.
OPER9XX	12-1	Handles specification of CLEAR when entered with other parameters.
NOEXTRA	12-1	Handles file not found.
EXIT0	12-1	Restores registers for exit from DMSIFC.
EXIT1	12-1	Clears handling of SVCs.
EXIT3	12-1	Frees storage allocated for OS parameter list.
EXIT9	12-1	Frees storage allocated for SVC simulation.
FIRSTSW	12-2	Sets indication of first time DMSREA is called.
HACC	12-1	Directs addition of ACC parameter to OS parameter list being built for EREP.
HAVETYPE	12-1	Handles the specification of an extra parameter on the CPEREP command line.
HCLEAR	12-1	Clears all error records from the error recording area.
HCLEARF	12-1	Clears SRF frame records and all error records and reformats the error recording area.
HCTLCRD	12-1	Writes CTLCRD information into SYSIN file for EREP to read.
HHIST	12-1	Directs addition of HIST parameter to OS parameter list being built for EREP.
HMERGE	12-1	Directs addition of MERGE parameter to OS parameter list being built for EREP.
HMES	12-1	Directs addition of MES and THRESHOLD parameters to OS parameter list being built for EREP.
HRDESUM	12-1	Directs addition of RDESUM parameter to OS parameter list being built for EREP.

Figure 12-2 (Part 1 of 2). DMSIFC and DMSREA Label Directory

Label	Diagram	Description
HSHARE	12-1	Writes SHARE parameter into SYSIN file for EREP to read.
HZERO	12-1	Directs addition of ZERO parameter to OS parameter list being built for EREP.
OPER4	12-2	Checks CC portion of entered CCB0R for valid range.
OPER7	12-2	Prepares for and issues DIAGNOSE command to read a page of error records.
OPER9	12-2	Prepares to read first record of next block.
OPER10	12-2	Retains address of block just read into buffer. Decides whether this block contains data or is empty.
OPER12	12-1	Handles special considerations for ACC parameter specification.
OPER13	12-1	Handles special considerations for HIST parameter specification.
OPER15	12-2	Restores registers and returns to caller from DMSREA.
OPER16	12-2	Sets error code for invlaid cylinder.
OPER17	12-2	Handles end of cylinder indication.
PARMWORK	12-1	Issues DMSFREE macro to get storage for building OS parameter list.
PLISTBLD	12-1	Adds passed parameters to OS parameter list being built for EREP.
RECLOOP	12-1	Increments counters to step through buffer until empty or end of specified record found.
RDCTLINE	12-1	Reads and returns one line of control parameters from the terminal or control file.
RDERR1	12-1	Handles errors reading control file from disk.
WANTCLR	12-1	Handles calling subroutine to perform CLEAR.

Figure 12-2 (Part 2 of 2). DMSIFC and DMSREA Label Directory

Data Areas

DMSREA

No system data areas are used by DMSREA. However, DMSREA uses 4K of unallocated storage at absolute location X'21000' as a page buffer in which to read the 4K blocks of error records.

DMSIFC

DMSIFC uses ADTECT (the ADT macro) and FSTSECT (FSTB macro) to read from but does not store into them. It uses SSAVE and NUCON also. SSAVE is the CMS system save area that saves the value of the SVC old PSW, the caller's registers, and other necessary control information required to process SVCs and return to the caller. NUCON contains all the nucleus constants for CMS. These are either listed at the end of the module or a description can be found in the *VM/SP HPO Data Areas and Control Block Logic* manual.

Diagnostic Aids

Figure 12-3 lists the messages issued by DMSIFC and DMSREA. The label of the message and the associated method of operation diagram in which it is documented are included in the list.

Message Code	Label	Diagram	Message Text
DMSIFC002E	NOEXTRA	12-1	[INPUT/OVERLAY] {FILE[(S)]/DATA SET} ['fn[ft[fm]],] NOT FOUND
DMSIFC007E	BADATTR	12-1	FILE 'fn ft fm' [IS] NOT FIXED 80 CHAR. RECORDS
DMSIFC023E	NORWDISK	12-1	NO FILETYPE SPECIFIED
DMSIFC070E	HAVETYPE	12-1	INVALID {PARAMETER 'parameter'/ARGUMENT 'argument'}
DMSIFC104S	RDERR1	12-1	ERROR 'nn' READING FILE 'fn ft fm' FROM DISK
DMSIFC825E	OPER9XX	12-1	'CLEAR' IS VALID ONLY WHEN SPECIFIED BY ITSELF
DMSIFC826E	DMSIFC	12-1	EREP TXTLIBS NOT FOUND
DMSIFC828I	CLROKAY	12-1	CPEREP ZERO OR CLEAR HAS BEEN COMPLETED
DMSIFC829W	CLEARRTN	12-1	ATTEMPTED 'ZERO' WAS SUPPRESSED. REQUIRES PRIVILEGE CLASS F
DMSIFC831E	PLISTBLD	12-1	MORE THAN 100 CHARS. OF OPTIONS SPECIFIED
DMSIFC832S	EXGENERR	12-1	SOFTWARE INCOMPATIBILITY AT THE CPEREP-EREP INTERFACE. CODE=nnn
DMSREA830E	OPER7	12-2	I/O ERROR READING A RECORD FROM THE ERROR RECORDING CYLINDERS

Figure 12-3. DMSIFC and DMSREA Messages

Chapter 13. DMKMSS — The MSS Communicator

Introduction

The DMKMSS program operates under the control of either OS/VS1 or OS/VS2 (MVS) in a virtual machine. It is a communications interface between the control program and the MSS Mass Storage Control. It uses a combination of CP-generated attention interrupts on a virtual I/O device, the DIAGNOSE code X'78' instruction, and OS/VS SVC 126 to provide communications.

Requests are received from CP in response to a DIAGNOSE code X'78' instruction issued by DMKMSS. They are passed to the MSC using the standard OS/VS SVC 126. Responses are received from the MSC and returned to CP using diagnose.

Method of Operation

This section describes the two major sections of the DMKMSS program.

Diagram 13-1 shows initialization using OS/VS control blocks.

Diagram 13-2 shows the processing of a request from CP.

Figure 13-1 shows the relationship of these diagrams.

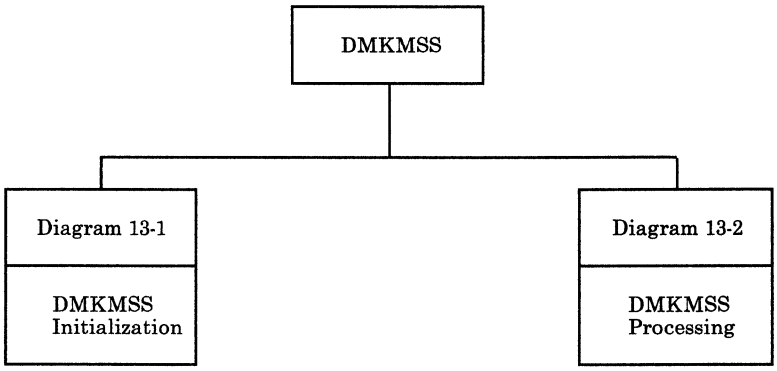
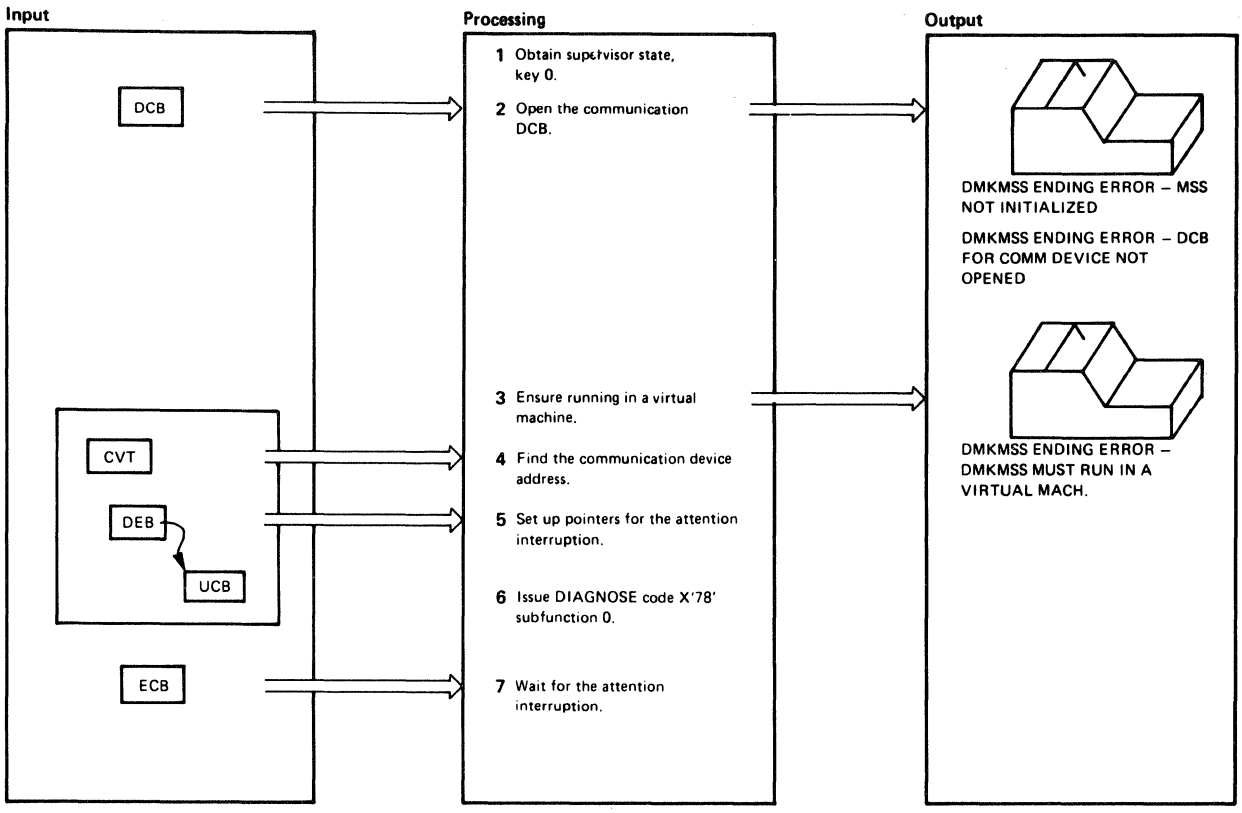


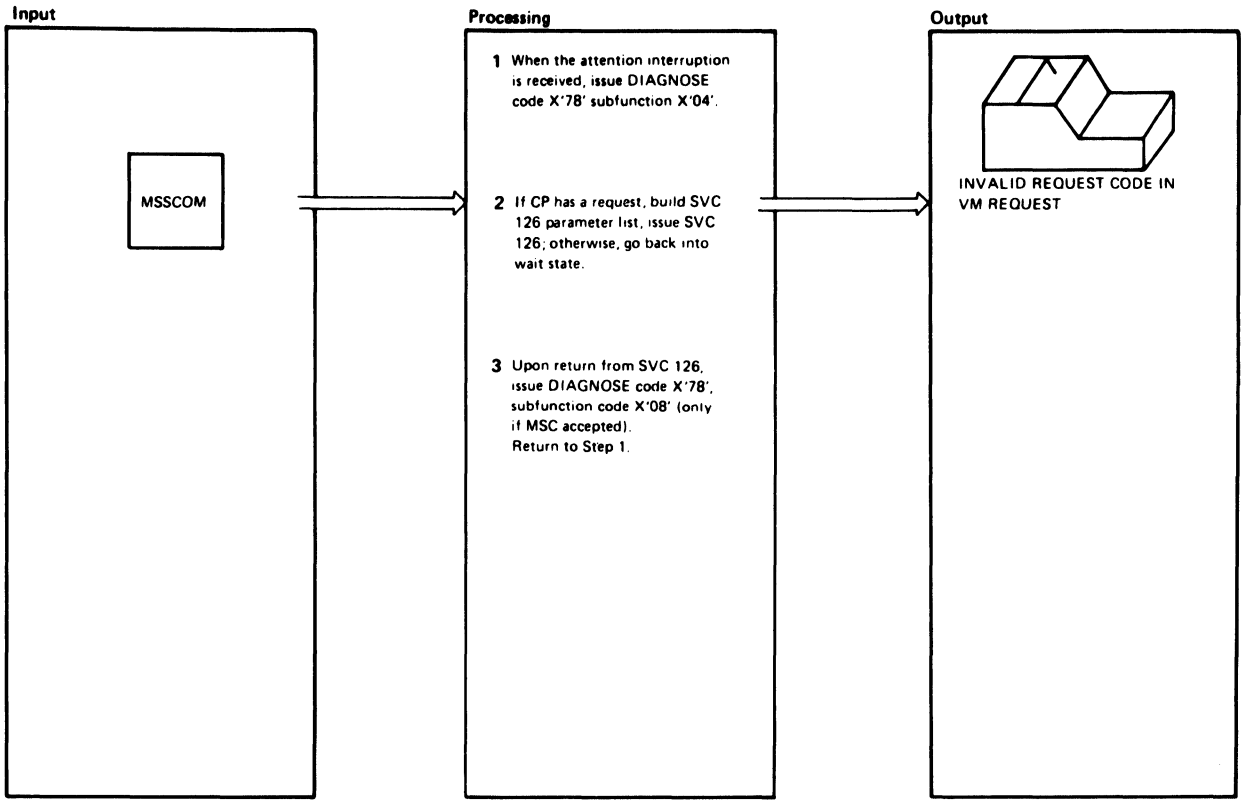
Figure 13-1. Key to the DMKMSS Method of Operation Diagrams



Notes	Module	Label	Ref
1 Use the VS MODESET SVC to get into supervisor state, key 0.	DMKMSS		
2 Use the VS OPEN SVC to connect the DCB to the VS control block. If MSS initializes incorrectly, issue message. If the DCB for the communication device does not open, issue message.		RF00092 RF00182	
3 Use the STIDP instruction to ensure running in a virtual machine. If not running in a virtual machine, issue message.		RF00082 RF00190	
4 Follow pointers through the DCB, DEB, and UCB control blocks to find the channel/unit address assigned by the VS scheduler.		L1	
5 Set the MSC's attention table index in the communication device's UCB. Also store the address of the ECB to be waited on in an unused field of this same communication UCB.			
6 Build and issue the DIAGNOSE code X'78' instruction to tell VM/SP the channel/unit address.		PROLOG	
7 Issue VS WAIT SVC, specifying that the event control block will be posted when the attention interruption is received.			

Notes	Module	Label	Ref

Diagram 13-1. DMKMSS Initialization



Notes	Module	Label	Ref
1 This loop will run in the VS machine as long as MSS support is in effect. The DIAGNOSE X'78' instruction points to a buffer in DMKMSS into which VM/SP places an MSSCOM, or zeros.	DMKMSS	MAINLOOP	
2 Look at MSSCOM to determine volume serial, 3330V device address, and type of request (mount or demount). If the request is invalid, issue a message. If there are no outstanding requests, go into a wait state.		L2 RF00149 RF00122	
3 The SVC 126 routines issue orders to the MSC. If the MSC rejects the order, it sends a unit check as ending status. SVC then sets a non-zero return code in register 15.		DIAG MSSCHECK	

Notes	Module	Label	Ref

Diagram 13-2. DMKMSS Processing

Program Organization

This section describes the program organization of the DMKMSS module.

DMKMSS

The MSS communicator program.

Attributes

Reentrant

Entry Point

DMKMSS

Register Usage

R0-R9: Work registers
R10: Workarea base
R11: Program base
R12: Work register
R13: Register savearea base
R14-R15: Work registers

Directory

Figure 13-2 is an alphabetical list of the major labels in the DMKMSS program. The figure indicates the associated method of operation diagrams and it provides a brief description of the operation performed at the point in the program associated with each label.

Label	Diagram	Description
DIAG	13-2	Issues DIAGNOSE code X'78' subfunction X'08' or X'0C'.
L1	13-1	Follows pointers through the DCB, DEB, and UCB to find the communicator device address.
L2	13-2	Determines the type of MSS request (mount or demount).
MAINLOOP	13-2	Issues DIAGNOSE code X'78' subfunction X'04', requesting work.
MSSCHECK	13-2	Sets the MSC completion code for CP.
PROLOG	13-1	Initializes for DIAGNOSE code X'78' subfunction X'00'.
RF00082	13-1	Issues STIDP instruction to ensure running in a virtual machine.
RF00092	13-1	Issues message that MSS is not initialized.
RF00122	13-2	Waits for the communicator device attention interruption.
RF00149	13-2	Issues message for invalid request code in VM request.
RF00182	13-1	Issues message that DCB is not opened.
RF00190	13-1	Issues message that this must run in a virtual machine.

Figure 13-2. DMKMSS Label Directory

Data Areas

The OS/VS control blocks used (CVT, DCB, DEB, and UCB are described in *OS/VS1 System Data Areas*, Order No. SY28-0605, and in *OS/VS2 System Debugging Library: Debugging Handbook*, Order No. GC28-0632).

The MSS communicator control block (MSSCOM) is described in *VM/SP HPO Data Areas and Control Block Logic – CP*.

Diagnostic Aids

Figure 13-3 lists the messages issued by the DMKMSS program. The nearest label and the associated method of operation diagram are identified.

Label	Diagram	Message Text
RF00092	13-1	DMKMSS ENDING ERROR - MSS NOT INITIALIZED
RF00149	13-2	INVALID REQUEST CODE IN VM REQUEST
RF00182	13-1	DMKMSS ENDING ERROR - DCB FOR COMM. DEVICE NOT OPENED
RF00190	13-1	DMKMSS ENDING ERROR - DMKMSS MUST RUN IN A VIRTUAL MACH.

Figure 13-3. DMKMSS Messages

Chapter 14. DMKOVR — The Command Class Override Program

Introduction

The DMKOVR program builds an internal class-override file on a volume previously formatted by the Format/Allocate program as type OVRD.

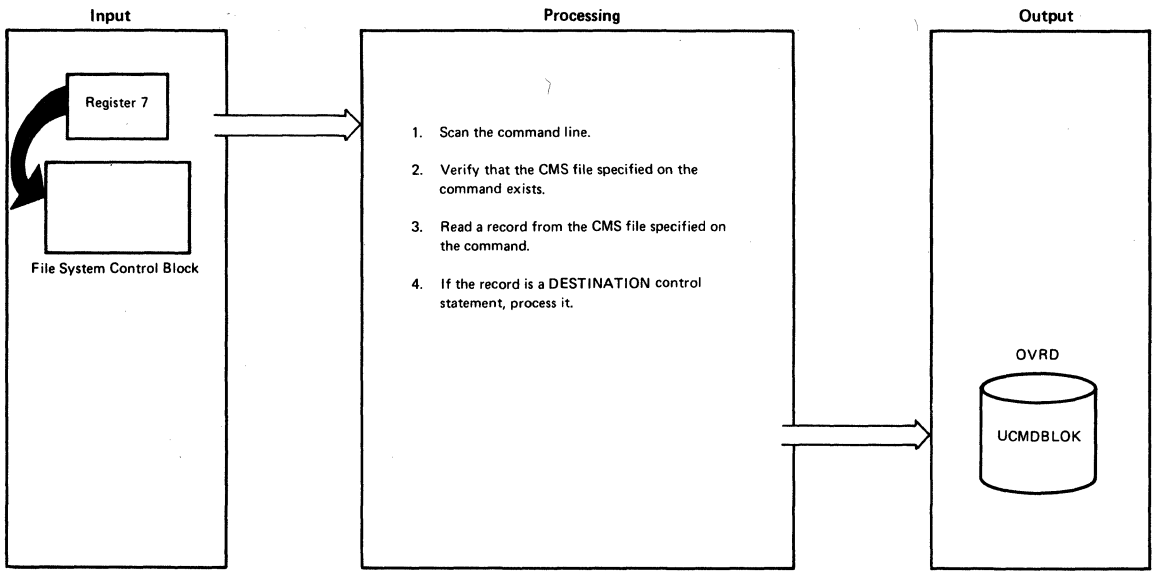
VM/SP HPO is distributed with the CP commands and DIAGNOSE codes assigned to one or more of the eight privilege classes (see the *CP Command Reference* for a list of what classes each command is assigned to). Installations can redefine the assignment of privilege classes using up to 32 classes (A through Z and 1 through 6) to tailor the authorization structure of their system.

To redefine the privilege classes for certain commands, the user creates a CMS file that contains a DESTINATION control statement and an OVERRIDE control statement for each command whose class is being changed. (See *CP for System Programming* for a detailed description of these control statements and the steps to take.) The changes to privilege classes described by these control statements are activated when the user enters an OVERRIDE command.

When the user issues the OVERRIDE command, DMKOVR receives control. DMKOVR scans the parameters specified on the OVERRIDE command, one of which is the filename of the CMS file that contains the DESTINATION and OVERRIDE control statements. Using this file, DMKOVR builds a class-override file in internal format that describes the new privilege classes for the specified commands.

Method of Operation

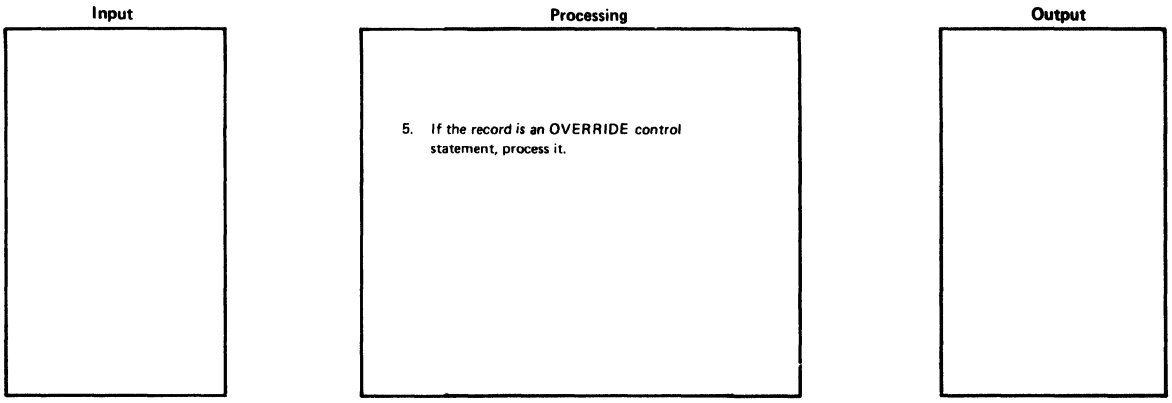
This section describes those functions that the DMKOVR program performs. There is only one method of operation diagram and that is Diagram 14-1.



Notes	Module	Label	Ref
1 Scan the OVERRIDE command. A. The user must specify the filename and filetype for the CMS override file. If not, issue the message: DMKOVR763E INVALID FILENAME OR FILE NOT FOUND Because there is no file to process, go to step 6 to clean up and return to the caller. B. If the user specified the FREE option, set an indicator (FREE) in OVRFLAG1. If the user specified the EDIT option, set an indicator (EDITMODE) in OVRFLAG1. If the user specified anything other than EDIT or FREE, issue the message: DMKOVR751E INVALID OPERAND - xxx Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file.	DMKOVR	CKINP ERR763 ERR751	
2 Issue that FSSTATE macro to determine if the CMS file specified on the OVERRIDE command exists. If the file does not exist, issue the message: DMKOVR763E INVALID FILENAME OR FILE NOT FOUND Because there is no file to process, go to step 6 to clean up and return to the caller.	DMKOVR	ERR763	

Notes	Module	Label	Ref
3 Issue the FSREAD macro to read a record from the CMS file specified on the OVERRIDE command.	DMKOVR	USERREAD	
4 If the record is a DESTINATION control statement, process it as follows. A. If the DESTINATION control statement was immediately preceded by another DESTINATION control statement, issue the message: DMKOVR751E INVALID OPERAND - DESTINAT Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file. B. Use the first three parameters on the control statement (cuu devtype volser) to help locate the override space on the CP-owned volume. The user has allocated this space with the Format/Allocate program as type OVRD. C. If the user specified the FREE option on the OVERRIDE command, set cylinder 0 record 3 on the CP-owned volume (offset 56) to blanks. This indicates that an override file does not exist. D. If the user did not specify the FREE option, put the address of the override space in cylinder 0 record 3 on the CP-owned volume. Go to step 3 to read the next record from the CMS file.	DMKOVR	DIRSTMT	

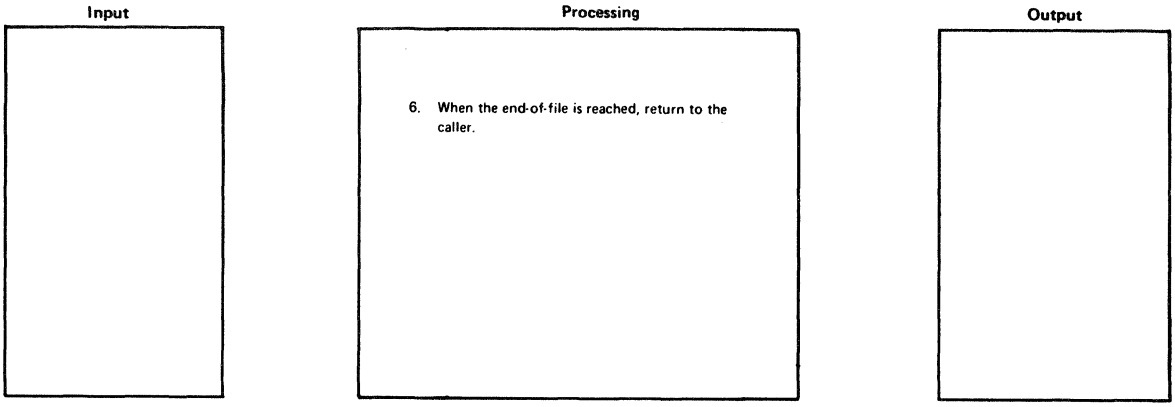
Diagram 14-1. DMKOVR - Class Override Program Processing (Part 1 of 3)



Notes	Module	Label	Ref
5 If the record is an OVERRIDE control statement, process it as follows.	DMKOVR	OVRSTMT	
A. If a DESTINATION control statement has not been read, issue the message: DMKOVR762E DESTINATION STATEMENT MISSING Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file.		ERR762	
B. Scan the control statement for the command name or DIAGNOSE code. Assume that the first nonblank character string is the command name or DIAGNOSE code. Compare this string to a table of valid commands and DIAGNOSE codes to ensure that it is valid. If it is not, issue the message: DMKOVR751E INVALID OPERAND - xxx Continue processing in edit mode. That is, scan the control statements for valid syntax, but do not build an internal override file.		SCANCOM ERR751	
C. Scan the control statement for the TYPE keyword. Ensure that TYPE is not specified for DIAGNOSE code, that there is only one character following the TYPE keyword, and that the type is valid for the specified command. If any of these errors are found, issue the message: DMKOVR751E INVALID OPERAND - xxx Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file.		SCANKEY DOTYPE ERR751	

Notes	Module	Label	Ref
If the type is valid, save it in TYPESAVE.			
D. Scan the control statement for the CLASS keyword. Verify that the classes specified are valid command classes (A trhrough Z or 1 through 6) and that no class is repeated. If either of these errors is found, issue the message: DMKOVR765E INVALID CLASS DEFINITION - x or DMKOVR766E DUPLICATE CLASS DEFINITION - x Continue processing in edit mode. That is, scan the control statements for valid syntax but do not build an internal override file. Set the class mask in CLASBITS to indicate the specified classes. If the class is an asterisk (*), all classes are allowed for this command; set the class mask in CLASBITS to indicate all classes.		DOCLASS ERR765 ERR766	
E. If the OVERRIDE control statement did not include the CLASS keyword, or if it included the TYPE keyword and there is more than one type for this command, issue the message: DMKOVR753E OPERAND MISSING Continue processing in edit mode. That is, scan the control statement for valid tyntax but do not build an internal override file.		ENDCARD ERR753	
F. Build the record to be placed in the internal override file. (If the user specified the EDIT option or if an error forced continuation in edit mode, skip this step; return to step 3 to read the next record.) Write the record in the UCMDBLOK.		OVRBUILD	

Diagram 14-1. DMKOVR - Class Override Program Processing (Part 2 of 3)



Notes	Module	Label	Ref
Go to step 3 to read the next record from the CMS file.			
6 If there are no more records in the CMS file, perform clean-up processing. If the user specified the FREE option on the OVERRIDE command or no options, write the volume label and allocation map, and issue the message: EOJ OVERRIDE FILE UPDATED If the user specified the EDIT option on the OVERRIDE command or if an error forced continuation in edit mode, issue the message: EOJ OVERRIDE FILE NOT UPDATED	DMKOVR	WRTVOL FINISH	

Notes	Module	Label	Ref

Diagram 14-1. DMKOVR - Class Override Program Processing (Part 3 of 3)

Program Organization

This section includes a program description of the DMKOVR module.

DMKOVR

Builds a command class-override file in internal format.

Entry Points

DMKOVRDE

Routines Called

None

Attributes

Not serially reusable; nonresident.

Registers at Exit

Register 15 contains a return code at exit.

Return Code	Meaning
0	Override file successfully updated.
4	Invalid operand or operand missing.
8	I/O error loading the override file.
12	Invalid option.
20	Invalid character in file-id.
24	Invalid filemode.
28	Invalid filename or file not found.
36	Disk not accessed.

where: xx is the CMS routine return code.

1xx Error in the CMS RDBU routine.

2xx Error in the CMS TYPLI routine.

Register Usage

R0: Not used.
R1: Work register.
R2: Work register from scanning routine.
R3: Work register.
R4: Work register.
R5: Work register.
R6: Work register.
R7: Work register.
R8: Work register.
R9: Pointer to device table entry.
R10: Base address for UHDRBLOK.
R11: Base register for work areas and constants.
R12: Base register for code.
R13: Base address for save area.
R14: Return address to CMS; linkage to subroutines.
R15: Entry address; on exit contains return code.

External References

None

Directory

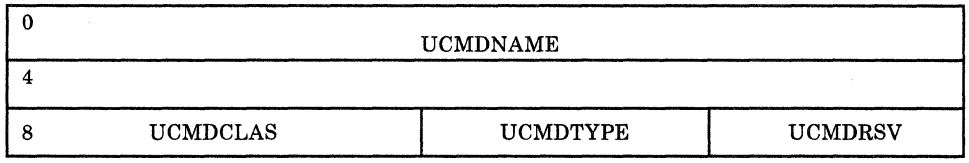
Figure 14-1 is an alphabetic list of the major labels of the class override program. The list references the associated method of operation diagram and includes a brief description of the function performed at the point in the program corresponding to each label.

Label	Diagram	Description
CHEKEOF		Receives control when there is an error reading a record from the CMS file.
CKINP	12-1	Scans the OVERRIDE command line.
DIRSTMT	12-1	Processes the DESTINATION control statement.
DMKOVRDE		Sets up base registers and initializes pointers.
DOCLASS	21-1	Processes the CLASS keyword on the OVERRIDE control statement.
DOIO		Performs DASD I/O such as writing a directory page and writing the volume label and allocation map.
DOTYPE	12-1	Processes the TYPE keyword on the OVERRIDE control statement.
ENDCARD	12-1	Verifies that all required fields were specified on the OVERRIDE control statement.
ERR751	12-1	Issues message DMKOVR751E.
ERR753	12-1	Issues message DMKOVR753E.
ERR754		Issues message DMKOVR754E.
ERR755		Issues message DMKOVR755E.
ERR760		Issues message DMKOVR760E.
ERR761		Issues message DMKOVR761E.
ERR762	12-1	Issues message DMKOVR762E.
ERR763	12-1	Issues message DMKOVR763E.
ERR764		Issues message DMKOVR764E.
ERR765	3-1	Issues message DMKOVR765E.
ERR766	3-1	Issues message DMKOVR766E.
EXIT		Returns to the caller.
FINISH	12-1	Determines whether to issue a message stating that the override file was updated or not updated.
GETPAGE		Gets the next page to use for the internal override file.
OVRBUILD	12-1	Builds the entry for the internal override file.
OVRSCAN		Scans the OVERRIDE control statement for the TYPE and CLASS keywords.
OVRSTMT	12-1	Processes the OVERRIDE control statement.
OVRVER1		Verifies that the command or DIAGNOSE code exists.
SCANCARD		Scans a record read from the CMS file, stopping at the first blank it encounters and converting alphabetic characters to uppercase.
SCANCOM	12-1	Scans the command or DIAGNOSE code specified on the OVERRIDE control statement, ensuring that it is valid.
SCANCUU		Scans a four-byte field, verifying that the first three bytes are hexadecimal characters and that the fourth byte is blank.
SCANDEV		Scans the table of devices (DEVTAB) to locate the address of the device specified on the DESTINATION control statement.
SCANKEY	12-1	Scans the OVERRIDE control statement for the TYPE and CLASS keywords and verifies that the keyword is followed by an equal sign.
UPALLOC		Scans through the allocation table, releasing the old override cylinder (if any) and locating the next available cylinder. Updates the volume label record with the pointer to the cylinder that will contain the internal override file.
USEREOF		Receives control when all records have been read from the CMS file.
USERREAD	12-1	Reads a record from the CMS file.
WRTVOL	12-1	Writes the volume label and allocation map.

Figure 14-1. The Class Override Program Label Directory

Data Areas

This section describes the UCMDBLOK DSECT that is used to map the internal class override records in the override space.



Displacement Hex	Dec	Field Name			Description
0	0	UCMDNAME	DC	CL8' '	Command or DIAGNOSE code name
8	8	UCMDCLAS	DC	XL4'00'	Class overrides
A	10	UCMDTYPE	DC	BL8'0'	Functional group type
B	11	UCMDRSV	DC	XL3'0'	Reserved

Figure 14-2. UCMDBLOK DSECT

Diagnostic Aids

Figure 14-3 lists the message issued by the class override program. The list includes the label of the message and the associate method of operation diagram.

Message Code	Label	Diagram	Message Text
DMKOVR751E	MSG751	12-1	INVALID OPERAND - operand
DMKOVR753E	MSG753	12-1	OPERAND MISSING
DMKOVR754E	MSG754		DEVICE raddr NOT OPERATIONAL
DMKOVR755E	MSG755		I/O ERROR raddr CSW csw SENSE sense
DMKOVR760E	MSG760		NOT ENOUGH SPACE ALLOCATED FOR {DIRECTORY|OVERRIDES}
DMKOVR761E	MSG761		VOLID READ IS volid1 NOT volid2 (ON raddr)
DMKOVR762E	MSG762	12-1	{DESTINATION|DIRECTORY} STATEMENT MISSING
DMKOVR763E	MSG763	12-1	INVALID FILENAME OR FILE NOT FOUND
DMKOVR764E	MSG764		ERROR IN routine
DMKOVR765E	MSG765	3-1	INVALID CLASS DEFINITION
DMKOVR766E	MSG766	3-1	DUPLICATE CLASS DEFINITION
	MSGEOK		EOJ OVERRIDE FILE UPDATED
	MSGEBAD		EOJ OVERRIDE FILE NOT UPDATED

Figure 14-3. The Class Override Program Messages

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

Z

Numerics

Virtual Machine/
System Product
High Performance Option

Service Routines
Program Logic

LY20-0898-5
File No. S370-37

READER'S COMMENT FORM

Order No. LY20-0898-5

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.

LY20-0898-5

LY20-0898-5
File No. S370-37

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

PRINTED IN U.S.A. LY20-0898-5

IBM®

Virtual Machine/
System Product
High Performance Option

Service Routines
Program Logic

LY20-0898-5
File No. S370-37

READER'S COMMENT FORM

Order No. LY20-0898-5

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.

LY20-0898-5

LY20-0898-5
File No. S370-37

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

PRINTED IN U.S.A. LY20-0898-5

IBM®

LY20-0898-5
File No. S370-37

VM/SP HPO Service Routines Program Logic File No. S370-37 Printed in U.S.A. LY20-0898-5

LY20-0898-05