File No. S370-36
Order No. SY20-0887-1

IBM Virtual Machine Facility/370: System Logic and Problem Determination Guide Volume 2

Conversational Monitor System (CMS)

Systems

| Release 6 PLC 1

This publication is intended for the IBM system hardware and software support personnel. It provides the following information for the CMS component of VM/370:

- Description of program logic
- Module descriptions and cross-references
- Abend codes

PREREQUISITE PUBLICATIONS

IBM Virtual Machine Facility/370:
Introduction, Order No. GC20-1800
Terminal User's Guide, Order No. GC20-1810
CMS Command and Macro Reference, Order No. GC20-1818
CMS User's Guide, Order No. GC20-1819

IBM

| Second Edition (March 1979)

| This is a major revision of, and obsoletes, SY20-0887-0 and Technical
| Newsletter SN25-0479. This edition applies to Release 6 PLC 1 (Program Level Change) of the IBM Virtual Machine Facility/370 and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Technical changes and additions to text and illustrations are indicated by a vertical bar to the left of the change.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 Bibliography, Order No. GC20-0001, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, VM/370 Publications, Dept. D58, Bldg. 706-2, P.O. Box 390, Poughkeepsie, New York 12602. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

This publication provides the IBM system hardware and software support personnel with the information needed to analyze problems that may occur on the IBM Virtual Machine Facility/370 (VM/370).

HOW THIS MANUAL IS ORGANIZED

This manual comprises three volumes:

"Volume 1. VM/370 Control Program (CP)," "Volume 2. Conversational Monitor System (CMS)," and "Volume 3. Remote Spooling Communications Subsystem (RSCS)" contain the logic description for each of the components. Each of these volumes is divided into four sections: Introduction, Method of Operation, Directory, and Diagnostic Aids.

The method of operation and program organization sections contain the functions and relationships of the program routines in VM/370. They indicate the program operation and organization in a general way to serve as a guide in understanding VM/370. They are not meant to be a detailed analysis of VM/370 programming and cannot be used as such.

The directories contain descriptions of all the assemble modules in CP, CMS, and RSCS. They also contain extensive cross-references between modules and labels within a VM/370 component.

The diagnostic aids sections contain additional information useful for determining the cause of a problem.

The Appendix -- which is in Volume 1 -- contains a description of VM/370 Extended Control-Program Support (ECPS).

HOW TO USE THIS MANUAL

- Isolate the component of VM/370 in which the problem occurred.
- Use the list of restrictions in _VM/370 System Messages_ to be certain that the operation that was being performed was valid.
- Use the directories and use the _VM/370 Data Areas and Control Block Logic_ to help you to isolate the problem.
- Use the method of operation and program organization sections, if necessary, to understand the operation that was being performed.

DEVICE TERMINOLOGY

The following terms in this publication refer to the indicated support devices:

- "2305" refers to IBM 2305 Fixed Head Storage, Models 1 and 2.
- "270x" refers to IBM 2701, 2702, and 2703 Transmission Control Units or the Integrated Communications Adapter (ICA) on the System/370 Model 135.
- "3330" refers to the IBM 3330 Disk Storage, Models 1, 2, or 11; the IBM 3333 Disk Storage and Control, Models 1 or 11; and the 3350 Direct Access Storage operating in 3330/3333 Model 1 or 3330/3333 Model 11 compatibility mode.
- "3340" refers to the IBM 3340 Disk Storage, Models A2, B1, and B2, and the 3344 Direct Access Storage Model B2.
- "3350" refers to the IBM 3350 Direct Access Storage Models A2 and B2 in native mode.
- "3704", "3705", or "370X" refers to IBM 3704 and 3705 Communications Controllers.
- The term "3705" refers to the 3705 I and the 3705 II unless otherwise noted.
- "2741" refers to the IBM 2741 and the 3767, unless otherwise specified.
- "3270" refers to a series of display devices, namely the IBM 3275, 3276, 3277, 3278 Display Stations. A specific device type is used only when a distinction is required between device types.

Information about display terminal usage also applies to the IBM 3036, 3138, 3148, and 3158 Display Consoles when used in display mode, unless otherwise noted.

Any information pertaining to the IBM 3284 or 3286 also pertains to the IBM 3287, 3288 and the 3289 printers, unless otherwise noted.

CMS COMPONENT

PREREQUISITE PUBLICATIONS

IBM Virtual Machine Facility/370

- Introduction, Order No. GC20-1800
- Terminal User's Guide, Order No. GC20-1810
- CMS Command and Macro Reference, Order No. GC20-1818
- CMS User's Guide, Order No. GC20-1819

COREQUISITE PUBLICATIONS

IBM Virtual Machine Facility/370

- Operator's Guide, Order No. GC20-1806
- CP Command Reference for General Users, Order No. GC20-1820
- System Programmer's Guide, Order No. GC20-1807
- System Messages, Order No. GC20-1808
- OLTSEP and Error Recording Guide, Order No. GC20-1809
- Operating Systems in a Virtual Machine, Order No. GC20-1821
- Service Routines Program Logic, Order No. SY20-0882
- Data Areas and Control Block Logic, Order No. SY20-0884

In addition, for EREP processing the following OS/VS Library publications are required:

OS/VS Environmental Recording Editing and Printing (EREP) Program, Order No. GC28-0772

OS/VS Environmental Recording Editing and Printing (EREP) Program Logic, Order No. SY28-0773

SUPPLEMENTARY PUBLICATIONS

IBM System/360 Principles of Operation, Order No. GA22-6821

IBM System/370 Principles of Operation, Order No. GA22-7000

IBM OS/VS, DOS/VS, and VM/370 Assembler Language, Order No. GC33-4010

IBM OS/VS and VM/370 Assembler Programmer's Guide, Order No. GC33-4021

RELATED PUBLICATION

IBM Virtual Machine Facility/370 Remote Spooling Communications Subsystem (RSCS) User's Guide, Order No. GC20-1816

MISCELLANEOUS INFORMATION

CMS/DOS is part of the CMS system and is not a separate system. The term CMS/DOS is used in this publication as a concise way of stating that the DOS simulation mode of CMS is currently active; that is, the CMS command

```
SET DOS ON
```

has been previously issued.

The phrase "CMS file system" refers to disk files that are in CMS's 800-byte block format; CMS's VSAM data sets are not included.

Contents

AUTOMATIC REINITIALIZATION SUPPORT

New: Program and Documentation

This support allows a CMS virtual machine to specify that control be given to a reinitialization program as an alternative to entering a disabled wait state after an abend. This information is included in the "CMS Method of Operation and Program Organization" section of this publication under "Processes IPL Line Parameters" and in the "CMS Diagnostic Aids" section of this publication under "Unrecoverable Termination."

Summary of Amendments
for SY20-0887-0
as updated by TNL SN25-0479
VM/370 Release 5 PLC 12

INDEX CORRECTION

Changed: Documentation only

The index for *VM/370 System Logic and Problem Determination Guide Volume 2 (CMS)* was in error and has been corrected.

SYSTEM LOGIC AND PROBLEM DETERMINATION GUIDE HAS BEEN REORGANIZED

Changed: Documentation only

VM/370 System Logic and Problem Determination Guide has been split into three volumes. Volume 1 contains the CP component, Volume 2 the CMS component, and Volume 3 the RSCS component.

The following material has been removed from this publication:

- "Introduction to Debugging" and "Debugging with CMS." This information can be found in _VM/370 System Programmer's Guide_.
- "Appendix A. VM/370 Coding Conventions." This information can be found in _VM/370 System Programmers Guide_.
- "Appendix B. DASD Record Formats." This information can be found in _VM/370 Service Routines Program Logic_ in the FORMAT section.
- "Appendix C. VM/370 Restrictions." This information can be found in _VM/370 Planning and System Generation Guide_ or _VM/370 System Messages_.
- "Appendix D. Applying PTFs." This information can be found in _VM/370 Planning and System Generation Guide_.

The following sections have been removed from the "CMS Diagnostic Aids" section of this publication:

- ZAP Service Program. A complete description of ZAP can be found in _VM/370 Operator's Guide_.
- DDR. A complete description of DDR can be found in _VM/370 Operator's Guide_.
- CMS Return Codes. These can be found in _VM/370 System Messages_.
- Commands for Debugging. A complete description of DEBUG can be found in _VM/370 CMS User's Guide_.

The following has been added to Volume 2:

- "Appendix A: CMS Macro Library"
- "Appendix B: CMS/DCS Macro Library"

The following topics have been removed from "CP Diagnostic Aids":

- CP Commands Used to Debug the Virtual Machine. These are contained in _VM/370 CP Command Reference for General Users_.
- CP Commands for System Programmers. These are contained in _VM/370 Operator's Guide_.

VM/370 SUPPORTS 3031, 3032, AND 3033 PROCESSORS

New: Program Feature

VM/370 provides support for the new channel-attached consoles that are part of the 3033 processors. VM/370 uses the 3033 processor model numbers in selecting model-dependent routines and setting pertinent time slices. The channels of the new processors are supported by the channel check error recovery routine.

During initialization of the machine check handler/channel check handler, error frames are read from the Service Record File (SRF) and written to the VM/370 error recording area as a new record type.

VM/370 MONITOR COMMAND ENHANCED

New: Program Feature

VM/370 monitor facilities now include, in addition to data collection on tape, spooling to disk. Operands have been added to the MONITOR command that allow:

- The automatic start and stop of data collection by defined time-fo-day values.

- The automatic start and stop of data collection by defining a high limit value.

- Specification of a userid as the recipient of the spooled monitor data.

MISCELLANEOUS

Changed: Programming and Documentation

Minor technical and editorial changes have been made in order to clarify the text.

Conversational Monitor System (CMS)

This section contains the following information:

- Introduction to CMS
- Interrupt Handling in CMS
- Functional Information
- OS Macros Under CMS
- DOS/VS Support Under CMS

Introduction To CMS

The Conversational Monitor System (CMS), the major subsystem of VM/370, provides a comprehensive set of conversational facilities to the user. Several copies of CMS may run under CP, thus providing several users with their own time sharing system. CMS is designed specifically for the VM/370 virtual machine environment.

Each copy of CMS supports a single user. This means that the storage area contains only the data pertaining to that user. Likewise, each CMS user has his own machine configuration and his own files. Debugging is simpler because the files and storage area are protected from other users.

Programs can be debugged from the terminal. The terminal is used as a printer to examine limited amounts of data. After examining program data, the terminal user can enter commands on the terminal that will alter the program. This is the most common method used to debug programs that run in CMS.

CMS, operating with the VM/370 Control Program, is a time sharing system suitable for problem solving, program development, and general work. It includes several programming language processors, file manipulation commands, utilities, and debugging aids. Additionally, CMS provides facilities to simplify the operation of other operating systems in a virtual machine environment when controlled from a remote terminal. For example, CMS capabilities are used to create and modify job streams, and to analyze virtual printer output.

Part of the CMS environment is related to the virtual machine environment created by CP. Each user is completely isolated from the activities of all other users, and each machine in which CMS executes has virtual storage available to it and managed for it. The CP commands are recognized by CMS. For example, the commands allow messages to be sent to the operator or to other users, and virtual devices to be dynamically detached from the virtual machine configuration.

The CMS Command Language

The CMS command language offers terminal users a wide range of functions. It supports a variety of programming languages, service functions, file manipulation, program execution control, and general system control. For detailed information on CMS commands, refer to the _VM/370 CMS Command and Macro Reference_.

Figure 4 describes CMS command processing.

The File System

The Conversational Monitor System interfaces with virtual disks, tapes, and unit record equipment. The CMS residence device is kept as a read-only, shared, system disk. Permanent user files may be accessed from up to nine active disks. Logical access to those virtual disks is controlled by CMS, while CP facilities manage the device sharing and virtual-to-real mapping.

User files in CMS are identified with three designators. The first is filename. The second is a filetype designator that may imply specific file characteristics to the CMS file management routines. The third is a filemode designator that describes the location and access mode of the file.

The compilers available under CMS default to particular input filetypes, such as ASSEMBLE, but the file manipulation and listing commands do not. Files of a particular filetype form a logical data library for a user; for example, the collection of all COBOL source files, or of all object (TEXT) decks, or of all EXEC procedures. This allows selective handling of specific groups of files with minimum input by the user.

User files can be created directly from the terminal with the CMS EDIT facility. EDIT provides extensive context editing services. File characteristics such as record length and format, tab locations, and serialization options can be specified. The system includes standard definitions for certain filetypes.

CMS automatically allocates compiler work files at the beginning of command execution on whichever active disk has the greatest amount of available space, and deallocates them at completion. Compiler object decks and listing files are normally allocated on the same disk as the input source file or on the primary read/write disk, and are identified by combining the input filename with the filetypes TEXT and LISTING. These disk locations may be overridden by the user.

A single user file is limited to a maximum of 65533 records and must reside on one virtual disk. The file management system limits the number of files on any one virtual disk to 3400. All CMS disk files are written as 800-byte records, chained together by a specific file entry that is stored in a table called the Master File Directory; a separate Master File Directory is kept for, and on, each virtual disk. The data records may be discontiguous, and are allocated and deallocated automatically. A subset of the Master File Directory (called the User File Directory) is made resident in virtual storage when the disk directory is made available to CMS; it is updated on the virtual disk at least once per command if the status of any file on that disk has been changed.

Virtual disks may be shared by CMS users; the facility is provided by VM/370 to all virtual machines, although a user interface is directly available in CMS commands. Specific files may be spooled between virtual machines to accomplish file transfer between users. Commands allow such file manipulations as writing from an entire disk or from a specific disk file to a tape, printer, punch, or the terminal. Other commands write from a tape or virtual card reader to disk, rename files, copy files, and erase files. Special macro libraries and text or program libraries are provided by CMS, and special commands are provided to update and use them. CMS files can be written onto and restored from unlabeled tapes via CMS commands.

<u>Caution</u>: Multiple write access under CMS can produce unpredictable results.

Problem programs which execute in CMS can create files on unlabeled tape in any record and block size; the record format can be fixed, variable, or undefined. Figure 1 describes the CMS file system.

Program Development

The Conversational Monitor System includes commands to create and compile source programs, to modify and correct source programs, to build test files, to execute test programs and to debug from the terminal. The commands of CMS are especially useful for OS and DOS/VS program development, but also may be used in combination with other operating systems to provide a virtual machine program development tool.

CMS utilizes the OS and DOS/VS compilers via interface modules; the compilers themselves normally are not changed. In order to provide suitable interfaces, CMS includes a certain degree of OS and DOS/VS simulation. The sequential, direct, and partitioned access methods are logically simulated; the data records are physically kept in the chained 800-byte blocks that are standard to CMS, and are processed internally to simulate OS data set characteristics. CMS supports VSAM catalogs, data spaces, and files on OS and DOS disks using the DOS/VS Access Method Services. OS Supervisor Call functions such as GETMAIN/FREEMAIN and TIME are simulated. The simulation restrictions concerning what types of OS object programs can be executed under CMS are primarily related to the OS/PCP, MFT, and MVT Indexed Sequential Access Method (ISAM) and the telecommunications access methods, while functions related to multitasking in OS and DOS/VS are ignored by CMS. For more information, see "OS Macro Simulation under CMS" and "DOS/VS Support under CMS."

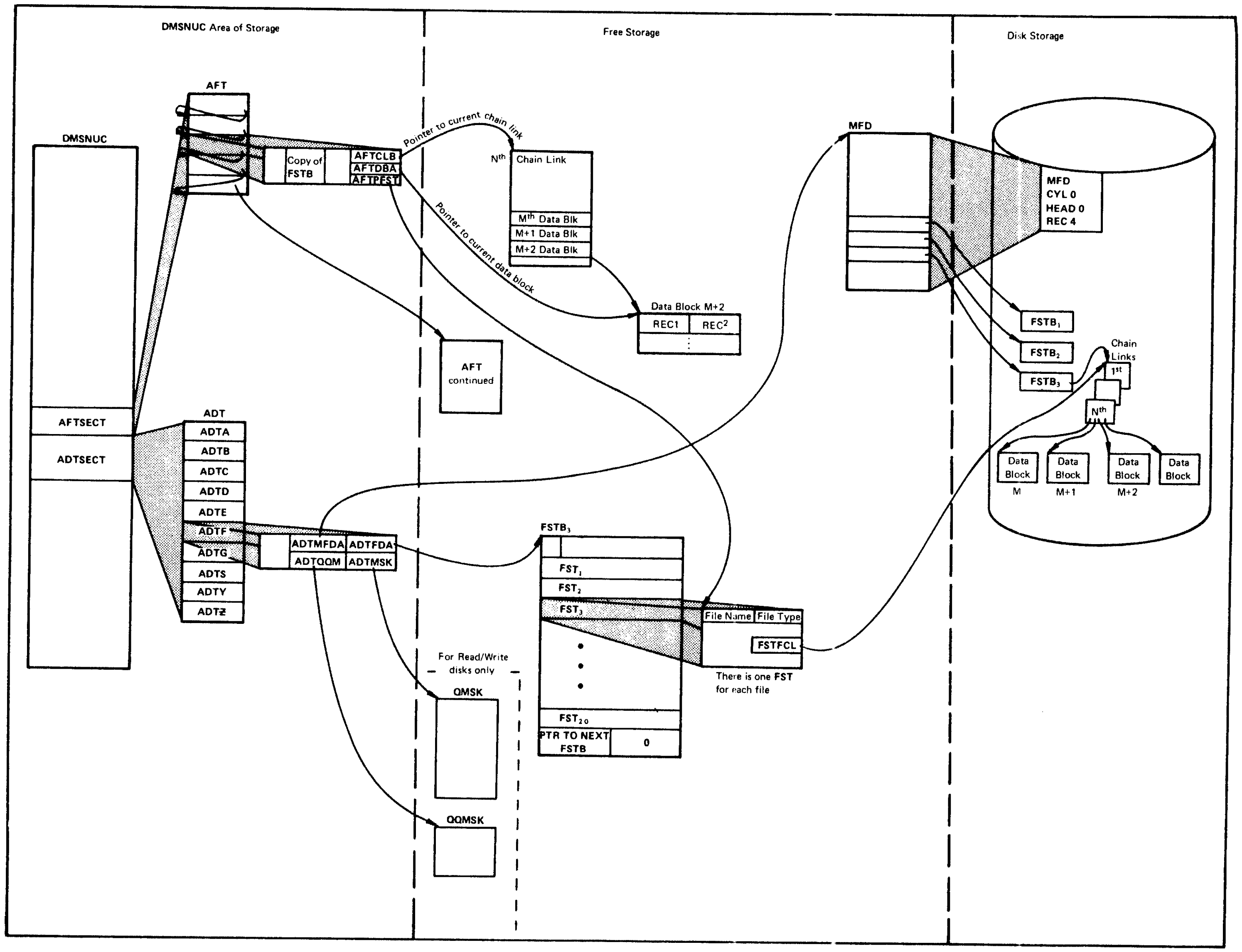


Figure 1. CMS File System

Interrupt Handling In CMS

CMS receives virtual SVC, input/output, program, machine, and external interruptions and passes control to the appropriate handling program.

SVC Interruptions

The Conversational Monitor System is SVC (supervisor call) driven. SVC interruptions are handled by the DMSITS resident routines. Two types of SVCs are processed by DMSITS: internal linkage SVC 202 and 203, and any other SVCs. The internal linkage SVC is issued by the command and function programs of the system when they require the services of other CMS programs. (Commands entered by the user from the terminal are converted to the internal linkage SVC by DMSINT). The OS SVCs are issued by the processing programs (for example, the Assembler).

INTERNAL LINKAGE SVCS

When DMSITS receives control as a result of an internal linkage SVC (202 or 203), it saves the contents of the general registers, floating-point registers, and the SVC old PSW, establishes the normal and error return addresses, and passes control to the specified routine. (The routine is specified by the first 8 bytes of the parameter list whose address is passed in register 1 for SVC 202, or by a halfword code following SVC 203.)

For SVC 202, if the called program is not found in the internal function table of nucleus (resident) routines, then DMSITS attempts to call in a module (a CMS file with filetype MODULE) of this name via the LOADMOD command.

If the program was not found in the function table, nor was a module successfully loaded, DMSITS returns an error code to the caller.

To return from the called program, DMSITS restores the calling program's registers, and makes the appropriate normal or error return as defined by the calling program.

OTHER SVCs

The general approach taken by DMSITS to process other SVCs supported under CMS is essentially the same as that taken for the internal linkage SVCs. However, rather than passing control to a command or function program, as is the case with the internal linkage SVC, DMSITS passes control to the appropriate routine. The SVC number determines the appropriate routine.

In handling non-CMS SVC calls, DMSITS refers first to a user-defined SVC table (if one has been set up by the DMSHDS program). If the user-defined SVC table is present, any SVC number (other than 202 or 203) is looked for in that table. If it is found, control is transferred to the routine at the specified address.

If the SVC number is not found in the user-defined SVC table (or if the table is nonexistent), DMSITS either transfers control to the CMSDOS shared segment (if SETDOS ON has been issued), or the standard system table (contained in DMSSVT) of OS calls is searched for that SVC number. If the SVC number is found, control is transferred to the corresponding address in the usual manner. If the SVC is not in either table, then the supervisor call is treated as an abend call.

The DMSHDS initialization program sets up the user-defined SVC table. It is possible for a user to provide his own SVC routines.

Input/Output Interruptions

All input/output interruptions are received by the I/O interrupt handler, DMSITI. DMSITI saves the I/O old PSW and the CSW (channel status word). It then determines the status and requirements of the device causing the interruption and passes control to the routine that processes interruptions from that device. DMSITI scans the entries in the device table until it finds the one containing the device address that is the same as that of the interrupting device. The device table (DEVTAB) contains an entry for each device in the system. Each entry for a particular device contains, among other things, the address of the program that processes interruptions from that device.

When the appropriate interrupt handling routine completes its processing, it returns control to DMSITI. At this point, DMSITI tests the wait bit in the saved I/O old PSW. If this bit is off, the interruption was probably caused by a terminal (asynchronous) I/O operation. DMSITI then returns control to the interrupted program by loading the I/O old PSW.

If the wait bit is on, the interruption was probably caused by a nonterminal (synchronous) I/O operation. The program that initiated the operation most likely called the DMSIOW function routine to wait for a particular type of interruption (usually a device end). In this case, DMSITI checks the pseudo-wait bit in the device table entry for the interrupting device. If this bit is off, the system is waiting for some event other than the interruption from the interrupting device; DMSITI returns to the wait state by loading the saved I/O old PSW. (This PSW has the wait bit on.)

If the pseudo-wait bit is on, the system is waiting for an interruption from that particular device. If this interruption is not the one being waited for, DMSITI loads the saved I/O old PSW. This will again place the machine in the wait state. Thus, the program that is waiting for a particular interruption will be kept waiting until that interruption occurs.

If the interruption is the one being waited for, DMSITI resets both the pseudo-wait bit in the device table entry and the wait bit in the I/O old PSW. It then loads that PSW. This causes control to be returned to the DMSIOW function routine, which, in turn, returns control to the program that called it to wait for the interruption.

Terminal Interruptions

Terminal input/output interruptions are handled by the DMSCIT module. All interruptions other than those containing device end, channel end, attention, or unit exception status are ignored. If device end status is present with attention and a write CCW was terminated, its buffer is unstacked. An attention interrupt causes a read to be issued to the terminal, unless attention exits have been queued via the STAX macro. The attention exit with the highest priority is given control at each attention until the queue is exhausted, then a read is issued. Device end status indicates that the last I/O operation has been completed. If the last I/O operation was a write, the line is deleted from the output buffer and the next write, if any, is started. If the last I/O operation was a normal read, the buffer is put on the finished read list and the next operation is started. If the read was caused by an attention interrupt, the line is first checked for the commands RT, HO, HT, or HX, and the appropriate flags are set if one is found. Unit exception indicates a canceled read. The read is reissued, unless it had been issued with ATTREST=NO, in which case unit exception is treated as device end.

Reader/Punch/Printer Interruptions

Interruptions from these devices are handled by the routines that actually issue the corresponding I/O operations. When an interruption from any of these devices occurs, control passes to DMSITI. Then DMSITI passes control to DMSIOW, which returns control to the routine that issued the I/O operation. This routine can then analyze the cause of the interruption.

User-Controlled Device Interruptions

Interrupts from devices under user control are serviced the same as CMS devices except that DMSIOW and DMSITI manipulate a user-created device table, and DMSITI passes control to any user-written interrupt processing routine that is specified in the user device table. Otherwise, the processing program regains control directly.

Program Interruptions

The program interruption handler, DMSITP, receives control when a program interruption occurs. When DMSITP gets control, it stores the program old PSW and the contents of the registers 14, 15, 0, 1, and 2 into the program interruption element (PIE). (The routine that handles the SPIE macro instruction has already placed the address of the program interruption control area (PICA) into PIE.) DMSITP then determines whether or not the event that caused the interruption was one of those selected by a SPIE macro instruction. If it was not, DMSITP passes control to the DMSABN abend recovery routine.

If the cause of the interruption was one of those selected in a SPIE macro instruction, DMSITP picks up the exit routine address from the PICA and passes control to the exit routine. Upon return from the exit routine, DMSITP returns to the interrupted program by loading the original program check old PSW. The address field of the PSW was modified by a SPIE exit routine in the PIE.

External Interruptions

An external interruption causes control to be passed to the external interrupt handler DMSITE. If the user has issued the HNDEXT macro to trap external interrupts, DMSITE passes control to the user's exit routine. If the interrupt was caused by the timer, DMSITE resets the timer and types the BLIP character at the terminal. The standard BLIP timer setting is two seconds, and the standard BLIP character is uppercase, followed by the lowercase (it moves the typeball without printing). Otherwise, control is passed to the DEBUG routine.

Machine Check Interruptions

Hard machine check interruptions on the real processor are not reflected to a CMS virtual user by CP. A message prints on the console indicating the failure. The user is then disabled and must IPL CMS again in order to continue.

Functional Information

The most important thing to remember about CMS, from a debugging standpoint, is that it is a one-user system. The supervisor manages only one user and keeps track of only one user's file and storage chains. Thus, everything in a dump of a particular machine relates only to that virtual machine's activity.

You should be familiar with register usage, save area structuring, and control block relationships before attempting to debug or alter CMS.

Register Usage

When a CMS routine is called, R1 must point to a valid parameter list (PLIST) for that program. On return, R0 may or may not contain meaningful information (for example, on return from a call to FILEDEF with no change, R0 will contain a negative address if a new FCB has been set up; otherwise, a positive address of the already existing FCB). R15 will contain the return code, if any. The use of Registers 0 and 2 through 11 varies.

On entry to a command or routine called by SVC 202 the following are in effect:

Register	Contents
1	The address of the PLIST supplied by the caller.
12	The address entry point of the called routine.
13	The address of a work area (12 doublewords) supplied by SVCINT.
14	The return address to the SVCINT routine.
15	The entry point (same as register 12).

On return from a routine, Register 15 contains:

Return Code	Meaning
0	No error occurred
<0	Called routine not found
>0	Error occurred

If a CMS routine is called by an SVC 202, registers 0 through 14 are saved and restored by CMS.

Most CMS routines use register 12 as a base register.

Structure of DMSNUC

DMSNUC is the portion of storage in a CMS virtual machine that contains system control blocks, flags, constants, and pointers.

The CSECTs in DMSNUC contain only symbolic references. This means that an update or modification to CMS, which changes a CSECT in DMSNUC, does not automatically force all CMS modules to be recompiled. Only those modules that refer to the area that was redefined must be recompiled.

USERSECT (USER AREA)

The USERSECT CSECT defines space that is not used by CMS. A modification or update to CMS can use the 18 fullwords defined for USERSECT. There is a pointer (AUSER) in the NUCON area to the user space.

DEVTAB (DEVICE TABLE)

The DEVTAB CSECT is a table describing the devices available for the CMS system. The table contains the following entries:

- 1 console
- 10 disks
- 1 reader
- 1 punch
- 1 printer
- 4 tapes

You can change some existing entries in DEVTAB. Each device table entry contains the following information:

- Virtual device address
- Device flags
- Device types
- Symbol device name
- Address of the interrupt processing routine (for the console)

The virtual address of the console is defined at IPL time. The virtual address of the user disks can be altered dynamically with the ACCESS command. The virtual address of the tapes can be altered in the device table. Changing the virtual address of the reader, printer, or punch will have no effect. Figure 2 describes the devices supported by CMS.

Structure of CMS Storage

Figure 3 describes how CMS uses its virtual storage. The pointers indicated (MAINSTRT, MAINHIGH, FREELOWE, and FREEUPPR) are all found in NUCON (the nucleus constant area).

The sections of CMS storage have the following uses:

- _DMSNUC_ (_X'00000' to approximately X'03000'_). This area contains pointers, flags, and other data updated by the various system routines.

- _Low-Storage DMSFREE Free Storage Area_ (_Approximately X'03000' to X'0E000'_). This area is a free storage area, from which requests from DMSFREE are allocated. The top part of this area contains the file directory for the System Disk (SSTAT). If there is enough room (as there will be in most cases), the FREETAB table also occupies this area, just below the SSTAT.

Virtual IBM Device	Virtual Address[1]	Symbolic Name	Device Type
3210, 3215, 1052, 3066, 3270	ccu	CON1	System console
2314, 3330, 3340 3350	190	DSK0	System disk (read-only)
2314, 3330, 3340 3350	191[2]	DSK1	Primary disk (user files)
2314, 2319, 3330, 3340, 3350	ccu	DSK2	Disk (user files)
2314, 2319, 3330, 3340, 3350	ccu	DSK3	Disk (user files)
2314, 2319, 3330, 3340, 3350	192	DSK4	Disk (user files)
2314, 2319, 3330, 3340, 3350	ccu	DSK5	Disk (user files)
2314, 2319, 3330, 3340, 3350	ccu	DSK6	Disk (user files)
2314, 2319, 3330, 3340, 3350	ccu	DSK7	Disk (user files)
2314, 2319, 3330, 3340, 3350	19E	DSK8	Disk (user files)
2314, 2319, 3330, 3340, 3350	ccu	DSK9	Disk (user files)
1403, 3203, 3211 1443	00E	PRN1	Line printer
2540, 2501, 3505	00C	RDR1	Card reader
2540, 3525	00D	PCH1	Card punch
2415, 2420, 3410, 3420	181-4	TAP1-TAP4	Tape drives

[1]The device addresses shown are those that are preassembled into the CMS resident device table. These need only be modified and a new device table made resident to change the addresses.
[2]The virtual device address (ccu) of a disk for user files can be any valid System/370 device address, and can be specified by the CMS user when he activates a disk. If the user does not activate a disk immediately after loading CMS, CMS automatically activates the primary disk at virtual address 191.

Figure 2. Devices Supported by a CMS Virtual Machine

- Transient Program Area (X'0E000' to X'10000'). Since it is not essential to keep all nucleus functions resident in storage all the time, some of them are made "transient." This means that when they are needed, they are loaded from the disk into the transient program area. Such programs may not be longer than two pages, because that is the size of the transient area. (A page is 4096 bytes of virtual storage.) All transient routines must be serially reusable since they are not read in each time they are needed.

- CMS Nucleus (X'10000' to X'20000'). Segment 1 of storage contains the reentrant code for the CMS Nucleus routines. In shared CMS systems, this is the "protected segment," which must consist only of reentrant code, and may not be modified under any circumstances. Thus, such functions as DEBUG breakpoints or CP address stops cannot be placed in Segment 1 when it is a protected segment in a saved system.

- User Program Area (X'20000' to Loader Tables). User programs are loaded into this area by the LOAD command. Storage allocated by means of the GETMAIN macro instruction is taken from this area, starting from the high address of the user program. In addition, this storage area can be allocated from the top down by DMSFREE, if there is not enough storage available in the low DMSFREE storage area. Thus, the usable size of the user program area is reduced by the amount of free storage that has been allocated from it by DMSFREE.

- Loader Tables (Top pages of storage). The top of storage is occupied by the loader tables, which are required by the CMS loader. These tables indicate which modules are currently loaded in the user program area (and the transient program area after a LOAD command). The size of the loader tables can be varied by the SET LDRTBLS command. However, to successfully change the size of the loader tables, the SET LDRTBLS command must be issued immediately after IPL.

Free Storage Management

Free storage can be allocated by issuing the GETMAIN or DMSFREE macros. Storage allocated by the GETMAIN macro is taken from the user program area, beginning after the high address of the user program.

Storage allocated by the DMSFREE macro can be taken from several areas.

If possible, DMSFREE requests are allocated from the low address free storage area. Otherwise, DMSFREE requests are satisfied from the storage above the user program area.

There are two types of DMSFREE requests for free storage: requests for USER storage and NUCLEUS storage. Because these two types of storage are kept in separate 4K pages, it is possible for storage of one type to be available in low storage, while no storage of the other type is available.

GETMAIN FREE STORAGE MANAGEMENT

All GETMAIN storage is allocated in the user program area, starting after the end of the user's actual program. Allocation begins at the location pointed to by the NUCON pointer MAINSTRT. The location MAINHIGH in NUCON is the "high extend" pointer for GETMAIN storage.

Before issuing any GETMAIN macros, user programs must use the STRINIT macro to set up user free storage pointers. The STRINIT macro is issued only once, preceding the initial GETMAIN request. The format of the STRINIT macro is:

```
┌──────────────────────────────────────────────────────────────────┐
|          |         | ┌         ┌    ┐┐                           |
| [label]  | STRINIT | |TYPCALL=|SVC  ||                           |
|          |         | |         |BALR||                           |
|          |         | └         └    ┘┘                           |
└──────────────────────────────────────────────────────────────────┘
```

where:

```
         ┌    ┐
TYPCALL= │SVC │
         │BALR│
         └    ┘
```

indicates how control is passed to DMSSTG, the routine that processes the STRINIT macro. Since DMSSTG is a nucleus-resident routine, other nucleus-resident routines can branch directly to it (TYPCALL=BALR) while routines that are not nucleus-resident must use linkage SVC (TYPCALL=SVC). If no operands are specified, the default is TYPCALL=SVC.

When the STRINIT macro is executed, both MAINSTRT and MAINHIGH are initialized to the end of the user's program, in the user program area. As storage is allocated from the user program area to satisfy GETMAIN requests, the MAINHIGH pointer is adjusted upward. Such adjustments are always in multiples of doublewords, so that this pointer is always on a doubleword boundary. As the allocated storage is released, the MAINHIGH pointer is adjusted downward.

The pointer MAINHIGH can never be higher than FREELOWE, the "low extend" pointer for DMSFREE storage allocated in the user program area. If a GETMAIN request cannot be satisfied without extending MAINHIGH above FREELOWE, then GETMAIN will take an error exit, indicating that insufficient storage is available to satisfy the request.

The area between MAINSTRT and MAINHIGH may contain blocks of storage that are not allocated and that are, therefore, available for allocation by a GETMAIN instruction. These blocks are chained together, with the first one pointed to by the NUCON location MAINSTRT. Refer to Figure 3 for a description of CMS virtual storage usage.

The format of an element on the GETMAIN free element chain is as follows:

```
      ┌────────┬────────┬────────┬────────┐
      │ FREPTR -- pointer to next free    │
 0(0) │     element in the chain, or 0    │
      │     if there is no next element   │
      ├────────┼────────┼────────┼────────┤
      │ FRELEN -- length, in bytes, of    │
 4(4) │    this element                   │
      │                                   │
      ├────────┼────────┼────────┼────────┤
      │  Remainder of this free element   │
      .                                   .
      .                                   .
      .                                   .
```

When issuing a variable-length GETMAIN, two and one-half pages are reserved for CMS usage; this is a design value. A user who needs additional reserved pages (for example, for larger directories) should free up some of the variable GETMAIN storage from the high end.

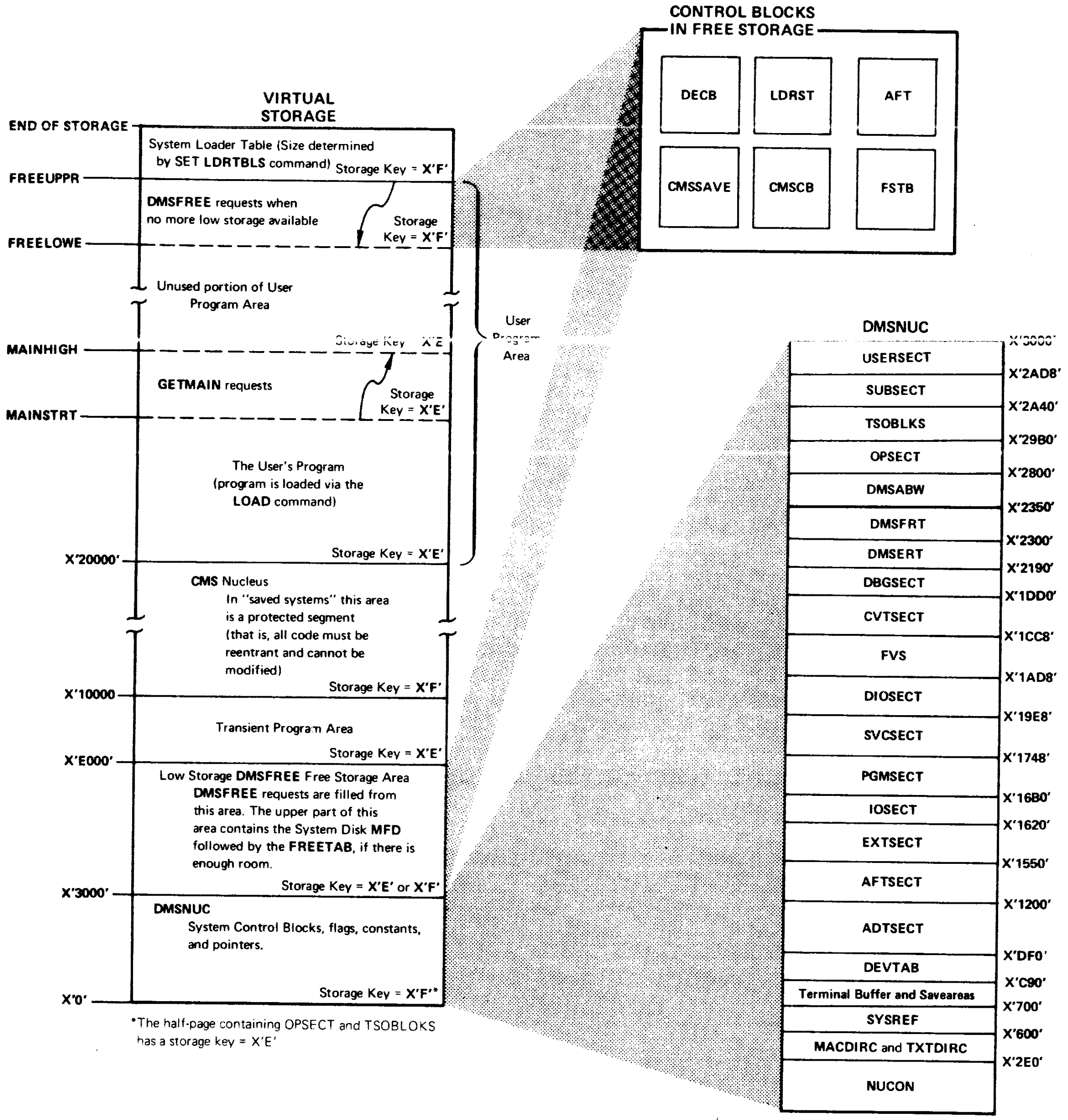

CONTROL BLOCKS IN FREE STORAGE
DECB
LDRST
AFT
CMSSAVE
CMSCB
FSTB
VIRTUAL STORAGE
END OF STORAGE
System Loader Table (Size determined by SET LDRTBLS command) Storage Key = X'F'
FREEUPPR
DMSFREE requests when no more low storage available
Storage Key = X'F'
FREELOWE
Unused portion of User Program Area
MAINHIGH
Storage Key = X'E'
GETMAIN requests
Storage Key = X'E'
MAINSTRT
The User's Program (program is loaded via the LOAD command)
User Program Area
X'20000'
Storage Key = X'E'
CMS Nucleus
In "saved systems" this area is a protected segment (that is, all code must be reentrant and cannot be modified)
Storage Key = X'F'
X'10000'
Transient Program Area
Storage Key = X'E'
X'E000'
Low Storage DMSFREE Free Storage Area DMSFREE requests are filled from this area. The upper part of this area contains the System Disk MFD followed by the FREETAB, if there is enough room.
Storage Key = X'E' or X'F'
X'3000'
DMSNUC
System Control Blocks, flags, constants, and pointers.
Storage Key = X'F'*
X'0'
*The half-page containing OPSECT and TSOBLOKS has a storage key = X'E'
DMSNUC
X'3000'
USERSECT
X'2AD8'
SUBSECT
X'2A40'
TSOBLKS
X'29B0'
OPSECT
X'2800'
DMSABW
X'2350'
DMSFRT
X'2300'
DMSERT
X'2190'
DBGSECT
X'1DD0'
CVTSECT
X'1CC8'
FVS
X'1AD8'
DIOSECT
X'19E8'
SVCSECT
X'1748'
PGMSECT
X'16B0'
IOSECT
X'1620'
EXTSECT
X'1550'
AFTSECT
X'1200'
ADTSECT
X'DF0'
DEVTAB
X'C90'
Terminal Buffer and Saveareas
X'700'
SYSREF
X'600'
MACDIRC and TXTDIRC
X'2E0'
NUCON

Figure 3. CMS Storage Map

DMSFREE FREE STORAGE MANAGEMENT

The DMSFREE macro allocates CMS free storage. The format of the DMSFREE macro is:

```
┌──────────────────────────────────────────────────────────────────────────────┐
│         │         │               ┌          ┐                               │
│ [label] │ DMSFREE │ DWORDS={ n  } │,MIN={ n  }│                              │
│         │         │        { (0)} │      { (1)}│                             │
│         │         │               └          ┘                               │
│         │         │ ┌       ┌        ┐┐ ┌     ┌     ┐┐                        │
│         │         │ │,TYPE=│USER    ││ │,ERR=│laddr││                        │
│         │         │ │       │NUCLEUS││ │     │  *  ││                        │
│         │         │ └       └        ┘┘ └     └     ┘┘                        │
│         │         │ ┌       ┌    ┐┐ ┌          ┌    ┐┐                        │
│         │         │ │,AREA=│LOW ││ │,TYPCALL=│SVC ││                         │
│         │         │ │       │HIGH││ │          │BALR││                        │
│         │         │ └       └    ┘┘ └          └    ┘┘                        │
└──────────────────────────────────────────────────────────────────────────────┘
```

where:

label
  is any valid assembler language label.

DWORDS={ n }
       { (0) }
  is the number of doublewords of free storage requested. DWORDS=n specifies the number of doublewords directly and DWORDS=(0) indicates that register 0 contains the number of doublewords requested.

MIN={ n }
    { (1) }
  indicates a variable request for free storage. If the exact number of doublewords indicated by the DWORDS operand is not available, then the largest block of storage that is greater than or equal to the minimum is returned. MIN=n specifies the minimum number of doublewords of free storage directly while MIN=(1) indicates that the minimum is in register 1. The actual amount of free storage allocated is returned to the requestor via general register 0.

```
     ┌       ┐
TYPE=│USER   │
     │NUCLEUS│
     └       ┘
```
  indicates the type of CMS storage with which this request for free storage is filled: USER or NUCLEUS.

```
    ┌     ┐
ERR=│laddr│
    │  *  │
    └     ┘
```
  is the return address if any error occurs. "laddr" is any address that can be referred to in an LA (load address) instruction. The error return is taken if there is a macro coding error or if there is not enough free storage available to fill the request. If the asterisk (*) is specified for the return address, the error return is the same as a normal return. There is no default for this operand. If it is omitted and an error occurs, the system will abend.

```
        ┌    ┐
AREA=│LOW │
        │HIGH│
        └    ┘
```

indicates the area of CMS free storage from which this request for free storage is filled. LOW indicates the low storage area between DMSNUC and the transient program area. HIGH indicates the area of storage between the user program area and the CMS loader tables. If AREA is not specified, storage is allocated wherever it is available.

```
           ┌    ┐
TYPCALL=│SVC │
           │BALR│
           └    ┘
```

indicates how control is passed to DMSFREE. Since DMSFREE is a nucleus-resident routine, other nucleus-resident routines can branch directly to it (TYPCALL=BALR) while routines that are not nucleus-resident must use linkage SVC (TYPCALL=SVC).

The pointers FREEUPPR and FREELOWE in NUCON indicate the amount of storage that DMSFREE has allocated from the high portion of the user program area. These pointers are initialized to the beginning of the loader tables.

The pointer FREELOWE is the "low extend" pointer of DMSFREE storage in the user program area. As storage is allocated from the user program area to satisfy DMSFREE requests, this pointer will be adjusted downward. Such adjustments are always in multiples of 4K bytes, so that this pointer is always on a 4K boundary. As the allocated storage is released, this pointer is adjusted upward.

The pointer FREELOWE can never be lower than MAINHIGH, the "high extend" pointer for GETMAIN storage. If a DMSFREE request cannot be satisfied without extending FREELOWE below MAINHIGH, then DMSFREE will take an error exit, indicating that storage is insufficient to satisfy the request. Figure 3 shows the relationship of these storage areas.

The FREETAB free storage table is kept in free storage, usually in low storage, just below the Master File Directory for the System Disk (S-disk). However, the FREETAB may be located at the top of the user program area. This table contains one byte for each page of virtual storage. Each such byte contains a code indicating the use of that page of virtual storage. The codes in this table are as follows:

<u>Code</u>	<u>Meaning</u>
USERCODE (X'01')	The page is assigned to user storage.
NUCCODE (X'02')	The page is assigned to nucleus storage.
TRNCODE (X'03')	The page is part of the transient program area.
USARCODE (X'04')	The page is part of the user program area.
SYSCODE (X'05')	The page is none of the above. The page is assigned to system storage, system code, or the loader tables.

Other DMSFREE storage pointers are maintained in the DMSFRT CSECT, in NUCON. The four chain header blocks are the most important fields in DMSFRT. The four chains of unallocated elements are:

- The low storage nucleus chain
- The low storage user chain
- The high storage nucleus chain
- The high storage user chain

For each of these chains of unallocated elements, there is a control block consisting of four words, with the following format:

0 (0)	POINTER -- pointer to the first free element on the chain, or zero, if the chain is empty.			
4 (4)	NUM -- the number of elements on the chain.			
8 (8)	MAX -- a value equal to or greater than the size of the largest element.			
12 (C)	FLAGS- Flag byte	SKEY - Storage key	TCODE - FREETAB code	Unused

<u>where</u>:

POINTER points to the first element on this chain of free elements. If there are no elements on this free chain, then the POINTER field contains all zeros.

NUM contains the number of elements on this chain of free elements. If there are no elements on this free chain, then this field contains all zeros.

MAX is used to avoid searches that will fail. It contains a number not exceeding the size, in bytes, of the largest element on the free chain. Thus, a search for an element of a given size will not be made if that size exceeds the MAX field. However, this number may actually be larger than the size of the largest free element on the chain.

FLAGS The following flags are used:

FLCLN (X'80') -- Clean-up flag. This flag is set if the chain must be updated. This will be necessary in the following circumstances:

- If one of the two high storage chains contains a 4K page to which FREELOWE points, then that page can be removed from the chain, and FREELOWE can be increased.

- All completely unallocated 4K pages are kept on the user chain, by convention. Thus, if one of the nucleus chains (low storage or high storage) contains a full page, then this page must be transferred to the corresponding user chain.

FLCLB (X'40') -- Destroyed flag. Set if the chain has been destroyed.

FLHC (X'20') -- High storage chain. Set for both the nucleus and user high-storage chains.

FLNU (X'10') -- Nucleus chain. Set for both the low storage and high storage nucleus chains.

FLPA (X'08') -- Page available. This flag is set if there is a full 4K page available on the chain. This flag may be set even if there is no such page available.

SKEY	contains the one-byte storage key assigned to storage on this chain.
TCODE	contains the one-byte FREETAB table code for storage on this chain.

Allocating User Free Storage

When DMSFREE with TYPE=USER (the default) is called, one or more of the following steps are taken in an attempt to satisfy the request. As soon as one of the following steps succeeds, then user free storage allocation processing terminates.

1. Search the low storage user chain for a block of the required size.
2. Search the high storage user chain for a block of the required size.
3. Extend high storage user storage downward into the user program area, modifying FREELOWE in the process.
4. For a variable request, put all available storage in the user program area onto the high storage user chain, and then allocate the largest block available on either the high storage user chain or the low storage user chain. The allocated block will not be satisfactory unless it is larger than the minimum requested size.

Allocating Nucleus Free Storage

When DMSFREE with TYPE=NUCLEUS is called, the following steps are taken in an attempt to satisfy the request, until one succeeds:

1. Search the low storage nucleus chain for a block of the required size.
2. Get free pages from the low storage user chain, if any are available, and put them on the low storage nucleus chain.
3. Search the high storage nucleus chain for a block of the required size.
4. Get free pages from the high storage user chain, if they are available, and put them on the high storage nucleus chain.
5. Extend high storage nucleus storage downward into the User Program Area, modifying FREELOWE in the process.
6. For variable requests, put all available pages from the user chains and the user program area onto the nucleus chains, and allocate the largest block available on either the low storage nucleus chains, or the high storage nucleus chains.

Releasing Storage

The DMSFRET macro releases free storage previously allocated with the DMSFREE macro. The format of the DMSFRET macro is:

```
┌──────────────────────────────────────────────────────────────────────────────┐
│ [label] │ DMSFRET │ DWORDS=⎧ n  ⎫,LOC=⎧laddr⎫                                 │
│         │         │        ⎩(0) ⎭     ⎩ (1) ⎭                                 │
│         │         │ ┌      ┌     ┐┐ ┌             ┌    ┐┐                     │
│         │         │ │,ERR=│laddr││ │,TYPCALL=│SVC ││                          │
│         │         │ │      │  *  ││ │             │BALR││                     │
│         │         │ └      └     ┘┘ └             └    ┘┘                     │
└──────────────────────────────────────────────────────────────────────────────┘
```

where:

label	is any valid Assembler language label.
DWORDS={ n | (0) }	is the number of doublewords of storage to be released. DWORDS=n specifies the number of doublewords directly and DWORDS=(0) indicates that register 0 contains the number of doublewords being released.
LOC={ laddr | (1) }	is the address of the block of storage being released. "laddr" is any address that can be referred to in an LA (load address) instruction. LOC=laddr specifies the address directly while LOC=(1) indicates the address is in register 1.
ERR=[laddr | *]	is the return address if an error occurs. "laddr" is any address that can be referred to by an LA (load address) instruction. The error return is taken if there is a macro coding error or if there is a problem returning the storage. If an asterisk (*) is specified, the error return address is the same as the normal return address. There is no default for this operand. If it is omitted and an error occurs, the system will abend.
TYPCALL=[<u>SVC</u> | BALR]	indicates how control is passed to DMSFRET. Since DMSFRET is a nucleus-resident routine, other nucleus-resident routines can branch directly to it (TYPCALL=BALR) while routines that are not nucleus-resident must use SVC linkage (TYPCALL=SVC).

When DMSFRET is called, the block being released is placed on the appropriate chain. At that point, the final update operation is performed, if necessary, to advance FREELOWE, or to move pages from the nucleus chain to the corresponding user chain.

Similar update operations will be performed, when necessary, after calls to DMSFREE, as well.

RELEASING ALLOCATED STORAGE

Storage allocated by the GETMAIN macro instruction may be released in any of the following ways:

1. A specific block of such storage may be released by means of the FREEMAIN macro instruction.

2. The STRINIT macro instruction releases all storage allocated by any previous GETMAIN requests.

3. Almost all CMS commands issue a STRINIT macro instruction. Thus, executing almost any CMS command will cause all GETMAIN storage to be released.

Storage allocated by the DMSFREE macro instruction may be released in any of the following ways:

1. A specific block of such storage may be released by means of the DMSFRET macro instruction.

2. Whenever any user routine or CMS command abnormally terminates (so that the routine DMSABN is entered), and the abend recovery facility of the system is invoked, all DMSFREE storage with TYPE=USER is released automatically.

Except in the case of abend recovery, storage allocated by the DMSFREE macro is never released automatically by the system. Thus, storage allocated by means of this macro instruction should always be released explicitly by means of the DMSFRET macro instruction.

DMSFREE SERVICE ROUTINES

The DMSFRES macro instruction is used by the system to request certain free storage management services.

The format of the DMSFRES macro is:

```
┌──────────────────────────────────────────────────────────────────┐
│ [label] │ DMSFRES │ INIT1  ┌           ┌    ┐┐                   │
│         │         │ INIT2  │,TYPCALL=  │SVC ││                   │
│         │         │ CHECK  │           │BALR││                   │
│         │         │ CKON   └           └    ┘┘                   │
│         │         │ CKOFF                                        │
│         │         │ UREC                                         │
│         │         │ CALOC                                        │
└──────────────────────────────────────────────────────────────────┘
```

where:

label	is any valid Assembler language label.
INIT1	invokes the first free storage initialization routine, so that free storage requests can be made to access the system disk. Before INIT1 is invoked, no free storage requests may be made. After INIT1 has been invoked, free storage requests may be made, but these are subject to the following restraints until the second free storage management initialization routine has been invoked:

- All requests for USER type storage are changed to requests for NUCLEUS type storage.

- Error checking is limited before initialization is complete. In particular, it is sometimes possible to release a block that was never allocated.

- All requests that are satisfied in high storage must be of a temporary nature, since all storage allocated in high storage is released when the second free storage initialization routine is invoked.

When CP's saved system facility is used, the CMS system is saved at the point just after the A-Disk has been made accessible. It is necessary for DMSFRE to be used before the size of virtual storage is known, since the saved system can be used on any size virtual machine. Thus, the first initialization routine initializes DMSFRE so that limited functions can be requested, while the second initialization routine performs the initialization necessary to allow the full functions of DMSFRE to be exercised.

INIT2 invokes the second initialization routine. This routine is invoked after the size of virtual storage is known, and it performs initialization necessary to allow all the functions of DMSFRE to be used. The second initialization routine performs the following steps:

- Releases all storage that has been allocated in the high storage area.
- Allocates the FREETAB free storage table. This table contains one byte for each 4K page of virtual storage, and so cannot be allocated until the size of virtual storage is known.
- The FREETAB table is initialized, and all storage protection keys are initialized.
- All completely unallocated 4K pages on the low storage nucleus free storage chain are removed to the user chain. Any other necessary operations are performed.

CHECK invokes a routine that checks all free storage chains for consistency and correctness. Thus, it checks to see whether or not any free storage pointers have been destroyed. This option can be used at any time for system debugging.

CKON turns on a flag that causes the CHECK routine to be invoked each time a call is made to DMSFREE or DMSFRET. This can be useful for debugging purposes (for example, when you wish to identify the routine that destroyed free storage management pointers). Care should be taken when using this option, since the CHECK routine is coded to be thorough rather than efficient. Thus, after the CKON option has been invoked, each call to DMSFREE or DMSFRET will take much longer to be completed than before.

CKOFF turns off the flag that was turned on by the CKON option.

UREC is used by DMSABN during the abend recovery process to release all user storage.

CALOC is used by DMSABN after the abend recovery process has been completed. It invokes a routine which returns, in register 0, the number of doublewords of free storage that have been allocated. This number is used by DMSABN to determine whether or not the abend recovery has been successful.

```
          ┌     ┐
TYPCALL= │SVC  │ indicates how control is passed to DMSFES.  Since DMSFRES
         │BALR │ is a nucleus-resident routine, other nucleus-resident
          └     ┘ routines can branch directly to it, (TYPCALL=BALR) while
                  routines that are not nucleus-resident must use SVC
                  linkage (TYPCALL=SVC).
```

ERROR CODES FROM DMSFRES, DMSFREE, AND DMSFRET

A nonzero return code upon return from DMSFRES, DMSFREE, or DMSFRET indicates that the request could not be satisfied. Register 15 contains this return code, indicating which error has occurred. The following codes apply to the DMSFRES, DMSFREE, and DMSFRET macros.

<u>Code</u>	<u>Error</u>
1	(DMSFREE) Insufficient storage space is available to satisfy the request for free storage. In the case of a variable request, even the minimum request could not be satisfied.
2	(DMSFREE or DMSFRET) User storage pointers destroyed.
3	(DMSFREE, DMSFRET, or DMSFRES) Nucleus storage pointers destroyed.
4	(DMSFREE) An invalid size was requested. This error exit is taken if the requested size is not greater than zero. In the case of variable requests, this error exit is taken if the minimum request is greater than the maximum request. (However, the latter error is not detected if DMSFREE is able to satisfy the maximum request.)
5	(DMSFRET) An invalid size was passed to the DMSFRET macro. This error exit is taken if the specified length is not positive.
6	(DMSFRET) The block of storage that is being released was never allocated by DMSFREE. Such an error is detected if one of the following errors is found: • The block does not lie entirely inside either the low storage free storage area or the user program area between FREELOWE and FREEUPPR. • The block crosses a page boundary that separates a page allocated for USER storage from a page allocated for NUCLEUS type storage. • The block overlaps another block already on the free storage chain.
7	(DMSFRET) The address given for the block being released is not doubleword aligned.
8	(DMSFRES) An invalid request code was passed to the DMSFRES routine. Since all request codes are generated by the DMSFRES macro, this error code should never appear.
9	(DMSFREE, DMSFRET, or DMSFRES) Unexpected and unexplained error in the free storage management routine.

CMS HANDLING OF PSW KEYS

The purpose of the CMS Nucleus protection scheme is to protect the CMS nucleus from inadvertent destruction by a user program. Without it, it would be possible, for example, for a FORTRAN user who accidentally assigns an incorrectly subscripted array element to destroy nucleus code, wipe out a crucial table or constant area, or even destroy an entire disk by destroying the contents of the master file directory.

In general, user programs and disk-resident CMS commands are executed with a PSW key of X'E', while nucleus code is executed with a PSW key of X'0'.

There are, however, some exceptions to this rule. Certain disk-resident CMS commands run with a PSW key of X'0', since they have a constant need to modify nucleus pointers and storage. The nucleus routines called by the GET, PUT, READ, and WRITE macros run with a user PSW key of X'E', to increase efficiency.

Two macros are available to any routine that wishes to change its PSW key for some special purpose. These are the DMSKEY macro and the DMSEXS macro.

The DMSKEY macro may be used to change the PSW key to the user value or the nucleus value. The DMSKEY NUCLEUS option causes the current PSW key to be placed in a stack, and a value of 0 to be placed in the PSW key. The DMSKEY USER option causes the current PSW key to be placed in a stack, and a value of X'E' to be placed in the PSW key. The DMSKEY RESET option causes the top value in the DMSKEY stack to be removed and re-inserted into the PSW.

It is a requirement of the CMS system that when a routine terminates, the DMSKEY stack must be empty. This means that a routine should execute a DMSKEY RESET option for each DMSKEY NUCLEUS option and each DMSKEY USER option executed by the routine.

The DMSKEY key stack has a current maximum depth of seven for each routine. In this context, a "routine" is anything invoked by an SVC call.

The DMSKEY LASTUSER option causes the current PSW key to be placed in the stack, and a new key inserted into the PSW, determined as follows: the SVC system save area stack is searched in reverse order (top to bottom) for the first save area corresponding to a user routine. The PSW key that was in effect in that routine is then taken for the new PSW key. (If no user routine is found in the search, then LASTUSER has the same effect as USER.) This option is used by OS macro simulation routines when they wish to enter a user-supplied exit routine; the exit routine is entered with the PSW key of the last user routine on the SVC system save area stack.

The NOSTACK option of DMSKEY may be used with NUCLEUS, USER, or LASTUSER (as in, for example, DMSKEY NUCLEUS,NOSTACK) if the current key is not to be placed on the DMSKEY stack. If this option is used, then no corresponding DMSKEY RESET should be issued.

The DMSEXS ("execute in system mode") macro instruction is useful in situations where a routine is being executed with a user protect key, but wishes to execute a single instruction that, for example, sets a bit in the NUCON area. The single instruction may be specified as the argument to the DMSEXS macro, and that instruction will be executed with a system PSW key.

Whenever possible, CMS commands are executed with a user protect key. This protects the CMS Nucleus in cases where there is an error in the system command that would otherwise destroy the nucleus. If the command must execute a single instruction or small group of instructions that modify nucleus storage, then the DMSKEY or DMSEXS macros are used, so that the system PSW key will be used for as short a period of time as is possible.

CMS SVC HANDLING

DMSITS (INTSVC) is the CMS system SVC handling routine. The general operation of DMSITS is as follows:

1. The SVC new PSW (low storage location X'60') contains, in the address field, the address of DMSITS1. The DMSITS module will be entered whenever a supervisor call is executed.
2. DMSITS allocates a system and user save area. The user save area is used as a register save area (or work area) by the called routine.
3. The called routine is called (via a LPSW or BALR).
4. Upon return from the called routine, the save areas are released.
5. Control is returned to the caller (the routine that originally made the SVC call).

SVC TYPES AND LINKAGE CONVENTIONS

SVC conventions are important to any discussion of CMS because the system is driven by SVCs (supervisor calls). SVCs 202 and 203 are the most common CMS SVCs.

SVC 202

SVC 202 is used both for calling nucleus-resident routines, and for calling routines written as commands (for example, disk resident modules).

A typical coding sequence for an SVC 202 call is the following:

```
LA   R1,PLIST
SVC  202
DC   AL4(ERRADD)
```

Whenever SVC 202 is called, register 1 must point to a parameter list (PLIST). The format of this parameter list depends upon the actual routine or command being called, but the SVC handler will examine the first eight bytes of this parameter list to find the name of the routine or command being called.

The "DC AL4(address)" instruction following the SVC 202 is optional, and may be omitted if the programmer does not expect any errors to occur in the routine or command being called. If included, an error return is made to the address specified in the DC. DMSITS determines whether this DC was inserted by examining the byte following the SVC call inline. A nonzero byte indicates an instruction, a zero value indicates that "DC AL4(address)" follows.

SVC 203

SVC 203 is called by CMS macros to perform various internal system functions. It is used to define SVC calls for which no parameter list is provided. For example, DMSFREE parameters are passed in registers 0 and 1.

A typical calling sequence for an SVC 203 call is as follows:

```
SVC   203
DC    H'code'
```

The halfword decimal code following the SVC 203 indicates the specific routine being called. DMSITS examines this halfword code, taking the absolute value of the code by an LPR instruction. The first byte of the result is ignored, and the second byte of the resulting halfword is used as an index to a branch table. The address of the correct routine is loaded, and control is transferred to it.

It is possible for the address in the SVC 203 index table to be zero. In this case, the index entry will contain an 8-byte routine or command name, which will be handled in the same way as the 8-byte name passed in the parameter list to an SVC 202.

The programmer indicates an error return by the sign of the halfword code. If an error return is desired, then the code is negative. If the code is positive, then no error return is made. The sign of the halfword code has no effect on determining the routine that is to be called, since DMSITS takes the absolute value of the code to determine the routine called.

Since only the second byte of the absolute value of the code is examined by DMSITS, seven bits (bits 1-7) are available as flags or for other uses. Thus, for example, DMSFREE uses these seven bits to indicate such things as conditional requests and variable requests.

When an SVC 203 is invoked, DMSITS stores the halfword code into the NUCON location CODE203, so that the called routine can examine the seven bits made available to it.

All calls made by means of SVC 203 should be made by macros, with the macro expansion computing and specifying the correct halfword code.

User-Handled SVCs

The programmer may use the HNDSVC macro to specify the address of a routine that will handle any SVC call other than for SVC 202 and SVC 203.

In this case, the linkage conventions are as required by the user-specified SVC-handling routine.

OS and DOS/VS Macro Simulation SVC Calls

CMS supports selected SVC calls generated by OS and DOS/VS macros, by simulating the effect of these macro calls. DMSITS is the initial SVC interrupt handler. If the SET DOS command has been issued, a flag in NUCON will indicate that DOS/VS macro simulation is to be used. Control is then passed to DMSDOS. Otherwise, OS macro simulation is assumed and DMSITS passes control to the appropriate OS simulation routine.

Invalid SVC Calls

There are several types of invalid SVC calls recognized by DMSITS.

1. Invalid SVC number. If the SVC number does not fit into any of the four classes described above, then it is not handled by DMSITS. An appropriate error message is displayed at the terminal, and control is returned directly to the caller.

2. Invalid routine name in SVC 202 parameter list. If the routine named in the SVC 202 parameter list is invalid or cannot be found, DMSITS handles the situation in the same way as it handles an error return from a legitimate SVC routine. The error code is -3.

3. Invalid SVC 203 code. If an invalid code follows SVC 203 inline, then an error message is displayed, and the abend routine is called to terminate execution.

SEARCH HIERARCHY FOR SVC 202

When a program issues SVC 202, passing a routine or command name in the parameter list, then DMSITS must be searched for the specified routine or command. (In the case of SVC 203 with a zero in the table entry for the specified index, the same logic must be applied.)

The search algorithm is as follows:

1. A check is made to see if there is a routine with the specified name currently occupying the system transient area. If this is the case, then control is transferred there.

2. The system function name table is searched, to see if a command by this name is a nucleus-resident command. If the search is successful, control goes to the specified nucleus routine.

3. A search is then made for a disk file with the specified name as the filename, and MODULE as the filetype. The search is made in the standard disk search order. If this search is successful, then the specified module is loaded (via the LOADMOD command), and control passes to the storage location now occupied by the command.

4. If all searches so far have failed, then DMSINA (ABBREV) is called, to see if the specified routine name is a valid system abbreviation for a system command or function. User-defined abbreviations and synonyms are also checked. If this search is successful, then steps 2 through 4 are repeated with the full function name.

5. If all searches fail, then an error code of -3 is issued.

Commands Entered from the Terminal

When a command is entered from the terminal, DMSINT processes the command line, and calls the scan routine to convert it into a parameter list consisting of eight-byte entries. The following search is performed:

1. DMSINT searches for a disk file whose filename is the command name, and whose filetype is EXEC. If this search is successful, EXEC is invoked to process the EXEC file.

If not found, the command name is considered to be an abbreviation and the appropriate tables are examined. If found, the abbreviation is replaced by its full equivalent and the search for an EXEC file is repeated.

2. If there is no EXEC file, DMSINT executes SVC 202, passing the scanned parameter list, with the command name in the first eight bytes. DMSITS will perform the search described for SVC 202 in an effort to execute the command.

3. If DMSITS returns to DMSINT with a return code of -3, indicating that the search was unsuccessful, then DMSINT uses the CP DIAGNOSE facility to attempt to execute the command as a CP command.

4. If all of these searches fail, then DMSINT displays the error message UNKNOWN CP/CMS COMMAND.

See Figure 4 for a description of this search for a command name.

USER AND TRANSIENT PROGRAM AREAS

Two areas can hold programs that are loaded from disk. These are called the user program area and the transient program area. (See Figure 3 for a description of CMS storage usage.) A summary of CP, CMS. IPCS, and RSCS modules and their attributes, including whether they reside in the user program area or the transient area is contained in the _IBM/370: Release 5 Guide_.

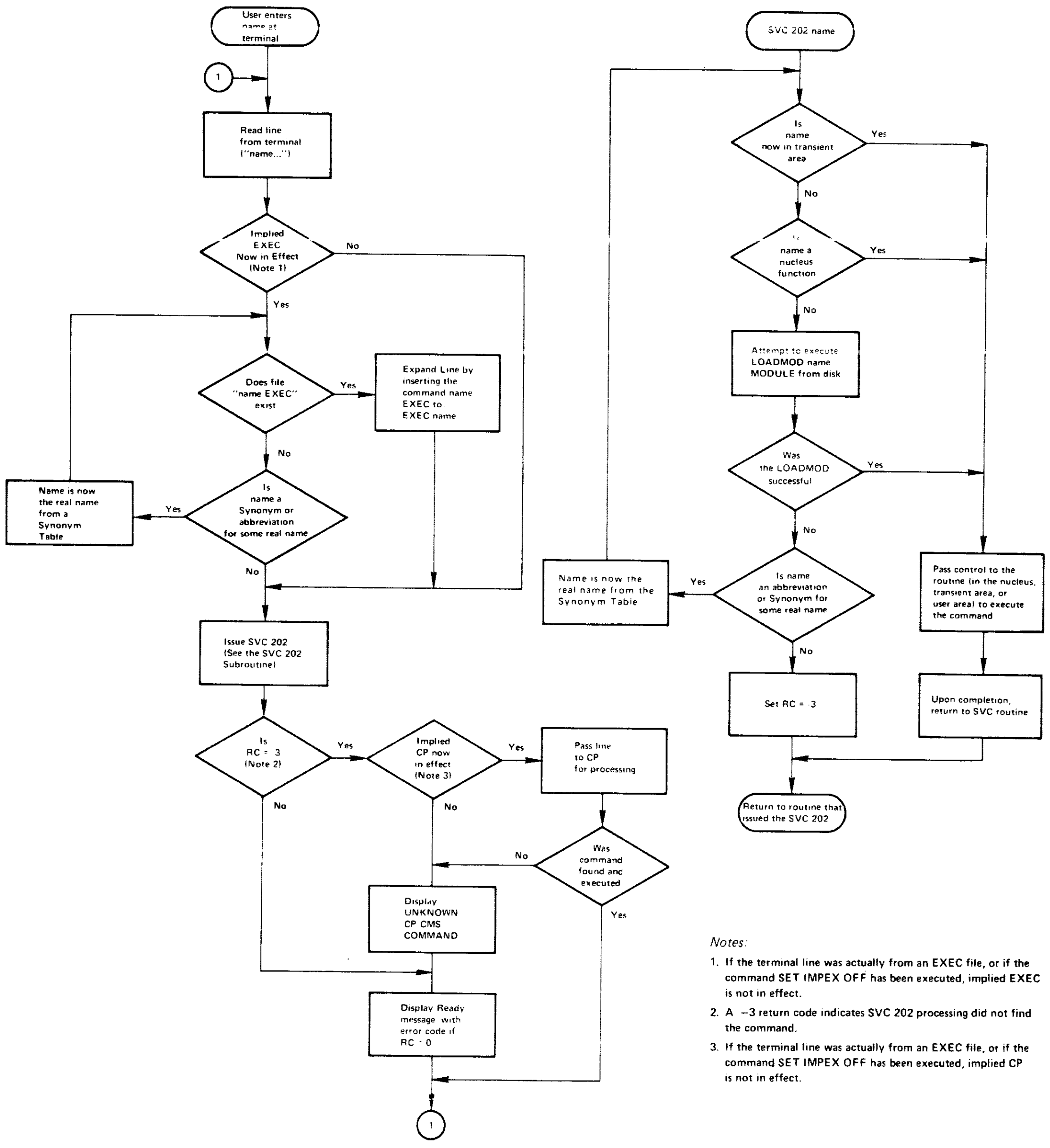
The user program area starts at location X'20000' and extends upward to the loader tables. Generally, all user programs and certain system commands (such as EDIT, and COPYFILE) are executed in the user program area. Since only one program can be executing in the user program area at any one time, it is impossible (without unpredictable results) for one program being executed in the user program area to invoke, by means of SVC 202, a module that is also intended to be executed in the user program area.

The transient program area is two pages long, extending from location X'E000' to location X'FFFF'. It provides an area for system commands that may also be invoked from the user program area by means of an SVC 202 call. When a transient module is called by an SVC, it is normally executed with the PSW system mask disabled for I/O and external interrupts.

The transient program area is also used to handle certain OS macro simulation SVC calls. OS SVC calls are handled by the OS simulation routines located either in the CMSSEG discontiguous shared segment or in the user program area, as close to the loader tables as possible. If DMSITS cannot find the address of a supported OS SVC handling routine, then it loads the file DMSSVT MODULE into the transient area, and lets that routine handle the SVC.

A program being executed in the transient program area may not invoke another program intended for execution in the transient program area, including OS macro simulation SVC calls that are handled by DMSSVT. For example, a program being executed in the transient program area may not invoke the RENAME command. In addition, it may not invoke the OS macro WTO, which generates an SVC 35, which is handled by DMSSVT.

DMSITS starts the programs to be executed in the user program area enabled for all interrupts but starts the programs to be executed in the transient program area disabled for all interrupts. The individual program may have to use the SSM (Set System Mask) instruction to change the current status of its system mask.


User enters name at terminal
1
Read line from terminal ("name...")
Implied EXEC Now in Effect (Note 1)
No
Yes
Does file "name EXEC" exist
Yes
Expand Line by inserting the command name EXEC to EXEC name
No
Is name a Synonym or abbreviation for some real name
Yes
Name is now the real name from a Synonym Table
No
Issue SVC 202 (See the SVC 202 Subroutine)
Is RC = 3 (Note 2)
Yes
Implied CP now in effect (Note 3)
Yes
Pass line to CP for processing
No
No
Was command found and executed
No
Yes
Display UNKNOWN CP CMS COMMAND
Display Ready message with error code if RC ≠ 0
1
SVC 202 name
Is name now in transient area
Yes
No
Is name a nucleus function
Yes
No
Attempt to execute LOADMOD name MODULE from disk
Was the LOADMOD successful
Yes
No
Is name an abbreviation or Synonym for some real name
Yes
Name is now the real name from the Synonym Table
No
Set RC = -3
Pass control to the routine (in the nucleus, transient area, or user area) to execute the command
Upon completion, return to SVC routine
Return to routine that issued the SVC 202

Notes:

1. If the terminal line was actually from an EXEC file, or if the command SET IMPEX OFF has been executed, implied EXEC is not in effect.
2. A -3 return code indicates SVC 202 processing did not find the command.
3. If the terminal line was actually from an EXEC file, or if the command SET IMPEX OFF has been executed, implied CP is not in effect.

Figure 4. CMS Command (and Request) Processing

CALLED ROUTINE START-UP TABLE

Figures 5 and 6 show how the PSW and registers are set up when the called routine is entered.

"Called" Type	System Mask	Storage Key	Problem Bit
SVC 202 or 203 - Nucleus resident	Disabled	System	Off
SVC 202 or 203 - Transient area MODULE	Disabled	User	Off
SVC 202 or 203 - User area	Enabled	User	Off
User-handled	Enabled	User	Off
OS - DOS/VS Nucleus resident	Disabled	System	Off
OS - DOS/VS Transient area module	Disabled	System	Off

Figure 5. PSW Fields When Called Routine Starts

Type	Registers 0 - 1	Registers 2 - 11	Register 12	Register 13	Register 14	Register 15
SVC 202 or 203	Same as caller	Unpredictable	Address of called routine	User save area	Return address to DMSITS	Address of called routine
Other	Same as caller	Same as caller	Address of caller	User save area	Return address to DMSITS	Same as caller

Figure 6. Register Contents When Called Routine Starts

RETURNING TO THE CALLING ROUTINE

When the called routine finishes processing, control is returned to DMSITS, which in turn returns control to the calling routine.

Return Location

The return is accomplished by loading the original SVC old PSW (which was saved at the time DMSITS was first entered), after possibly modifying the address field. The address field modification depends upon the type of SVC call, and upon whether or not the called routine indicated an error return.

For SVC 202 and 203, the called routine indicates a normal return by placing a zero in register 15 and an error return by placing a nonzero code in register 15. If the called routine indicates a normal return, then DMSITS makes a normal return to the calling routine. If the called routine indicates an error return, DMSITS passes the error return to the calling routine, if one was specified, and abnormally terminates if none was specified.

For an SVC 202 not followed by "DC AL4(address)", a normal return is made to the instruction following the SVC instruction, and an error return causes an abend. For an SVC 202 followed by "DC AL4(address)", a normal return is made to the instruction following the DC, and an error return is made to the address specified in the DC. In either case, register 15 contains the return code passed back by the called routine.

For an SVC 203 with a positive halfword code, a normal return is made to the instruction following the halfword code, and an error return causes an abend. For an SVC 203 with a negative halfword code, both normal and error returns are made to the instruction following the halfword code. In any case, register 15 contains the return code passed back by the called routine.

For macro simulation SVC calls, and for user-handled SVC calls, no error return is recognized by DMSITS. As a result, DMSITS always returns to the calling routine by loading the SVC old PSW, which was saved when DMSITS was first entered.

Register Restoration

Upon entry to DMSITS, all registers are saved as they were when the SVC instruction was first executed. Upon exiting from DMSITS, all registers are restored from the area in which they were saved at entry.

The exception to this is register 15 in the case of SVC 202 and 203. Upon return to the calling routine, register 15 always contains the value that was in register 15 when the called routine returned to DMSITS after it had completed processing.

Called Routine Modifications to System Area

If the called routine has system status, so that it runs with a PSW storage protect key of 0, then it may store new values into the System Save Area.

If the called routine wishes to modify the location to which control is to be returned, it must modify the following fields:

- For SVC 202 and 203, it must modify the NUMRET and ERRET (normal and error return address) fields.

- For other SVCs, it must modify the address field of OLDPSW.

To modify the registers that are to be returned to the calling routine, the fields EGPR1, EGPR2, ..., EGPR15 must be modified.

If this action is taken by the called routine, then the SVCTRACE facility may print misleading information, since SVCTRACE assumes that these fields are exactly as they were when DMSITS was first entered. Whenever an SVC call is made, DMSITS allocates two save areas for that particular SVC call. Save areas are allocated as needed. For each SVC call, a system and user save area are needed.

When the SVC-called routine returns, the save areas are not released, but are kept for the next SVC. At the completion of each command, all SVC save areas allocated by that command are released.

The System Save Area is used by DMSITS to save the value of the SVC old PSW at the time of the SVC call, the calling routine's registers at the time of the call, and any other necessary control information. Since SVC calls can be nested, there can be several of these save areas at one time. The system save area is allocated in protected free storage.

The user save area contains 12 doublewords (24 words), allocated in unprotected free storage. DMSITS does not use this area at all, but simply passes a pointer to this area (via register 13.) The called routine can use this area as a temporary work area, or as a register save area. There is one user save area for each system save area. The USAVEPTR field in the system save area points to the user save area.

The exact format of the system save area can be found in the _VM/370 Data Areas and Control Block Logic_. The most important fields, and their uses, are as follows:

Field	Usage
CALLER	(Fullword) The address of the SVC instruction that resulted in this call.
CALLEE	(Doubleword) Eight-byte symbolic name of the called routine. For OS and user-handled SVC calls, this field contains a character string of the form SVC nnn, where nnn is the SVC number in decimal.
CODE	(Halfword) For SVC 203, this field contains the halfword code following the SVC instruction line.
OLDPSW	(Doubleword) The SVC old PSW at the time that DMSITS was entered.
NRMRET	(Fullword) The address of the calling routine to which control is to be passed in the case of a normal return from the called routine.
ERRET	(Fullword) The address of the calling routine to which control is to be passed in the case of an error return from the called routine.
EGPRS	(16 Fullwords, separately labeled EGPR0, EGPR1, EGPR2, EGPR3, ..., EGPR15) The entry registers. The contents of the general registers at entry to DMSITS are stored in these fields.
EFPRS	(4 Doublewords, separately labeled EFPR0, EFPR2, EFPR4, EFPR6) The entry floating-point registers. The contents of the floating-point registers at entry to DMSITS are stored in these fields.
SSAVENXT	(Fullword) The address of the next system save area in the chain. This points to the system save area that is being used, or will be used, for any SVC call nested in relation to the current one.
SSAVEPRV	(Fullword) The address of the previous system save area in the chain. This points to the system save area for the SVC call in relation to which the current call is nested.
USAVEPTR	(Fullword) Pointer to the user save area for this SVC call.

CMS Interface for Display Terminals

CMS has an interface that allows it to display large amounts of data in a very rapid fashion. This interface for 3270 display terminals (also 3138, 3148, and 3158) is much faster and has less overhead than the normal write because it displays up to 1760 characters in one operation, instead of issuing 22 individual writes of 80 characters each (that is one write per line on a display terminal). Data that is displayed in the screen output area with this interface is nct placed in the console spool file.

The DISPW macro allows you to use this display terminal interface. It generates a calling sequence for the CMS display terminal interface module, DMSGIO. DMSGIO creates a channel program and issues a DIAGNOSE instruction (Code X'58') to display the data. DMSGIC is a TEXT file which must be loaded in order to use DISPW. The format of the CMS DISPW macro is:

```
┌──────────────────────────────────────────────────────────────────────────┐
|            |            |                                                |
| [label]    |  DISPW     |  bufad |,LINE=n|  |,BYTES=bbbb|                |
|            |            |        |,LINE=0|  |,BYTES=1760|                |
|            |            |        └       ┘  └           ┘                |
|            |            |         [ERASE=YES]   [CANCEL=YES]             |
└──────────────────────────────────────────────────────────────────────────┘
```

<u>where</u>:

label	is an optional macro statement label.
bufad	is the address of a buffer containing the data to be written to the display terminal.

LINE=n
LINE=0
	is the number of the line, 0 to 23, on the display terminal that is to be written. Line number 0 is the default.

BYTES=bbbb
BYTES=1760
	is the number of bytes (0 to 1760) to be written on the display terminal. 1760 bytes is the default.

[ERASE=YES]	specifies that the display screen is to be erased before the current data is written. The screen is erased regardless of the line or number of bytes to be displayed. Specifying ERASE=YES causes the screen to go into "MORE" status.
[CANCEL=YES]	causes the CANCEL operation to be performed: the output area is erased.

Note: It is advisable for the user to save registers before issuing the DISPW macro and to restore them after the macro, because neither the macro nor its called modules save the user's registers.

OS Macro Simulation Under CMS

When a language processor or a user-written program is executing in the CMS environment and using OS-type functions, it is not executing OS code. Instead, CMS provides routines that simulate the OS functions required to support OS language processors and their generated object code.

CMS functionally simulates the OS macros in a way that presents equivalent results to programs executing under CMS. The OS macros are supported only to the extent stated in the publications for the supported language processors, and then only to the extent necessary to successfully satisfy the specific requirement of the supervisory function.

The restrictions for COBOL and PL/I program execution listed in "Executing a Program that Uses OS Macros" in the _VM/370 Planning and System Generation Guide_ exist because of the limited CMS simulation of the OS macros.

Figure 7 shows the OS macro functions that are partially or completely simulated, as defined by SVC number.

OS Data Management Simulation

The disk format and data base organization of CMS are different from those of OS. A CMS file produced by an OS program running under CMS and written on a CMS disk, has a different format from that of an OS data set produced by the same OS program running under OS and written on an OS disk. The data is exactly the same, but its format is different. (An OS disk is one that has been formatted by an OS program, such as IBCDASDI.)

HANDLING FILES THAT RESIDE ON CMS DISKS

CMS can read, write, or update any OS data that resides on a CMS disk. By simulating OS macros, CMS simulates the following access methods so that OS data organized by these access methods can reside on CMS disks:

direct	identifying a record by a key or by its relative position within the data set.
partitioned	seeking a named member within the data set.
sequential	accessing a record in a sequence in relation to preceding or following items in the data set.

Refer to Figure 7 and the "Simulation Notes," then read "Access Method Support" to see how CMS handles these access methods.

Since CMS does not simulate the indexed sequential access method (ISAM), no OS program that uses ISAM can execute under CMS. Therefore, no program can write an indexed sequential data set on a CMS disk.

HANDLING FILES THAT RESIDE ON OS OR DOS DISKS

By simulating OS macros, CMS can read, but not write or update, CS sequential and partitioned data sets that reside on OS disks. Using the same simulated OS macros, CMS can read DOS sequential files that reside on DOS disks. The OS macros handle the DOS data as if it were OS data. Thus, a DOS sequential file can be used as input to an OS program running under CMS.

However, an OS sequential or partitioned data set that resides on an OS disk can be written or updated only by an OS program running in a real OS machine.

CMS can execute programs that read and write VSAM files from OS programs written in the VS BASIC, COBOL, or PL/I programming languages. This CMS support is based on the DOS/VS Access Method Services and Virtual Storage Access Method (VSAM) and, therefore, the OS user is limited to those VSAM functions that are available under DOS/VS.

Macro Name	SVC Number	Function
XDAP[1]	00	Read or write direct access volumes
WAIT	01	Wait for an I/O completion
POST	02	Post the I/O completion
EXIT/RETURN	03	Return from a called phase
GETMAIN	04	Conditionally acquire user storage
FREEMAIN	05	Release user-acquired storage
GETPOOL	-	Simulate as SVC 10
FREEPOOL	-	Simulate as SVC 10
LINK	06	Link control to another phase
XCTL	07	Delete, then link control to another load phase
LOAD	08	Read a phase into storage
DELETE	09	Delete a loaded phase
GETMAIN/ FREEMAIN	10	Manipulate user free storage
TIME[1]	11	Get the time of day
ABEND	13	Terminate processing
SPIE[1]	14	Allow processing program to handle program interrupts
RESTORE[1]	17	Effective NOP
BLDL/FIND[1]	18	Manipulate simulated partitioned data files
OPEN	19	Activate a data file
CLOSE	20	Deactivate a data file
STOW[1]	21	Manipulate partitioned directories
OPENJ	22	Activate a data file
TCLOSE	23	Temporarily deactivate a data file
DEVTYPE[1]	24	Obtain device-type physical characteristics
TRKBAL	25	NOP
FEOV	31	Set forced EOV error code
WTO/WTOR[1]	35	Communicate with the terminal
EXTRACT[1]	40	Effective NOP
IDENTIFY[1]	41	Add entry to loader table
ATTACH[1]	42	Effective LINK
CHAP[1]	44	Effective NOP
TTIMER[1]	46	Access or cancel timer
STIMER[1]	47	Set timer
DEQ[1]	48	Effective NOP
SNAP[1]	51	Dump specified areas of storage
ENQ[1]	56	Effective NOP
FREEDBUF	57	Release a free storage buffer
STAE	60	Allow processing program to decipher abend conditions
DETACH[1]	62	Effective NOP
CHKPT[1]	63	Effective NOP
RDJFCB[1]	64	Obtain information from FILEDEF command
SYNAD[1]	68	Handle data set error conditions
BSP[1]	69	Back up a record on a tape or disk
GET/PUT	-	Access system-blocked data
READ/WRITE	-	Access system-record data
NOTE/POINT	-	Manage data set positioning
CHECK	-	Verify READ/WRITE completion
TGET/TPUT	93	Read or write a terminal line
TCLEARQ	94	Clear terminal input queue
STAX	96	Create an attention exit block

[1]Simulated in the transient routine DMSSVT. Other simulation routines reside in the nucleus.

Figure 7. Simulated OS Supervisor Calls

SIMULATION NOTES

Because CMS has its own file system and is a single-user system operating in a virtual machine with virtual storage, there are certain restrictions for the simulated OS function in CMS. For example, HIARCHY options and options that are used only by OS multitasking systems are ignored by CMS.

Due to the design of the CMS loader, an XCTL from the explicitly loaded phase, followed by a LINK by succeeding phases, may cause unpredictable results.

Listed below are descriptions of all the OS macro functions that are simulated by CMS as seen by the programmer. Implementation and program results that differ from those given in _OS Data Management Macro Instructions_ and _OS Supervisor Services and Macro Instructions_ are stated. HIARCHY options and those used only by OS multitasking systems are ignored by CMS. Validity checking is not performed within the simulation routines. The entry point name in LINK, XCTL, and LOAD (SVC 6, 7, 8) must be a member name or alias in a TXTLIB directory unless the COMPSWT is set to on. If the COMPSWT is on, SVC 6, 7, and 8 must specify a module name. This switch is turned on and off by using the COMPSWT macro. See the _VM/370 CMS Command and Macro Reference_ for descriptions of all CMS user macros.

Macro-SVC No.	Differences in Implementation
XDAP-SVC0	The TYPE option must be R or W; the V, I, and K options are not supported. The BLKREF-ADDR must point to an item number acquired by a NOTE macro. Other options associated with V, I, or K are not supported.
WAIT-SVC1	All options of WAIT are supported. The WAIT routine waits for the completion bit to be set in the specified ECBs.
POST-SVC2	All options of POST are supported. POST sets a completion code and a completion bit in the specified ECB.
EXIT/RETURN -SVC3	Post ECB, execute end of task routines, release phase storage, unchain and free latest request block, and restore registers depending upon whether this is an exit or return from a linked or an attached routine.
GETMAIN-SVC4	All options of GETMAIN are supported except SP and HIARCHY, which are ignored by CMS, and LC and LV, which will result in abnormal termination if used. GETMAIN gets blocks of free storage.
FREEMAIN-SVC5	All options of FREEMAIN are supported except SP, which is ignored by CMS, and L, which will result in abnormal termination if used. FREEMAIN frees blocks of storage acquired by GETMAIN.
LINK-SVC6	The DCB and HIARCHY options are ignored by CMS. All other options of LINK are supported. LINK loads the specified program into storage (if necessary) and passes control to the specified entry point.
XCTL-SVC7	The DCB and HIARCHY options are ignored by CMS. All other options of XCTL are supported. XCTL loads the specified program into storage (if necessary) and passes control to the specified entry point.

Macro-SVC No.	Differences in Implementation
LOAD-SVC8	The DCB and HIARCHY options are ignored by CMS. All other options of LOAD are supported. LOAD loads the specified program into storage (if necessary) and returns the address of the specified entry point in register zero. However, if the specified entry point is not in core when SVC 8 is issued, and the subroutine contains VCONs that cannot be resolved within that TXTLIB member, CMS will attempt to resolve these references, and may return another entry point address. To insure a correct address in register zero, the user should bring such subroutines into core either by the CMS LOAD/INCLUDE commands or by a VCON in the user program.
GETPOOL/ FREEPOOL	All the options of GETPOOL and FREEPOOL are supported. GETPOOL constructs a buffer pool and stores the address of a buffer pool control block in the DCB. FREEPOOL frees a buffer pool constructed by GETPOOL.
DELETE-SVC9	All the options of DELETE are supported. DELETE decreases the use count by one and, if the result is zero, frees the corresponding virtual storage. Code 4 is returned in register 15 if the phase is not found.
GETMAIN/ FREEMAIN-SVC10	All the options of GETMAIN and FREEMAIN are supported except SP and HIARCHY, which are ignored by CMS.
TIME-SVC11	All the options of TIME except MIC are supported. TIME returns the time of day to the calling program.
ABEND-SVC13	The completion code parameter is supported. The DUMP parameter is not. If a STAE request is outstanding, control is given to the proper STAE routine. If a STAE routine is not outstanding, a message indicating that an abend has occurred is printed on the terminal along with the completion code.
SPIE-SVC14	All the options of SPIE are supported. The SPIE routine specifies interruption exit routines and program interruption types that will cause the exit routine to receive control.
RESTORE-SVC17	The RESTORE routine in CMS is a NOP. It returns control to the user.
BLDL-SVC18	BLDL is an effective NOP for LINKLIBs and JOBLIBs. For TXTLIBs and MACLIBs, item numbers are filled in the TTR field of the BLDL list; the K, Z, and user data fields, as described in <u>OS/VS Data Management Macro Instructions</u>, are set to zeros. The "alias" bit of the C field is supported, and the remaining bits in the C field are set to zero.
FIND-SVC18	All the options of FIND are supported. FIND sets the read/write pointer to the item number of the specified member.
STOW-SVC21	All the options of STOW are supported. The "alias" bit is supported, but the user data field is not stored in the MACLIB directory since CMS MACLIBs do not contain user data fields.

Macro-SVC No.	Differences in Implementation
OPEN/OPENJ-SVC19/22	All the options of OPEN and OPENJ are supported except for the DISP and RDBACK options, which are ignored. OPEN creates a CMSCB (if necessary), completes the DCB, and merges necessary fields of the DCB and CMSCB.
CLOSE/TCLOSE-SVC20/23	All the options of CLOSE and TCLOSE are supported except for the DISP option, which is ignored. The DCB is restored to its condition before OPEN. If the device type is disk, the file is closed. If the device type is tape, the REREAD option is treated as a REWIND.
DEVTYPE-SVC24	All the options of DEVTYPE are supported except for the RPS option, which is ignored. DEVTYPE moves device characteristic information for a specified data set into a specified user area.
FEOV-SVC31	Control is returned to CMS with an error code of 4 in register 15.
WTO/WTOR-SVC35	All options of WTO and WTOR are supported except those options concerned with multiple console support. WTO displays a message at the operator's console. WTOR displays a message at the operator's console, waits for a reply, moves the reply to the specified area, sets a completion bit in the specified ECB, and returns.
EXTRACT-SVC40	The EXTRACT routine in CMS is essentially a NOP. The user-provided answer area is set to zeros and control is returned to the user with a return code of 4 in register 15.
IDENTIFY-SVC41	The IDENTIFY routine in CMS adds a RPQUEST block to the load request chain for the requested name and address.
ATTACH-SVC42	All the options of ATTACH are supported in CMS as in OS PCP. The following options are ignored by CMS: DCB, LPMOD, DPMOD, HIARCHY, GSPV, GSPL, SHSPV, SHSPL, SZERO, PURGE, ASYNCH, and TASKLIB. ATTACH passes control to the routine specified, fills in an ECB completion bit if an ECB is specified, passes control to an exit routine if one is specified, and returns control to the instruction following the ATTACH. Since CMS is not a multitasking system, a phase requested by the ATTACH macro must return to CMS.
CHAP-SVC44	The CHAP routine in CMS is a NOP. It returns control to the user.
TTIMER-SVC46	All the options of TTIMER are supported.
STIMER-SVC47	All options of STIMER are supported except for TASK and WAIT. The TASK option is treated as if the REAL option had been specified, and the WAIT option is treated as a NOP; it returns control to the user.
DEQ-SVC48	The DEQ routine in CMS is a NOP. It returns control to the user.

Macro-SVC No.	Differences in Implementation
SNAP-SVC51	Except for SDATA, PDATA, and DCB, all options of the SNAP macro are processed normally. SDATA and PDATA are ignored. Processing for the DCB option is as follows. The DBC address specified with SNAP is used to verify that the file associated with the DCB is open. If it is not open, control is returned to the caller with a return code of 4. If the file is open, then storage is dumped (unless the FCB indicates a DUMMY device type). SNAP always dumps output to the printer. The dump contains the PSW, the registers, and the storage specified.
ENQ-SVC56	The ENQ routine in CMS is a NOP. It returns control to the user.
FREEDBUF-SVC57	All the options of FREEDBUF are supported. FREEDBUF returns a buffer to the buffer pool assigned to the specified DCB.
STAE-SVC60	All the options of STAE are supported except for the XCTL option, which is set to XCTL=YES; the PURGE option, which is set to HALT; and the ASYNCH option, which is set to NO. STAE creates, overlays, or cancels a STAE control block as requested. STAE retry is not supported.
DETACH-SVC62	The DETACH routine in CMS is a NOP. It returns control to the user.
CHKPT-SVC63	The CHKPT routine is a NOP. It returns control to the user.
RDJFCB-SVC64	All the options of RDJFCB are supported. RDJFCB causes a Job File Control Block (JFCB) to be read from a CMS Control Block (CMSCB) into real storage for each data control block specified. CMSCBs are created by FILEDEF commands.
SYNADAF-SVC68	All the options of SYNADAF are supported. SYNADAF analyzes an I/O error and creates an error message in a work buffer.
SYNADRLS-SVC68	All the options of SYNADRLS are supported. SYNADRLS frees the work area acquired by SYNAD and deletes the work area from the save area chain.
BSP-SVC69	All the options of BSP are supported. BSP decrements the item pointer by one block.
TGET/TPUT-SVC93	TGET and TPUT operate as if EDIT and WAIT were coded. TGET reads a terminal line. TPUT writes a terminal line.
TCLEARQ-SVC94	TCLEARQ in CMS clears the input terminal queue and returns control to the user.
STAX-SVC96	Updates a queue of CMTAXEs each of which defines an attention exit level.
NOTE	All the options of NOTE are supported. NOTE returns the item number of the last block read or written.

Macro-SVC No.	Differences in Implementation
POINT	All the options of POINT are supported. POINT causes the control program to start processing the next read or write operation at the specified item number. The TTR field in the block address is used as an item number.
CHECK	All the options of CHECK are supported. CHECK tests the I/O operation for errors and exceptional conditions.
DCB	The following fields of a DCB may be specified, relative to the particular access method indicated:

Operand	BDAM	BPAM	BSAM	QSAM
BFALN	F,D	F,D	F,D	F,D
BLKSIZE	n(number)	n	n	n
BUFCB	a(address)	a	a	a
BUFL	n	n	n	n
BUFNO	n	n	n	n
DDNAME	s(symbol)	s	s	s
DSORG	DA	PO	PS	PS
EODAD	-	a	a	a
EXLST	a	a	a	a
KEYLEN	n	-	n	-
LIMCT	n	-	-	-
LRECL	-	n	n	n
MACRF	R,W	R,W	R,W, P	G,P,L,M
OPTCD	A,E,F,R	-	-	-
RECFM	F,V,U	F,V,U	F,V,B,S,A,M,U	F,V,B,U,A,M,S
SYNAD	a	a	a	a
NCP	-	n	n	-

ACCESS METHOD SUPPORT

The manipulation of data is governed by an access method. To facilitate the execution of OS Code under CMS, the processing program must see data as OS would present it. For instance, when the processors expect an access method to acquire input source cards sequentially, CMS invokes specially written routines that simulate the OS sequential access method and pass data to the processors in the format that the OS access methods would have produced. Therefore, data appears in storage as if it had been manipulated using an OS access method. For example, block descriptor words (BDW), buffer pool management, and variable records are updated in storage as if an OS access method had processed the data. The actual writing to and reading from the I/O device is handled by CMS file management. Note that the character string X'61FFFF61' is interpreted by CMS as an end of file indicator.

The essential work of the volume table of contents (VTOC) and the data set control block (DSCB) is done in CMS by a master file directory (MFD) which updates the disk contents, and a file status table (FST) (one for each data file). All disks are formatted in physical blocks of 800 bytes.

CMS continues to update the OS format, within its own format, on the auxiliary device, for files whose filemode number is 4. That is, the block and record descriptor words (BDW and RDW) are written along with the data. If a data set consists of blocked records, the data is written to, and read from, the I/O device in physical blocks, rather than logical records. CMS also simulates the specific methods of manipulating data sets.

To accomplish this simulation, CMS supports certain essential macros for the following access methods:

- BDAM (direct) -- identifying a record by a key or by its relative position within the data set.
- BPAM (partitioned) -- seeking a named member within data set.
- BSAM/QSAM (sequential) -- accessing a record in a sequence in relation to preceding or following records.
- VSAM (direct or sequential) -- accessing a record sequentially or directly by key or address.

  Note: CMS support of OS VSAM files is based on DOS/VS Access Method Services and Virtal Storage Access Method (VSAM). Therefore, the OS user is restricted to those functions available under "DOS/VS Access Method Services." See the section "CMS Support for OS and DCS VSAM Functions" for details.

CMS also updates those portions of the OS control blocks that are needed by the OS simulation routines to support a program during execution. Most of the simulated supervisory OS control blocks are contained in the following two CMS control blocks:

CMSCVT
: simulates the communication vector table. Location 16 contains the address of the CVT control section.

CMSCB
: is allocated from system free storage whenever a FILEDEF command or an OPEN (SVC 19) is issued for a data set. The CMS Control Block consists of a file control block (FCB) for the data file, and partial simulation of the job file control block (JFCB), input/output block (IOB), and data extent block (DEB).

The data control block (DCB) and the data event control block (DECB) are used by the access method simulation routines of CMS.

Note: The results may be unpredictable if two DCBs access the same data set at the same time.

The GET and PUT macros are not supported for use with spanned records. READ and WRITE are supported for spanned records, provided the filemode number is 4, and the data set is physical sequential (BSAM) format.

GET (QSAM)
: All the QSAM options of GET are supported. Substitute mode is handled the same as move mode. If the DCBRECFM is FB, the filemode number is 4, and the last block is a short block, an EOF indicator (X'61FFFF61') must be present in the last block after the last record.

GET (QISAM)
: QISAM is nct supported in CMS.

PUT (QSAM)
: All the QSAM options of PUT are supported. Substitute mode is handled the same as move mode. If the DCBRECFM is FB, the filemode number is 4, and the last block is a short blcck, an EOF indicator is written in the last block after the last record.

PUT (QISAM)
QISAM is not supported in CMS.

PUTX
PUTX support is provided only for data sets opened for QSAM-UPDATE with simple buffering.

READ/WRITE (BISAM)
BISAM is not supported in CMS.

READ/WRITE (BSAM and BPAM)
All the BSAM and BPAM options of READ and WRITE are supported except for the SE option (read backwards).

READ (Offset Read of Keyed BDAM dataset)
This type of READ is not supported because it is used only for spanned records.

READ/WRITE (BDAM)
All the BDAM and BSAM (create) options of READ and WRITE are supported except for the R and RU options.

When an input or output error occurs, do not depend on OS sense bytes. An error code is supplied by CMS in the ECB in place of the sense bytes. These error codes differ for various types of devices and their meaning can be found in the IBM VM/370: System Messages, under DMS message 120S.

BDAM Restrictions

The four methods of accessing BDAM records are:

1. Relative Block RRR
2. Relative Track TTR
3. Relative Track and Key TTKey
4. Actual Address MBBCCHHR

The restrictions on these access methods are as follows:

- Only the BDAM identifiers underlined above can be used to refer to records, since CMS files have a two-byte record identifier.

- CMS BDAM files are always created with 255 records on the first logical track, and 256 records on all other logical tracks, regardless of the block size. If BDAM methods 2, 3, or 4 are used and the RECFM is U or V, the BDAM user must either write 255 records on the first track and 256 records on every track thereafter, or he must not update the track indicator until a NO SPACE FOUND message is returned on a write. For method 3 (WRITE ADD), this message occurs when no more dummy records can be found on a WRITE request. For methods 2 and 4, this will not occur, and the track indicator will be updated only when the record indicator reaches 256 and overflows into the track indicator.

- Two files of the same filetype, both of which use keys, cannot be open at the same time. If a program that is updating keys does not close the file it is updating for some reason, such as a system failure or another IPL operation, the original keys for files that are not fixed format are saved in a temporary file with the same filetype and a filename of $KEYSAVE. To finish the update, run the program again.

- Once a file is created using keys, additions to the file must not be made without using keys and specifying the original length.

- The number of records in the data set extent must be specified using the FILEDEF command. The default size is 50 records.

- The minimum LRECL for a CMS BDAM file with keys is eight bytes.

READING OS DATA SETS AND DOS FILES USING OS MACROS

CMS users can read OS sequential and partitioned data sets that reside on OS disks. The CMS MOVEFILE command can be used to manipulate those data sets, and the OS QSAM, BPAM, and BSAM macros can be executed under CMS to read them.

The CMS MOVEFILE command and the same OS macros can also be used to manipulate and read DOS sequential files that reside on DOS disks. The OS macros handle the DOS data as if it were OS data.

The following OS Release 20.0 BSAM, BPAM, and QSAM macros can be used with CMS to read OS data sets and DOS files:

BLDL	ENQ	RDJFCB
BSP	FIND	READ
CHECK	GET	SYNADAF
CLOSE	NOTE	SYNADRLS
DEQ	POINT	WAIT
DEVTYPE	POST	

CMS supports the following disk formats for the OS and OS/VS sequential and partitioned access methods:

- Split cylinders
- User labels
- Track overflow
- Alternate tracks

As in OS, the CMS support of the BSP macro produces a return code of 4 when attempting to backspace over a tape mark or when a beginning of an extent is found on an OS data set or a DOS file. If the data set or file contains split cylinders, an attempt to backspace within an extent, resulting in a cylinder switch, also produces a return code of 4.

The ACCESS Command

Before CMS can read an OS data set or DOS file that resides on a non-CMS disk, you must issue the CMS ACCESS command to make the disk on which it resides available to CMS.

The format of the ACCESS command is:

```
ACCESS cuu mode[/ext]
```

You must not specify options or file identification when accessing an OS or DOS disk.

The FILEDEF Command

You then issue the FILEDEF command to assign a CMS file identification to the OS data set or DOS file so that CMS can read it. The format of the FILEDEF command used for this purpose is:

```
┌─────────┬──────────────────────────────────────────────────────────────┐
│         │                                                              │
│ FIledef │ ┌ddname┐ ┌ ┌               ┌  ┐┐ ┌               ┐ ┐         │
│         │ │  nn  │ │ │DISK fn ft     │fm││ │DSN ?          │ │         │
│         │ │  *   │ │ │               │A1││ │DSN q1 [q2...]││ │         │
│         │ └      ┘ │ └               └  ┘┘ └               ┘ │         │
│         │          │      ┌              ┌  ┐┐                │         │
│         │          │ DISK │fn    ft      │fm││                │         │
│         │          │      │FILE ddname   │A1││                │         │
│         │          │      └              └  ┘┘                │         │
│         │          │                                          │         │
│         │          │ DUMMY                                    │         │
│         │                   ┌                 ┐                         │
│         │ Related Option:   │MEMBER membername│                         │
│         │                   │CONCAT           │                         │
│         │                   └                 ┘                         │
└─────────┴──────────────────────────────────────────────────────────────┘
```

If you are issuing a FILEDEF for a DOS file, note that the OS program that will use the DOS file must have a DCB for it. For "ddname" in the FILEDEF command line, use the ddname in that DCB. With the DSN operand, enter the file-id of the DOS file.

Sometimes, CMS issues the FILEDEF command for you. Although the CMS MOVEFILE command, the supported CMS program product interfaces, and the CMS OPEN routine each issue a default FILEDEF, you should issue the FILEDEF command yourself to ensure the appropriate file is defined.

After you have issued the ACCESS and FILEDEF commands for an OS sequential or partitioned data set or DOS sequential file, CMS commands (such as ASSEMBLE and STATE) can refer to the OS data set or DOS file just as if it were a CMS file.

Several other CMS commands can be used with OS data sets and DOS files that do not reside on CMS disks. See the _VM/370 CMS Command and Macro Reference_ for a complete description of the CMS ACCESS, FILEDEF, LISTDS, MOVEFILE, QUERY, RELEASE, and STATE commands.

For restrictions on reading OS data sets and DOS files under CMS, see the _VM/370 Planning and System Generation Guide_.

The CMS FILEDEF command allows you to specify the I/O device and the file characteristics to be used by a program at execution time. In conjunction with the OS simulation scheme, FILEDEF simulates the functions of the data definition JCL statement.

FILEDEF may be used only with programs using OS macros and functions. For example:

```
    filedef  file1  disk  proga  data  a1
```

After issuing this command, your program referring to FILE1 would access PROGA DATA on your A-disk.

If you wished to supply data from your terminal for FILE1, you could issue the command:

```
filedef file1 terminal
```

and enter the data for your program without recompiling.

```
fi tapein tap2 (recfm fb lrecl 50 block 100 9track den 800)
```

After issuing this command, programs referring to TAPEIN will access a tape at virtual address 182. (Each tape unit in the CMS environment has a symbolic name associated with it.) The tape must have been previously attached to the virtual machine by the VM/370 operator.

The AUXPROC Option of the FILEDEF Command

The AUXPROC option can only be used by a program call to FILEDEF and not from the terminal. The CMS language interface programs use this feature for special I/O handling of certain (utility) data sets.

The AUXPROC option, followed by a fullword address of an auxiliary processing routine, allows that routine to receive control from DMSSEB before any device I/O is performed. At the completion of its processing, the auxiliary routine returns control to DMSSEB signaling whether or not I/O has been performed. If it has not been done, DMSSEB performs the appropriate device I/O.

When control is received from DMSSEB, the general-purpose registers contain the following information:

```
              GPR2  = Data Control Block (DCB) address
              GPR3  = Base register for DMSSEB
              GPR8  = CMS OPSECT address
              GPR11 = File Control Block (FCB) address
              GPR14 = Return address in DMSSEB
              GPR15 = Auxiliary processing routine address
all other registers = Work registers
```

The auxiliary processing routine must provide a save area in which to save the general registers; this routine must also perform the save operation. DMSSEB does not provide the address of a save area in general register 13, as is usually the case. When control returns to DMSSEB, the general registers must be restored to their original values. Control is returned to DMSSEB by branching to the address contained in general register 14.

GPR15 is used by the auxiliary processing routine to inform to DMSSEB of the action that has been or should be taken with the data block as follows:

Register Content	Action
GPR15=0	No I/O performed by AUXPROC routine; DMSSEB will perform I/O.
GPR15<0	I/O performed by AUXPROC routine and error was encountered. DMSSEB will take error action.
GPR15>0	I/O performed by AUXPROC routine with residual count in GPR15; DMSSEB returns normally.
GPR15=64K	I/O performed by AUXPROC routine with zero residual count.

DOS/VS Support Under CMS

CMS supports interactive program development for DOS/VS Release 31, 32, 33 and 34. This includes creating, compiling, testing, debugging, and executing commercial application programs. The DOS/VS programs can be executed in a CMS virtual machine or in a CMS Batch Facility virtual machine.

DOS/VS files and libraries can be read under CMS. VSAM data sets can be read and written under CMS.

The CMS DOS environment (called CMS/DOS) provides many of the same facilities that are available in DOS/VS. However, CMS/DOS supports only those facilities that are supported by a single (background) partition. The DOS/VS facilities supported by CMS/DOS are:

- DOS/VS linkage editor
- Fetch support
- DOS/VS Supervisor and I/O macros
- DOS/VS Supervisor control block support
- Transient area support
- DOS/VS VSAM macros

This environment is entered each time the CMS SET DOS ON command is issued; VSAM functions are available in CMS/DOS only if the SET DOS ON (VSAM) command is issued. In the CMS/DOS environment, CMS supports many DOS/VS facilities, but does not support OS simulation. When you no longer need DOS/VS support under CMS, you issue the SET DOS OFF command and DOS/VS facilities are no longer available.

CMS/DOS can execute programs that use the sequential access method (SAM) and virtual storage access method (VSAM), and can access DOS/VS libraries.

CMS/DOS cannot execute programs that have execution-time restrictions, such as programs that use sort exits, teleprocessing access methods, or multitasking. DOS/VS COBOL, DOS PL/I, and Assembler language programs are executable under CMS/DOS.

All of the CP and CMS online debugging and testing facilities (such as the CP ADSTOP and STORE commands and the CMS DEBUG environment) are supported in the CMS/DOS environment. Also, CP disk error recording and recovery is supported in CMS/DOS.

With its support of a CMS/DOS environment, CMS becomes an important tool for DOS/VS application program development. Because CMS/DOS was designed as a DOS/VS program development tool, it assumes that a DOS/VS system exists, and uses it. The following sections describe what is supported, and what is not.

CMS SUPPORT FOR OS AND DOS VSAM FUNCTIONS

CMS supports interactive program development for OS and DOS programs using VSAM. CMS supports VSAM for OS programs written in VS BASIC, OS/VS COBOL, or OS PL/I programming languages; or DOS programs written in DOS/VS COBOL or DOS PL/I programming languages. CMS does not support VSAM for OS or DOS assembler language programs.

CMS also supports Access Method Services to manipulate OS and DOS VSAM and SAM data sets.

Under CMS, VSAM data sets can span up to nine DASD volumes. CMS does not support VSAM data set sharing; however, CMS already supports the sharing of minidisks or full pack minidisks.

VSAM data sets created in CMS are not in the CMS file format. Therefore, CMS commands currently used to manipulate CMS files cannot be used for VSAM data sets which are read or written in CMS. A VSAM data set created in CMS has a file format that is compatible with OS and DOS VSAM data sets. Thus a VSAM data set created in CMS can later be read or updated by OS or DOS.

Because VSAM data sets in CMS are not a part of the CMS file system, CMS file size, record length, and minidisk size restrictions do not apply. The VSAM data sets are manipulated with Access Method Services programs executed under CMS, instead of with the CMS file system commands. Also, all VSAM minidisks and full packs used in CMS must be initialized with the IBCDASDI program; the CMS FORMAT command must not be used.

CMS supports VSAM control blocks with the GENCB, MODCB, TESTCB, and SHOWCB macros.

In its support of VSAM data sets, CMS uses RPS (rotational position sensing) wherever possible. CMS does not use RPS for 2314/2319 devices, or for 3340 devices that do not have the feature.

Hardware Devices Supported

Because CMS support of VSAM data sets is based on DOS/VS VSAM and DOS/VS Access Method Services, only disks supported by DOS/VS can be used for VSAM data sets in CMS. These disks are:

- IBM 2314 Direct Access Storage Facility
- IBM 2319 Disk Storage
- IBM 3330 Disk Storage, Models 1 and 2
- IBM 3330 Disk Storage, Model 11
- IBM 3340 Direct Access Storage Facility
- IBM 3344 Direct Access Storage
- IBM 3350 Direct Access Storage

CMS Method of Operation and Program Organization

This section contains the following information:

- Initialization of the CMS Virtual Machine Environment
- Processing and Executing CMS Files
- Handling I/O Operations
- Simulating Non-CMS Operating Environments
- Performing Miscellaneous CMS Functions

The CMS description is in two parts. The first part contains figures showing the functional organization of CMS. The second part contains general information about the internal structure of CMS programs and their interaction with one another.

CMS program organization is in two figures. Figure 8 is an overview of the functional areas of CMS. Each block is numbered and corresponds to a more detailed outline of the function found in Figure 9.

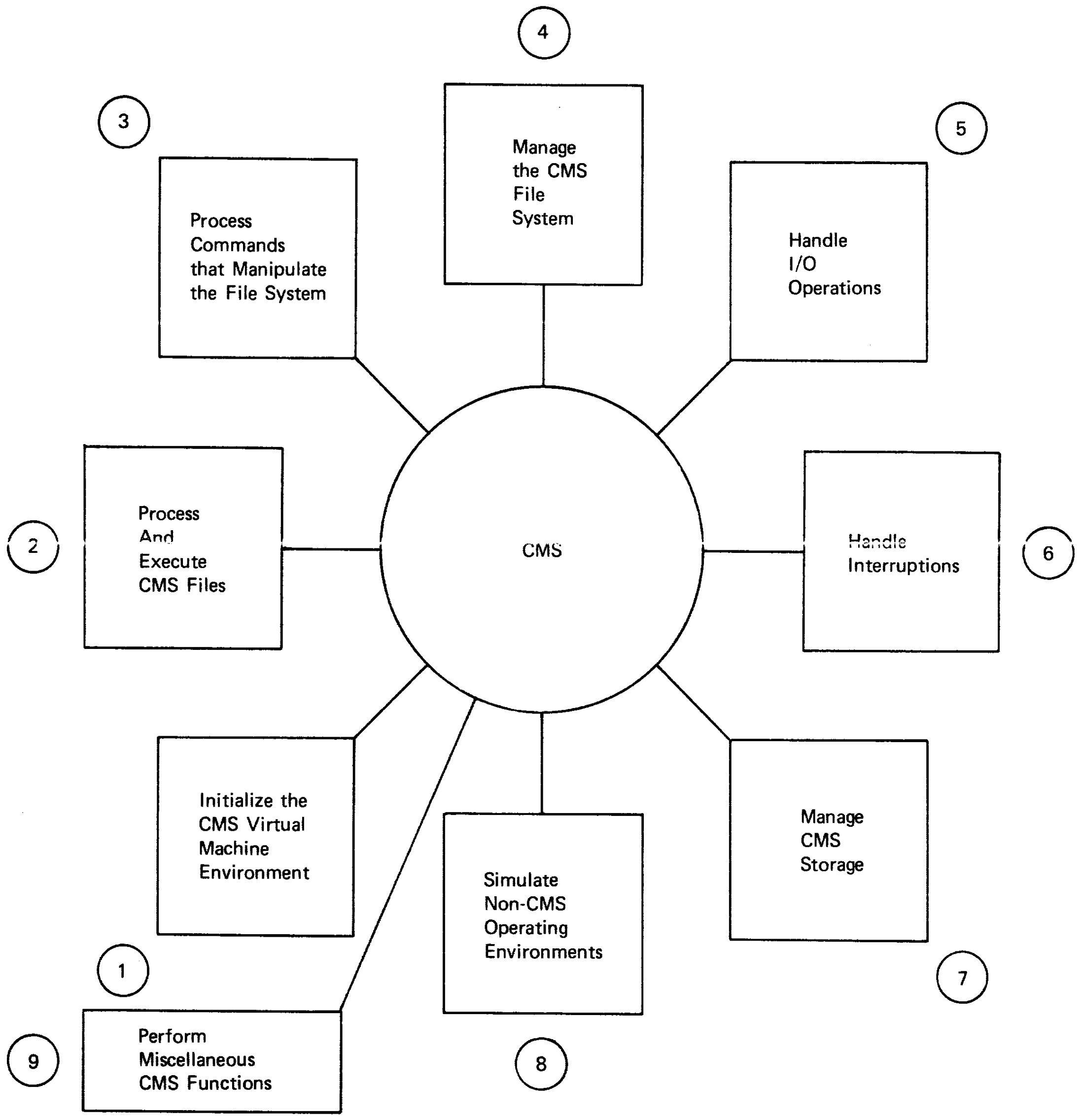
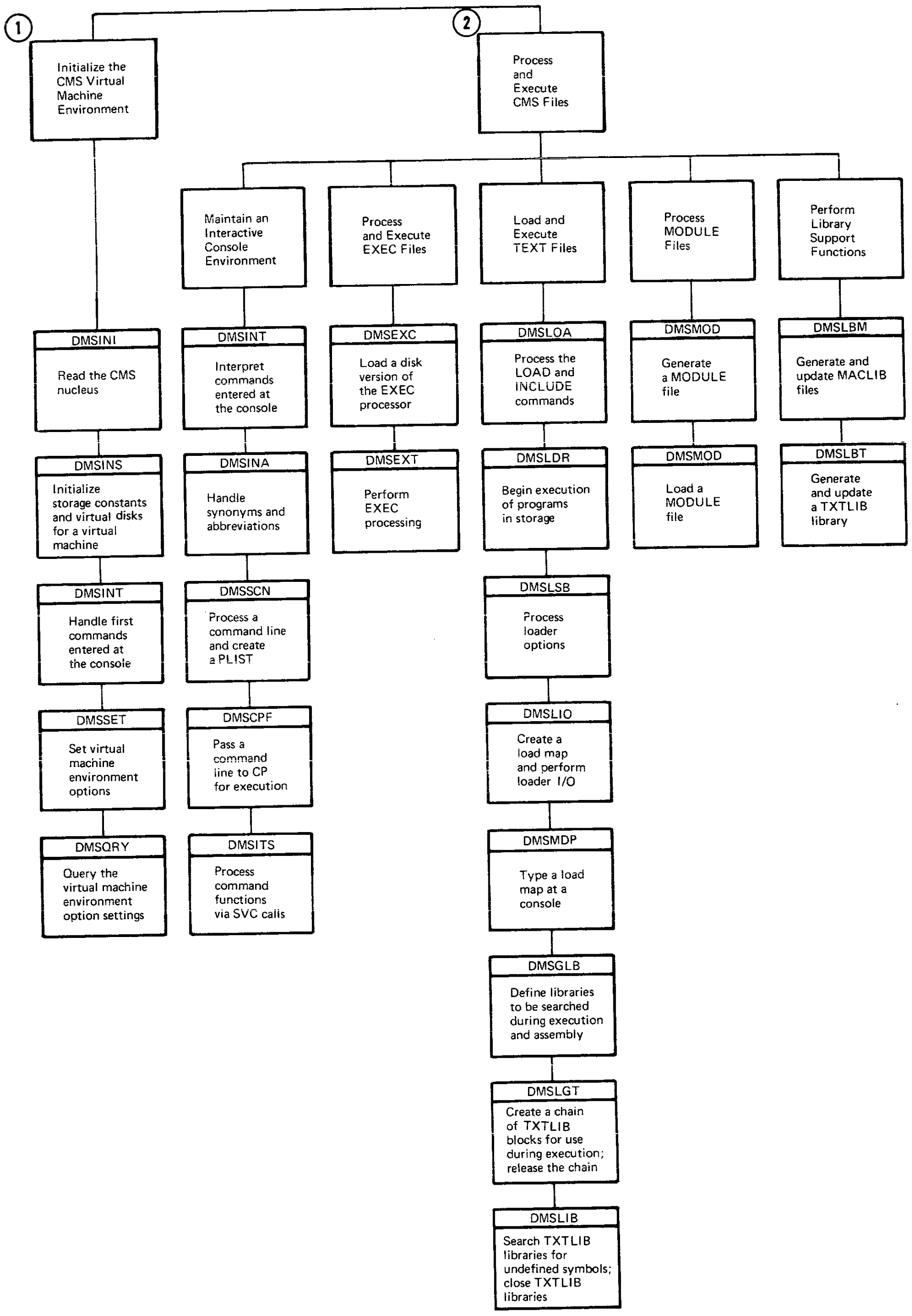

4
Manage the CMS File System
3
Process Commands that Manipulate the File System
5
Handle I/O Operations
2
Process And Execute CMS Files
CMS
Handle Interruptions
6
Initialize the CMS Virtual Machine Environment
Simulate Non-CMS Operating Environments
Manage CMS Storage
1
7
9
Perform Miscellaneous CMS Functions
8

Figure 8. An Overview of the Functional Areas of CMS



1
Initialize the CMS Virtual Machine Environment
2
Process and Execute CMS Files
Maintain an Interactive Console Environment
Process and Execute EXEC Files
Load and Execute TEXT Files
Process MODULE Files
Perform Library Support Functions
DMSINI
Read the CMS nucleus
DMSINS
Initialize storage constants and virtual disks for a virtual machine
DMSINT
Handle first commands entered at the console
DMSSET
Set virtual machine environment options
DMSQRY
Query the virtual machine environment option settings
DMSINT
Interpret commands entered at the console
DMSINA
Handle synonyms and abbreviations
DMSSCN
Process a command line and create a PLIST
DMSCPF
Pass a command line to CP for execution
DMSITS
Process command functions via SVC calls
DMSEXC
Load a disk version of the EXEC processor
DMSEXT
Perform EXEC processing
DMSLOA
Process the LOAD and INCLUDE commands
DMSLDR
Begin execution of programs in storage
DMSLSB
Process loader options
DMSLIO
Create a load map and perform loader I/O
DMSMDP
Type a load map at a console
DMSGLB
Define libraries to be searched during execution and assembly
DMSLGT
Create a chain of TXTLIB blocks for use during execution; release the chain
DMSLIB
Search TXTLIB libraries for undefined symbols; close TXTLIB libraries
DMSMOD
Generate a MODULE file
DMSMOD
Load a MODULE file
DMSLBM
Generate and update MACLIB files
DMSLBT
Generate and update a TXTLIB library

Figure 9. Details of CMS System Functions and the Routines that Perform Them (Part 1 of 4)

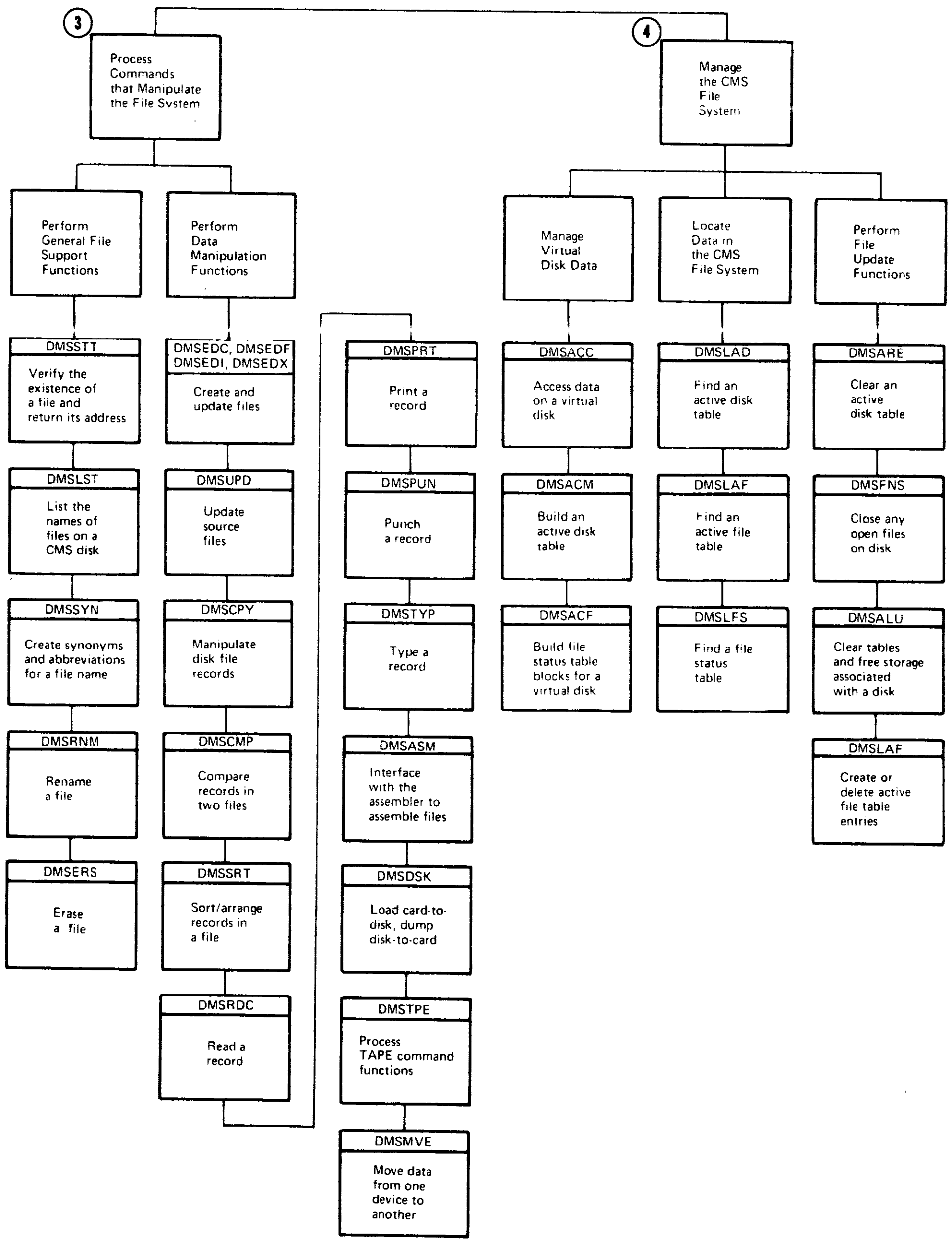

3
Process Commands that Manipulate the File System
4
Manage the CMS File System
Perform General File Support Functions
Perform Data Manipulation Functions
Manage Virtual Disk Data
Locate Data in the CMS File System
Perform File Update Functions
DMSSTT
Verify the existence of a file and return its address
DMSLST
List the names of files on a CMS disk
DMSSYN
Create synonyms and abbreviations for a file name
DMSRNM
Rename a file
DMSERS
Erase a file
DMSEDC, DMSEDF DMSEDI, DMSEDX
Create and update files
DMSUPD
Update source files
DMSCPY
Manipulate disk file records
DMSCMP
Compare records in two files
DMSSRT
Sort/arrange records in a file
DMSRDC
Read a record
DMSPRT
Print a record
DMSPUN
Punch a record
DMSTYP
Type a record
DMSASM
Interface with the assembler to assemble files
DMSDSK
Load card-to-disk, dump disk-to-card
DMSTPE
Process TAPE command functions
DMSMVE
Move data from one device to another
DMSACC
Access data on a virtual disk
DMSACM
Build an active disk table
DMSACF
Build file status table blocks for a virtual disk
DMSLAD
Find an active disk table
DMSLAF
Find an active file table
DMSLFS
Find a file status table
DMSARE
Clear an active disk table
DMSFNS
Close any open files on disk
DMSALU
Clear tables and free storage associated with a disk
DMSLAF
Create or delete active file table entries

Figure 9. Details of CMS System Functions and the Routines that Perform Them (Part 2 of 4)

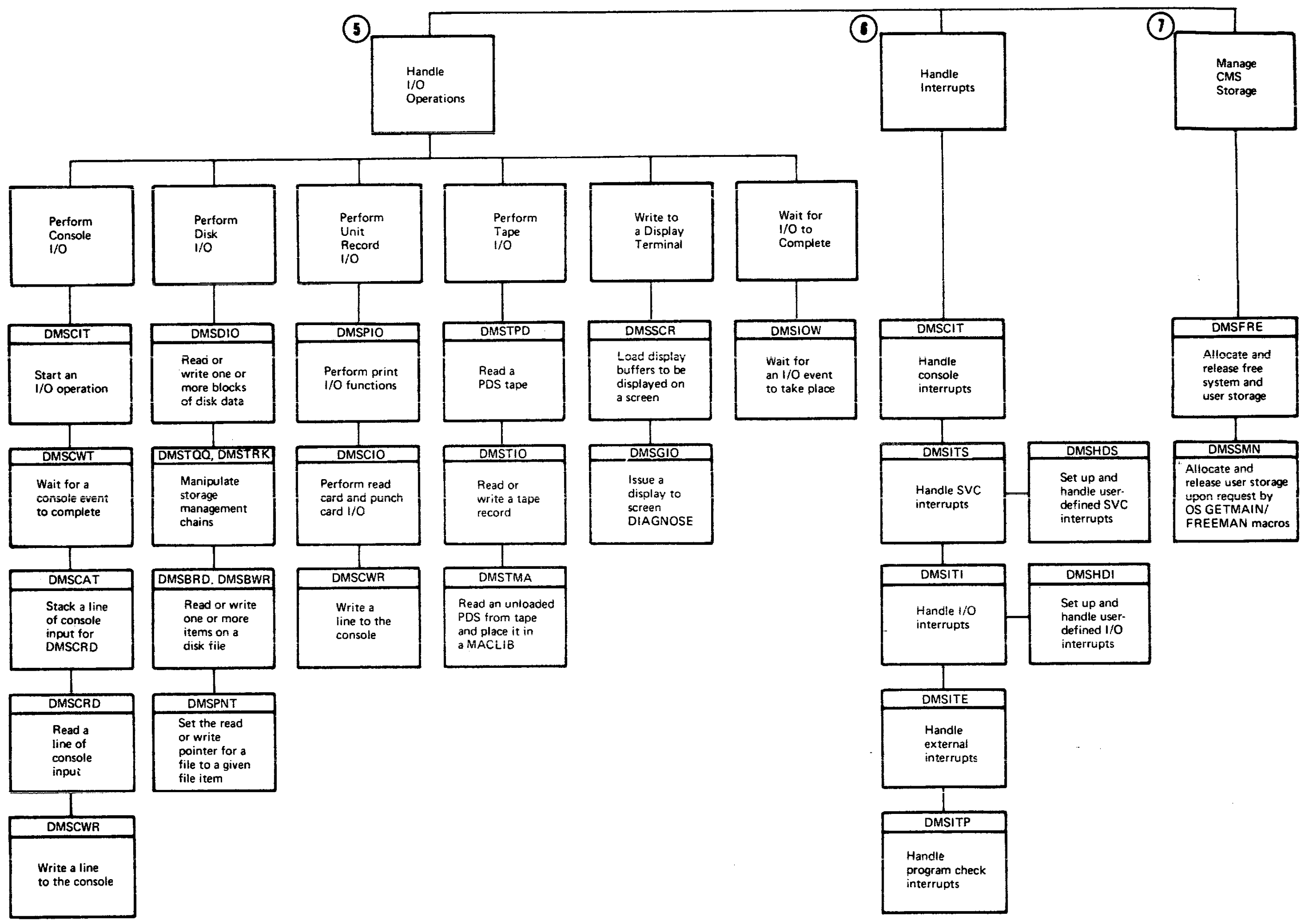
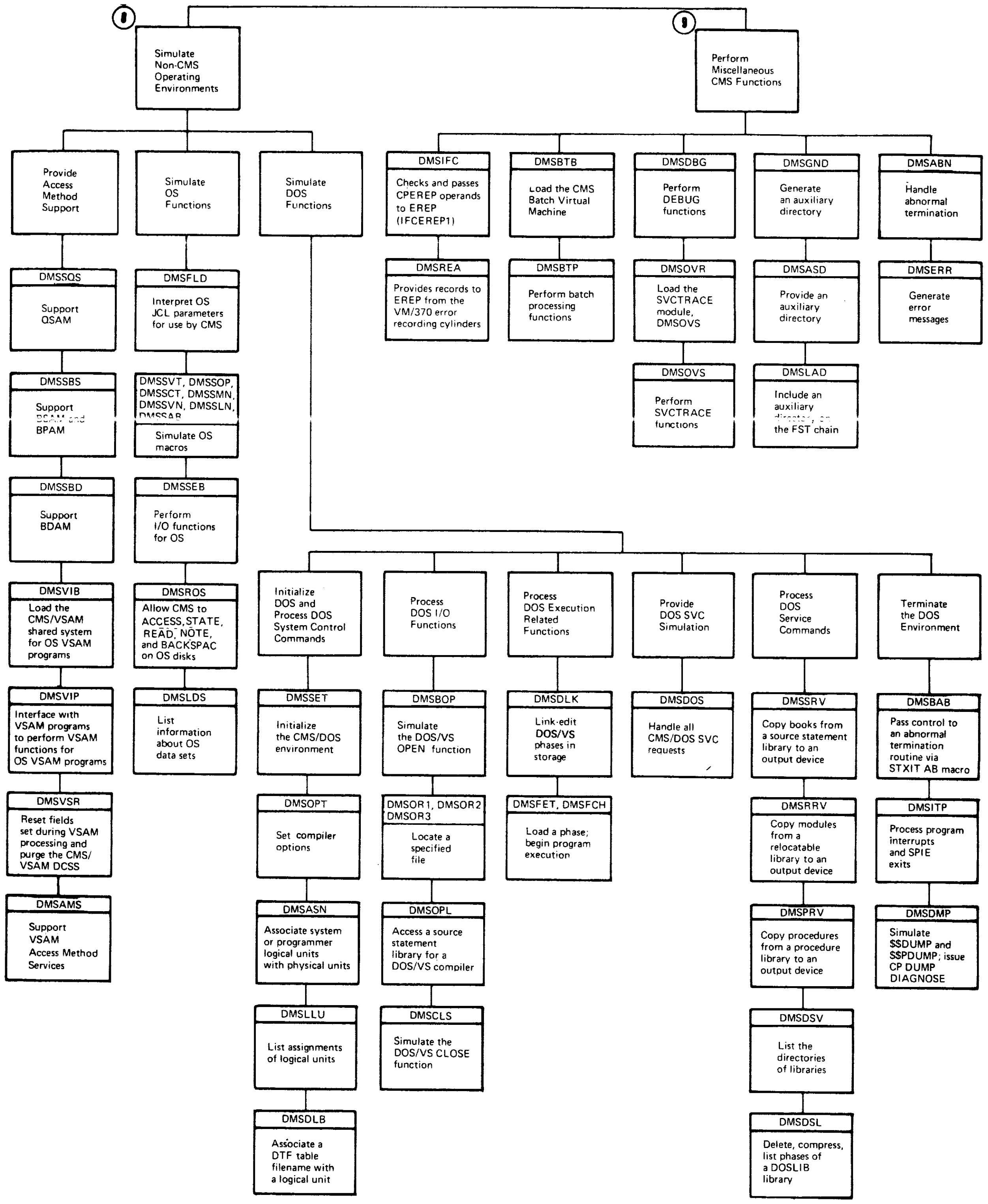

5
Handle I/O Operations
6
Handle Interrupts
7
Manage CMS Storage
Perform Console I/O
Perform Disk I/O
Perform Unit Record I/O
Perform Tape I/O
Write to a Display Terminal
Wait for I/O to Complete
DMSCIT Start an I/O operation
DMSCWT Wait for a console event to complete
DMSCAT Stack a line of console input for DMSCRD
DMSCRD Read a line of console input
DMSCWR Write a line to the console
DMSDIO Read or write one or more blocks of disk data
DMSTQQ, DMSTRK Manipulate storage management chains
DMSBRD, DMSBWR Read or write one or more items on a disk file
DMSPNT Set the read or write pointer for a file to a given file item
DMSPIO Perform print I/O functions
DMSCIO Perform read card and punch card I/O
DMSCWR Write a line to the console
DMSTPD Read a PDS tape
DMSTIO Read or write a tape record
DMSTMA Read an unloaded PDS from tape and place it in a MACLIB
DMSSCR Load display buffers to be displayed on a screen
DMSGIO Issue a display to screen DIAGNOSE
DMSIOW Wait for an I/O event to take place
DMSCIT Handle console interrupts
DMSITS Handle SVC interrupts
DMSHDS Set up and handle user-defined SVC interrupts
DMSITI Handle I/O interrupts
DMSHDI Set up and handle user-defined I/O interrupts
DMSITE Handle external interrupts
DMSITP Handle program check interrupts
DMSFRE Allocate and release free system and user storage
DMSSMN Allocate and release user storage upon request by OS GETMAIN/FREEMAN macros

Figure 9. Details of CMS System Functions and the Routines that Perform Them (Part 3 of 4)



Simulate Non-CMS Operating Environments
Perform Miscellaneous CMS Functions
Provide Access Method Support
Simulate OS Functions
Simulate DOS Functions
DMSIFC Checks and passes CPEREP operands to EREP (IFCEREP1)
DMSBTB Load the CMS Batch Virtual Machine
DMSDBG Perform DEBUG functions
DMSGND Generate an auxiliary directory
DMSABN Handle abnormal termination
DMSREA Provides records to EREP from the VM/370 error recording cylinders
DMSBTP Perform batch processing functions
DMSOVR Load the SVCTRACE module, DMSOVS
DMSASD Provide an auxiliary directory
DMSERR Generate error messages
DMSOVS Perform SVCTRACE functions
DMSLAD Include an auxiliary the FST chain
DMSSQS Support QSAM
DMSFLD Interpret OS JCL parameters for use by CMS
DMSSBS Support BPAM
DMSSVT, DMSSOP, DMSSCT, DMSSMN, DMSSVN, DMSSLN, DMSSAB Simulate OS macros
DMSSBD Support BDAM
DMSSEB Perform I/O functions for OS
DMSVIB Load the CMS/VSAM shared system for OS VSAM programs
DMSROS Allow CMS to ACCESS,STATE, READ, NOTE, and BACKSPAC on OS disks
DMSVIP Interface with VSAM programs to perform VSAM functions for OS VSAM programs
DMSLDS List information about OS data sets
DMSVSR Reset fields set during VSAM processing and purge the CMS/ VSAM DCSS
DMSAMS Support VSAM Access Method Services
Initialize DOS and Process DOS System Control Commands
Process DOS I/O Functions
Process DOS Execution Related Functions
Provide DOS SVC Simulation
Process DOS Service Commands
Terminate the DOS Environment
DMSSET Initialize the CMS/DOS environment
DMSBOP Simulate the DOS/VS OPEN function
DMSDLK Link-edit DOS/VS phases in storage
DMSDOS Handle all CMS/DOS SVC requests
DMSSRV Copy books from a source statement library to an output device
DMSBAB Pass control to an abnormal termination routine via STXIT AB macro
DMSOPT Set compiler options
DMSOR1, DMSOR2 DMSOR3 Locate a specified file
DMSFET, DMSFCH Load a phase; begin program execution
DMSRRV Copy modules from a relocatable library to an output device
DMSITP Process program interrupts and SPIE exits
DMSASN Associate system or programmer logical units with physical units
DMSOPL Access a source statement library for a DOS/VS compiler
DMSPRV Copy procedures from a procedure library to an output device
DMSDMP Simulate $$DUMP and $$PDUMP; issue CP DUMP DIAGNOSE
DMSLLU List assignments of logical units
DMSCLS Simulate the DOS/VS CLOSE function
DMSDSV List the directories of libraries
DMSDLB Associate a DTF table filename with a logical unit
DMSDSL Delete, compress, list phases of a DOSLIB library

Figure 9. Details of CMS System Functions and the Routines that Perform Them (Part 4 of 4)

Initialization of the CMS Virtual Machine Environment

There are four steps involved in initializing a CMS virtual machine:

- Processing the IPL command for a virtual card reader.
- Processing the IPL command for a disk device or a named or saved system.
- Processing the first command line entered at the CMS virtual console.
- Setting up the options for the virtual machine operating environment.

DMSINI and DMSINS are the two routines that are mainly responsible for the one-time initialization process in which the virtual card reader is initial program loaded. DMSINI also handles the IPL process when a named or saved system is loaded. The CMS command interpreter, DMSINT, processes the first line entered from the console as a special case; the processing performed by this code is a part of the initialization process. DMSSET sets up the user-specified virtual machine environment features; DMSQRY allows the user to query the status of these settings.

Initialization: Loading a CMS Virtual Machine from Card Reader

When a virtual card reader is specified by the IPL command, for example 00C, initialization processing begins. Initialization refers to the process of loading from a card reader as opposed to reading a nucleus from a cylinder of a CMS minidisk or reading a named or shared system (description follows).

IPL 00C invokes the CMS module DMSINI, which requests that the operator enter information such as the address of the DASD where the nucleus is to be written, the cylinder address where the write operation is to begin, and which version of CMS is to be written (if there is more than one to choose from).

When all questions are answered, the requested nucleus is written to the DASD.

Once written on the DASD, a copy of the nucleus is read into virtual machine storage. One track at a time is read from the disk-resident nucleus into virtual storage. DMSINS is then invoked to initialize storage constants and to set up the disks and storage space required by this virtual machine.

DMSINS performs three general functions:

- Initializes storage constants and system tables.
- Processes IPL command line parameters (SEG= and BATCH).
- Initializes for OS SVC processing, in the case where a saved segment is not available for use in processing OS simulation requests.

INITIALIZES STORAGE CONTENTS AND SYSTEM TABLES

<u>DMSINS</u>
Saves the address of this virtual machine in NUCON.

<u>DMSLAD</u>
Locates and returns the address of the ADT for this virtual machine.

<u>DMSFRE</u>
Allocates free storage to be used during initialization.

<u>DMSFRE</u>
Allocates all low free storage so that the system status table (SSTAT) will be built in high free storage.

<u>DMSACM</u>
Reads the S-disk ADT entry and builds the SSTAT.

<u>DMSFRE</u>
Releases the low free storage allocated above (to force SSTAT into high storage) so that it can be used again.

<u>DMSINS</u>
Stores the address of SSTAT into ASSTAT and ADTFDA in NUCON.

<u>DMSALU</u>
Sorts the entries in the SSTAT.

PROCESSES IPL COMMAND LINE PARAMETERS

<u>DMSINS</u>
| Checks for parameters BATCH, and SEG=, or AUTOCR. If BATCH is specified, DMSINS sets the flag BATFLAGS. If SEG= is specified, DMSINS loops through again to read the segment name. At this point, all the parameters on the command line have been scanned.

If SEG= is specified, the DIAGNOSE 64 FINDSYS function is issued to determine whether the segment specified on the command line exists. If it does, the DCSSAVAL flag is temporarily set.

| If AUTOCR is specified, a local flag is set so that the subsequent
| console read may be bypassed and the null line input simulated. This
| action causes a PROFILE EXEC to be executed.

<u>DMSINS</u>
Issues DIAGNOSE 24 to obtain the device type of the console.

<u>DMSCWR</u>
Writes the system id message to the console.

<u>DMSCRD</u>
Reads the IPL command line from the console.

<u>DMSSCN</u>
Puts the IPL command line in PLIST format.

<u>DMSINS</u>
If the FINDSYS DIAGNOSE validated the segment name specified on the IPL command line, DMSINS issues a DIAGNOSE 64 SAVESYS function for that segment.

DMSINS
Clears DCSSAVAL and ensures that all the parameters on the command line are valid; branches back to label INITLOOP to reprocess for the segment just saved.

DMSINS
If BATCH is specified, sets BATFLAGS and BATFLAG2 in NUCON. Saves the name of the BATCH saved system in SYSNAME in NUCON.

DMSACC
Issues ACCESS 195 A to access the batch virtual machine A-disk.

DMSINS
Issues DIAGNOSE 60 to get the size of the virtual machine; sets up enough storage for this virtual machine.

DMSINS
If the DCSSAVAL flag is set, sees if the size of the CMSSEG segment overlaps the size of the virtual machine. If this is the case, DMSINS sets the flag DCSSOVLP and continues the initialization procedure for a CMS virtual machine running without the use of the CMSSEG segment, that is, performs time-of-day processing and OS initialization.

If the CMSSEG segment can be used, DMSINS issues the DIAGNOSE 64 LOADSYS function as the final check to see if the segment is usable. If the segment is loaded successfully, it can be used whenever one of the functions contained in it is requested. Because it is not required immediately, DMSINS issues the DIAGNOSE 64 PURGESYS function to purge the segment.

If the segment cannot be successfully loaded, DMSINS turns off the DCSSAVAL flag.

INITIALIZE OS SVC-HANDLING WITHOUT THE USE OF THE CMSSEG SEGMENT

DMSINS
Checks for the availability of CMSSEG.

DMSSTT
Finds and returns the address of DMSSVT, the CMS OS SVC-handler.

DMSFRE
Acquires enough free storage to contain DMSSVT.

DMSLOA
Loads DMSSVT.

DMSINS
Sets the flag DCSSVTLD.

DMSINS
If the BATCH virtual machine is not being loaded, determines whether there is a PROFILE EXEC or a first command line to be handled. If so, issues SVC 202's to process these commands and passes control to DMSINT, the CMS console manager.

DMSACC
If the BATCH virtual machine is being initial program loaded, accesses the D-disk and passes control to DMSINT, the console manager.

Initializing a Named or Saved Systems

A named system is a copy of the nucleus that has been saved and named with the CP SAVESYS command. It is faster to IPL a named system than to IPL by disk address because CP maintains the named system in page format instead of CMS disk format. That is, the saved system is on disk in 4096-byte blocks instead of 800-byte blocks. The initialization of a saved system is also faster because the SSTAT is already built.

The shared system is a variant of the saved system. In the shared system, reentrant portions of the nucleus are placed in storage pages that are available to all users of the shared system. Each user has his own copy of nonreentrant portions of the nucleus. The shared pages are protected by CP, and may not be altered by any virtual machine.

During DMSINI processing, the virtual machine operator is asked if the nucleus must be written (via message DMSINI607R). If the operator answers no, control passes directly to DMSINS to initialize the named or saved system specified by the operator in his answer to message DMSINI606R.

Handling the First Command Line Passed to CMS

DMSINT, the CMS console manager, contains the code to handle commands stacked by module DMSINS during initialization processing. DMSINT checks for the presence of a stacked command line, and if there is one to process, processes it just as it would a command entered during a terminal session. That is, DMSINT calls the WAITREAD subroutine and issues an SVC 202 to execute the command. When first command processing completes, DMSINT receives control to handle commands entered at the console for the duration of the session.

Setting and Querying Virtual Machine Environment Options

DMSSET sets up the virtual machine environment options, as outlined in the publication _VM/370 CMS Command and Macro Reference_. DMSQRY displays these settings at the user console. Both of these modules are structured and relatively easy to follow, except for some sections of DMSSET.

DMSSET: SET DOS ON (VSAM) PROCESSING

DMSSET
(label DOS) If a disk mode is specified on the command line, ensure that it is valid.

DMSLAD
If the disk mode specified is valid, locates and returns the address of the disk.

DMSSET
Issues DIAGNOSE 64 FINDSYS to locate the CMSDOS segment. If the segment is not already loaded, issues DIAGNOSE 64 LOADSYS to load it.

DMSSET
Sets up the $$B-transient area for use by DOS routines.

DMSSET
If SET DOS OFF has been specified, issues the DIAGNOSE 64 PURGESYS function for the CMSDOS segment and, if VSAM has been loaded, for the CMSVSAM segment.

DMSSET: SET SYSNAME PROCESSING

DMSSET
Determines whether the name of the CMSSEG segment is being changed.

DMSSET
Determines whether NONSHARE is specified. If so, the segment may be loaded and kept. If NONSHARE is not specified, the segment is purged, because it is needed only on demand.

DMSSET
Once a new name is placed in the SYSNAMES table replacing CMSSEG, the DIAGNOSE 64 FINDSYS function is issued to determine whether the new name has been entered correctly. If the FINDSYS is successful, the size of the virtual machine is compared to beginning address of the segment to determine whether the segment overlays virtual machine storage.

DMSSET
If the segment can be used (i.e. does not overlay the virtual machine storage) the DIAGNOSE 64 LOADSYS function is performed. If the LOADSYS executes successfully, control passes to DMSINT, where the segment is purged (because it is only needed on demand).

Processing and Executing CMS Files

As shown in Part 1 of Figure 9, the five general topics form the category "Process and Execute CMS Files." Two of these topics are discussed in this section: "Maintaining an Interactive Console Environment" and "Loading and Executing TEXT files."

Maintaining an Interactive Console Environment

Two levels of information are discussed in the following section. The first level is a general discussion of how CMS maintains an interactive console environment. The second level is a more detailed discussion of the methods of operation mainly responsible for this function.

Console Management and Command Handling in CMS

There are two major functions concerned with maintaining an interactive terminal environment for CMS: console management and command processing. The CMS module that manages the virtual machine console is DMSINT. The module responsible for command processing is DMSITS. Many CMS modules are called in support of these two functions but the modules in the following list are primarily responsible for supporting the functions:

DMSCRD
   Reads a line from the console.

DMSCWR
   Writes a line to the console.

DMSSCN
   Converts a command line to PLIST format.

DMSINA
   Converts abbreviated commands to their full names.

DMSCPF
   Passes a command line to CP for execution.

Maintaining an Interactive Command/Response Session

Three main lines of control maintain the continuity for an interactive CMS session: (1) handling of commands passed to DMSINT by the initialization module, DMSINS (2) handling of commands entered at the console during a session, and (3) handling of commands entered as subset commands. The following lists show the main logic paths for first two functions.

EXECUTE COMMANDS PASSED VIA DMSINS

DMSINT
On entry from DMSINA, processes any commands passed via the console read put on the user's console by that routine; that is processes any commands the user stacks on the line as the first read that DMSINT processes. In handling the first read, if that read is null, control passes to the main loop of the program, which is described in the following section.

DMSINM
Get the current time.

DMSCRD
Branch to the waitread subroutine to read a command line at the console.

DMSSCN
Waitread then calls DMSSCN to convert the line just read into plist format. Once converted to plist format, an SVC 202 is issued (at label INIT1A) to execute the function. This cycle is repeated until all stacked commands are executed.

DMSFNS
When command execution completes, calls DMSFNS (at label UPDAT) to close any files that may have remained open during the command processing.

DMSVSR
Ensures that any fields set by VSAM processing are reset for CMS. Also ensures that the VSAM discontiguous shared segment is purged.

DMSINT
Sets up an appropriate status message (CMS, CMS SUBSET, CMS/DOS, etc.).

DMSCWR
Writes the status message to the console.

HANDLE COMMANDS ENTERED DURING A CMS TERMINAL SESSION

DMSINT
Branches (from label INLOOP2) to the waitread subroutine to read a line entered at the console.

DMSCRD
Reads a line entered at the console (subroutine waitread).

DMSSCN
Converts the command line to PLIST format (subroutine waitread).

DMSINT
Determines whether the command line is a null line or a comment.

DMSLFS
If the command line is neither a command line nor a comment, determines whether the command is an EXEC file.

DMSINA (ABBREV)
Determines whether the command is an abbreviation and, if it is, returns its full name.

DMSITS
Passes the command line to DMSITS via an SVC 202. DMSITS is the CMS SVC handler. For a detailed description of the SVC handler, see "Method of Operation for DMSITS."

DMSCPF
If the command could not be executed by the SVC handler, passes the command to CP to see if CP can execute it.

DMSFNS
On return from processing the command line (label UPDAT), closes any files that may have been opened during processing.

DMSSMN
Resets any flags or fields that may have been set during OS processing.

DMSVSR
Ensures that any fields set for VSAM processing are reset for CMS. Also ensures that the VSAM discontiguous shared segment is purged.

DMSINT
When the command line has been successfully executed, builds a CMS ready message for the user (label PRNREADY).

DMSCWR
Writes the ready message to the console.

DMSINT
Returns control to DMSINT at label INLOOP2 to continue monitoring the CMS terminal session.

Method of Operation for DMSINT

DMSINT, the console manager, maintains the continuity of operation of the CMS command environment. The main control loop of DMSINT is initiated by a call to DMSCRD to get the next command. When the command is entered, DMSINT calls DMSINM to initialize the CPU time for the new command and then puts it in standard parameter list form by calling the scan function program DMSSCN. After calling DMSSCN, DMSINT checks to see if an EXEC filetype exists with a filename of the typed-in command. (For example, if ABC was typed in, it checks to see if ABC EXEC exists.) If the EXEC file does exist, DMSINT adjusts register 1 to point to the same command as set up by DMSSCN, but preceded by CL8'EXEC', and then issues an SVC 202 to call the corresponding EXEC procedure ('ABC EXEC' in the example).

If no such EXEC file exists for the first word typed in, DMSINT makes a further check using the CMS abbreviation-check routine, DMSINA. If, for example, the first word typed in had been 'E', DMSINT looks up 'E' via the DMSINA routine. If an equivalent is found for 'E', DMSINT looks for an EXEC file with the name of the equivalent word (for example, EDIT EXEC); if such a file is found, DMSINT adjusts register 1 as described above to call EXEC and substitutes the equivalent word, EDIT, for the first word typed in. Thus, if 'E' is a valid abbreviation for 'EDIT' and the user has an EXEC file called EDIT EXEC, he invokes this when he merely types in 'E' from the terminal.

If no EXEC file is found either for the entered command name or for any equivalent found by DMSINA, DMSINT leaves the terminal command as processed by DMSSCN and then issues an SVC 202 to pass control to DMSITS which, in turn, passes control to the appropriate command program.

When the command terminates execution, or if DMSITS cannot execute it, the return code is passed in register 15.

A zero return code indicates successful completion of the command.

A positive return code indicates that the command was completed, but with an apparent error; and a negative code returned by DMSITS indicates that the typed in command could not be found or executed at all.

In the last case, DMSINT assumes that the command is a CP command and issues a DIAGNOSE instruction to pass the command line to the CP environment. If the command is not a CP command, DMSINT calls DMSCWR to type a message indicating that the command is unknown and the main control loop of DMSINT is entered at the beginning.

If the return code from DMSITS is positive or zero, DMSINT saves the return code briefly and calls module DMSAUD to update the Master File Directory (MFD) on the user's appropriate user's disk. DMSINT also frees the TXTLIB chain and releases pages of storage if required.

After updating the master file directory, DMSINT checks the return code that was passed back. If the code is zero, DMSINT types a ready message and the processor time used by the given command. Control is passed to the beginning of the main control loop of DMSINT. If the return code is positive, an error message is typed, along with the processor time used. The command caused the typing of an error message of the format: DMSxxxnnnt 'text' where DMSxxx is the module name, nnn is the message identification number, t is the message type, and 'text' is the message explaining the error. Control is then passed to the beginning of the main control loop.

Method of Operation for DMSITS

DMSITS (INTSVC) is the CMS system SVC handling routine. Since CMS is SVC driven, the SVC interruption processor is more complex than the other interruption processors.

The general operation of DMSITS is as follows:

1. The SVC new PSW (low-storage location X'60') contains, in the address field, the address of DMSITS1. Thus, the DMSITS routine is entered whenever a supervisor call is executed.

2. DMSITS allocates a system and user save area, as described below. The user save area is a register save area used by the routine, which is invoked later as a result of the SVC call.

3. The called routine is invoked.

4. Upon return from the called routine, the save areas are deallocated.

5. Control is returned to the caller (the routine which originally made the SVC call).

The following expands upon various features of the general operation that has just been described.

TYPES OF SVCS AND LINKAGE CONVENTIONS

The types of SVC calls recognized by DMSITS, and the linkage conventions for each are as follows:

SVC 201: When a called routine returns control to DMSITS, the user storage key may be in the PSW. Because the called routine may also have turned on the problem bit in the PSW, the most convenient way for DMSITS to restore the system PSW is to cause another interruption, rather than to attempt the privileged Load PSW instruction. DMSITS does this by issuing SVC 201, which causes a recursive entry into DMSITS. DMSITS determines if the interruption was caused by SVC 201, and if so, determines if the SVC 201 was from within DMSITS. If both conditions are met, control returns to the instruction following the SVC 201 with a PSW that has the problem bit off and the system key restored.

SVC 202: SVC 202 is the most commonly used SVC in the CMS system. It is used for calling nucleus resident routines and for calling routines written as commands.

A typical coding sequence for an SVC 202 call is the following:

```
LA    R1,PLIST
SVC   202
DC    AL4(ERRADD)
```

Whenever SVC 202 is called, register 1 must point to a parameter list (PLIST). The format of this parameter list depends upon the actual routine or command being called, but the SVC handler examines the first 8 bytes of the list to find the name of the routine or command being called. It searches for the routine or module as described for SVC 201.

The DC AL4(address) following the SVC 202 is optional, and may be omitted if the programmer does not expect any errors to occur in the routine or command being called. DMSITS can determine whether this DC was inserted by examining the byte following the SVC call. If it is nonzero, then it is an instruction; if it is zero, then it is a "DC AL4(address)".

SVC 203: SVC 203 is used by CMS macros to perform various internal system functions. SVC 203 is an SVC call for which no parameter list is provided. An example is DMSFREE, for which the parameters are passed in registers 0 and 1.

A typical sequence for an SVC 203 call follows:

```
SVC   203
DC    H'code'
```

The halfword decimal code following the SVC 203 indicates the specific routine being called. DMSITS examines this halfword code as follows: (1) the absolute value of the code is taken, using an LPR instruction, (2) the first byte of the result is ignored, and the second byte of the resulting halfword is an index into a branch table, (3) the address of the correct routine is loaded, and control is transferred there, as the called routine.

It is possible for the address in the SVC 203 index table to be zero. In this case, the index entry contains an 8-byte routine or command name, which is processed in the same way as the 8-byte name passed in the parameter list passed to SVC 202.

The sign of the halfword code indicates whether the programmer expects an error return; if so, the code is negative: if not, the code is positive. Note that the sign of the halfword code has no effect on determining the routine which is to be called, because DMSITS takes the absolute value of the code to determine the called routine.

Because only the second byte of the absolute value of the code is examined by DMSITS, seven bits (bits 1-7) are available as flags or for other uses. For example, DMSFREE uses these seven bits to indicate such things as conditional requests and variable requests. Therefore, DMSITS considers the codes H'3' and H'259' to be identical, and handles them the same as H'-3' and H'-259', except for error returns.

When an SVC 203 is invoked, DMSITS stores the halfword code into the NUCON location CODE203, so that the called routine can interrogate the seven bits made available to it.

<u>USER-HANDLED SVCs</u>: The programmer may use the HNDSVC macro to specify the address of a routine that processes any SVC call for SVC numbers 0 through 200 and 206 through 255.

If the HNDSVC macro is used, the linkage conventions are as required by the user specified SVC-handling routine.

There is no way to specify a normal or error return from a user-handled SVC routine.

<u>OS MACRO SIMULATION SVC CALLS</u>: CMS supports certain of the SVC calls generated by OS macros, by simulating the effect of these macro calls.

The proper linkages are set up by the OS macro generations. DMSITS does not recognize any way to specify a normal or error return from an OS macro simulation SVC call.

<u>DOS SVC CALLS</u>: All SVC functions supported for CMS/DOS are handled by the CMS module DMSDOS. DMSDOS receives control from DMSITS (the CMS SVC handler) when that routine intercepts a DOS SVC code and finds that the DOSSVC flag in DOSFLAGS is set in NUCON.

DMSDOS acquires the specified SVC code from the OLDPSW field of the current SVC save area. Using this code, DMSDOS computes the address of the routine where the SVC is to be handled.

Many CMS/DOS routines (including DMSDOS) are contained in a discontiguous shared segment (DCSS). Most SVC codes are executed within DMSDOS, but some are in separate modules external to DMSDOS. If the SVC code requested is external to DMSDOS, its address is computed using a table called DCSSTAB; if the code requested is executed within DMSDOS, the table SVCTAB is used to compute the address of the code to handle the SVC.

DOS SVC calls are discussed in more detail in "Simulating a DOS Environment Under CMS" in this section.

<u>INVALID SVC CALLS</u>: There are several types of invalid SVC calls recognized by DMSITS. These are:

- Invalid SVC number. If the SVC number does not fit into any of the classes described above, it is not handled by DMSITS. An error message is displayed at the terminal, and control is returned directly to the caller.

- Invalid routine name in SVC 202 parameter list. If the routine named in the SVC 202 parameter list is invalid or cannot be found, then

DMSITS handles the situation in the same way it handles an error return from a legitimate SVC routine. The error code is -3.

- Invalid SVC 203 code. If an illegal code follows SVC 203, an error message is displayed, and the ABEND routine is called to terminate execution.

SEARCH HIERARCHY FOR SVC 202

When a program issues SVC 202, and passes a routine or command name in the parameter list, DMSITS must search for the specified routine or command. (In the case of SVC 203 with a zero in the table entry for the specified index, the same logic must be applied.)

The search order is as follows:

1. A check is made to see if there is a routine with the specified name currently in the system transient area. If so, then control is transferred there.

2. The system function name table is searched to see if a command by this name is nucleus resident. If successful, control goes to the specified nucleus routine.

3. A search is made for a disk file with the specified name as the filename, and MODULE as the filetype. The search is made in the standard disk search order. If this search is successful, then the specified module is loaded by LOADMOD and control passes to the storage location now occupied by the command.

4. If all searches so far have failed, then DMSINA (ABBREV) is called to see if the specified routine name is a valid system abbreviation for a system command or function. User-defined abbreviations and synonyms are checked at the same time. If this search is successful, then steps 2 through 4 are repeated with the full nonabbreviated name.

5. If all searches fail, then an error code of -3 is forced.

USER AND TRANSIENT PROGRAM AREAS

There are two areas which can hold program modules which are loaded by LOADMOD from the disk. These are called the user program area and the transient program area.

The user program area starts at location X'20000' and extends upward to the loader tables. However, the high-address end of that area can be allocated as free storage by DMSFREE. Generally, all user programs and certain system commands, such as EDIT and COPYFILE, execute in the user program area. Because only one program can be executing in the user program area at one time, unless it is an overlay structure, it is impossible for one program in the user program area to invoke, by means of SVC 202, a module which is also intended to execute the user program area.

The transient program area is two pages, running from location X'E000' to location X'10000'. It provides an area for system commands that may also be invoked from the user program area by means of an SVC

202 call. For example, a program in the user program area may invoke the RENAME command, because this command is loaded into the transient program area.

The transient program area also handles certain OS macro simulation SVC calls. If DMSITS cannot find the address of a supported OS macro simulation SVC handling routine, it calls LOADMOD to load the file DMSSVT module into the transient area, and lets that routine handle the SVC.

A program in the transient program area may not invoke another program intended to execute in the transient program area, including OS macro simulation SVC calls that are handled by DMSSVT. Thus, for example, a program in the transient program area may not invoke the RENAME command. In addition, it may not invoke the OS macro WTO, which generates an SVC 35, which is handled by DMSSVT.

There is one further functional difference between the use of the two program areas. DMSITS starts a program in the user program area so that it is enabled for all interruptions. It starts a program in the transient program area so that it is disabled for all interruptions. Thus, the individual program may have to use the SSM (Set System Mask) instruction to change the current status of its system mask.

CALLED ROUTINE START-UP TABLE

Figures 10 and 11 show how the PSW and registers are set up when the called routine is entered.

Called Type	System Mask	Storage Key	Problem Bit
SVC 202 or 203 - Nuc resident	Disabled	System	Off
SVC 202 or 203 - Transient area MODULE	Disabled	User	Off
SVC 202 or 203 - User Area	Enabled	User	Off
User-handled	Enabled	User	Off
OS - Nuc res	Disabled	System	Off
OS - in DMSSVT	Disabled	System	Off

Figure 10. PSW Fields when Called Routine is Started

RETURNING TO THE CALLER

When the called routine is finished processing it returns control to DMSITS, which then must return control to the caller.

<u>RETURN LOCATION</u>: The return is effected by loading the original SVC old PSW (which was saved at the time DMSITS was first entered), after

Type	0 - 1	2 - 11	12	13	14	15
SVC 202 or 203	Same as caller	Unpredict-able	Address of called routine	User save area	Return address to DMSITS	Address of called routine
Other	Same as caller	Same as caller	Address of called routine	User save area	Return address to DMSITS	Same as caller

Figure 11. Register Contents when Called Routine is Started

possibly modifying the address field. How the address field is modified depends upon the type of SVC call, and on whether the called routine indicated an error return address.

For SVC 202 and 203, the called routine indicates a normal return by means of a zero returned in register 15, and an error return by means of a nonzero in register 15. If the called routine indicates a normal return, then DMSITS makes a normal return to the caller. If the called routine indicates an error return, then DMSITS returns to the caller's error return address, if one was specified, and abnormally terminates if none was specified.

For SVC 202 not followed by "DC AL4(address)", a normal return is made to the instruction following the SVC instruction, and an error return causes an abnormal termination. For SVC 202 followed by "DC AL4(address)", a normal return is made to the instruction following the DC, and an error return is made to the address specified in the DC. In either case, register 15 contains the return code passed by the called routine.

For SVC 203 with a positive halfword code, a normal return is made to the instruction following the halfword code, and an error return causes an abnormal termination. For SVC 203 with a negative halfword code, both normal and error returns are made to the instruction following the halfword code. In any case, register 15 contains the return code passed back by the called routine.

For OS macro simulation SVC calls, and for user-handled SVC calls, no error return is recognized by DMSITS. As a result, DMSITS always returns to the caller by loading the SVC old PSW that was saved when DMSITS was first entered.

<u>REGISTER RESTORATION</u>: Upon entry to DMSITS, all registers are saved as they were when the SVC instruction was first executed. Upon exiting from DMSITS, all registers are restored to the values that were saved at entry.

The exception to this is register 15 for SVC 202 and 203. Upon return to the caller, register 15 contains the value that was in register 15 when the called routine returned to DMSITS after it had completed processing.

SYSTEM AND USER SAVE AREA FORMATS

Whenever an SVC call is made, DMSITS allocates two save areas for that particular SVC call.

DMSITS uses the system save area (DSECT SSAVE) to save the value of the SVC old PSW at the time of the SVC call, the caller's registers at the time of the call, and any other necessary control information. Since SVC calls can be nested, there can be several of these save areas at one time. The system save area is allocated in protected free storage.

The user save area contains (DSECT EXTUAREA) 12 doublewords (24 fullwords), allocated in unprotected free storage. DMSITS does not use this area at all, but simply passes to the called routine a pointer to this area in register 13. Thus, the called routine can use this area as a temporary work area, or as a register save area. There is one user save area for each system save area, and the latter contains a pointer to the former in the USAVEPTR field.

Load and Execute Text Files

The CMS loader consists of a nucleus resident loader (DMSLDR), a file and message handler program (DMSLIO), a library search program (DMSLIB), and other subroutine programs. DMSLDR starts loading at the user first location (AUSRAREA) specified in NUCON or at a user specified location. When performing an INCLUDE function, loading resumes at the next available location after the previous LOAD, INCLUDE, or LOADMOD.

The loader reads in the entire user's program, which consists of one or more control sections, each defined by a type 0 ESD record ("card"). Each control section contains a type 1 ESD card for each entry point and may contain other control cards.

Once the user's program is in storage, the loader begins to search his files for library subprograms called by the program. The loader reads the library subprograms into storage, relocating and linking them as required. To relocate programs, the loader analyzes information on the SLC, ICS, ESD, TXT, and REP cards. To establish linkages, it operates on ESD, and RLD cards. Information for end-of-load transfer of control is provided by the END and LDT cards, the ENTRY control card, START command, or RESET option.

The loader also analyzes the options specified on the LOAD and INCLUDE commands. In response to specified options, the loader can:

- Set the load area to zeros before loading (CLEAR option).
- Load the program at a specified location (ORIGIN option).
- Suppress creation of the load-map file on disk (NOMAP option).
- Suppress the printing of invalid card images in the load map (NOINV option).
- Suppress the printing of REP card images in the load map (NOREP option).
- Load program into "transient area" (ORIGIN TRANS option).
- Suppress TXTLIB search (NOLIBE option).
- Suppress text file search (NOAUTO option).
- Execute the loaded program (START option).

- Type the load map (TYPE option).
- Set the program entry point (RESET option).

During its operation, the loader uses a loader table (REFTBL), and external symbol identification table (ESIDTB), and a location counter (LOCCNT). The loader table contains the names of control sections and entry points, their current location, and the relocation factor. (The relocation factor is the difference between the compiler-assigned address of a control section and the address of the storage location where it is actually loaded.) The ESIDTB contains pointers to the entries in REFTBL for the control section currently being processed by the loader. The loader uses the location counter to determine where the control section is to be loaded. Initially, the loader obtains from the nucleus constant area the address (LOCCNT) of the next location at which to start loading. This value is subsequently incremented by the length indicated on an ESD (type0), END, or ICS card, or it may be reset by an SLC card.

The loader contains a distinct routine for each type of input card. These routines perform calculations using information contained in the nucleus constant area, the location counter, the ESIDTB, the loader table, and the input cards. Other loader routines perform initialization, read cards into storage, handle error conditions, provide disk and typewritten output, search libraries, convert hexadecimal characters to binary, process end-of-file conditions, and begin execution of programs in core.

Following are descriptions of the individual subprocessors with LDR.

SLC CARD ROUTINE

<u>Function</u>
This routine sets the location counter (LOCCT) to the address specified on an SLC card, or to the address assigned (in the REFTBL) to a specified symbolic name.

<u>Entry</u>
The routine is entered at the first instruction when it receives control from the initial and resume loading routine. It is entered at ORG2 whenever a loader routine requires the current address of a symbolic location specified on an SLC card.

<u>Operation</u>
This routine determines which of the following situations exists, and takes the indicated action:

1. The SLC card does not contain an address or a symbolic name. The SLC card routine branches, via BADCRD in the reference table search routine, to the disk and type output routine (DMSLIO), which generates an error message.

2. The SLC card contains an address only. The SLC card routine sets the location counter (LOCCT) to that address and returns to RD, in the initial and resume loading routine, to read another card.

3. The SLC card contains a name only, and there is a reference table entry for that name. The SLC card routine sets LOCCT to the current address of that name (at ORG2) and returns to the initial and resume loading routine to get another card.

4. The SLC card contains a name only, and there is no reference table entry for that name. The SLC card routine branches via ERRSLC to the Disk and Type Output routine (DMSLIO), which generates an error message for that name.

5. The SLC card contains both an address and a name. If there is a REFTBL entry for the name, the sum of the current address of the name and the address specified on the SLC card is placed in LOCCT; control returns to the initial and resume loading routine to get another card. If there is no REFTBL entry for the name, the SLC card routine branches via ERRSLC to the Disk and Type Output routine, which generates an error message for the name.

ICS CARD ROUTINE - C2AE1

Function

This routine establishes a reference table entry for the control-segment name on the ICS card if no entry for that name exists, adjusts the location counter to a fullword boundary, if necessary, and adds the card-specified control-segment length to the location counter if necessary.

Entry

This routine has one entry point, named C2AE1. The routine is entered from the initial and resume loading routine when it finds an ICS card.

Operation

1. The routine begins its operation with a test of card type. If the card being processed is not an ICS card, the routine branches to the ESD card analysis routine; otherwise, processing continues in this routine.

2. The routine tests for a hexadecimal address on the ICS card. If an address is present, the routine links to the DMSLSBA subroutine to convert the address to binary, otherwise the routine branches via BADCRD to the disk and type output routine (DMSLIO).

3. The routine next links to the REFTBL search routine, which determines whether there is a reference table entry for the card-specified control-segment name. If such an entry is found, the REFTBL search routine branches to the initial and resume loading routine; otherwise, the REFTBL search routine places the control-segment name in the reference table, and processing continues.

4. The routine determines whether the card-specified control-segment length is zero or greater than zero. If the length is zero, the routine places the current location counter value in the reference table entry as the control segment's starting address (ORG2), and branches to the initial and resume loading routine. If the length is greater than zero, the routine sets the current location counter value at a fullword boundary address. The routine then places this adjusted current location counter value in the reference table entry, adjusts the location counter by adding the specified control-segment length to it, and branches to RD in the initial and resume loading routine to get another card.

ESD TYPE 0 CARD ROUTINE - C3AA3

<u>Function</u>
This routine creates loader table and ESID table entries for the card-specified control section.

<u>Entry</u>
This routine has one entry point, location C3AA3. The routine is entered from the ESD card analysis routine.

<u>Operation</u>

1. If this is the first section definition, its ESDID is proved.

2. This routine first determines whether a loader table (REFTBL) entry has already been established for the card-specified control section. To do this, the routine links to the REFTBL search routine. The ESD type 0 card routine's subsequent operation depends on whether there already is a REFTBL entry for this control section. If there is such an entry, processing continues with operation 5, below; if there is not, the REFTBL search routine places the name of this control section in REFTBL, and processing continues with operation 3.

3. The routine obtains the card-specified control section length and performs operation 4.

4. The routine links to location C2AJ1 in the ICS card routine and returns to C3AD4 to obtain the current storage address of the control section from the REFTBL entry, inserts the REFTBL entry position (N - where this is the Nth REFTBL entry) in the card-specified ESID table location, and calculates the difference between the current (relocated) address of the control section and its card-specified (assembled) address. This difference is the relocation factor; it is placed in the REFTBL entry for this control section. If previous ESD's have been waiting for this CSECT, a branch is taken to SDDEF, where the waiting elements are processed. A flag is set in the REFTBL entry to indicate a section definition.

5. The entry found in the REFTBL is examined to determine whether it had been defined by a COMMON. If so, it is converted from a COMMON to a CSECT and performs operation 3.

6. If the entry had not been defined previously by an ESD type 0, processing continues at 3.

7. If the entry had been defined previously as other than COMMON, DMSLIO is called via ERRORM to print a warning message, "DUPLICATE IDENTIFIER". The entry in the ESID table is set negative so that the CSECT will be skipped (that is, not loaded) by the TXT and RLD processing routines.

ESD TYPE 1 CARD ROUTINE - ENTESD

<u>Function</u>
This routine establishes a loader table entry for the entry point specified on the ESD card, unless such an entry already exists.

<u>Entry</u>
This routine is entered from the ESD card analysis routine.

Operation

1. Branches and links to REFADR to find loader table entry for first section definition of the text deck saved by the ESD 0 routine.

2. The routine then adds the relocation factor and the address of the ESD found in operation 1 or the address in LOCCNT if an ESD has not yet been encountered. The sum is the current storage address of the entry point.

3. The routine links to the REFTBL search routine to find whether there is already a REFTBL entry for the card-specified entry point name. If such an entry exists, the routine performs operation 4. If there is no entry, the routine performs operation 5.

4. Upon finding a REFTBL entry that has been previously defined for the card-specified name, the routine then compares the REFTBL-specified current storage address with the address computed in operation 2. If the addresses are different, the routine branches and links to the DMSLIO routine (duplicate symbol warning); if the addresses are the same, the routine branches to location RD in the initial and resume loading routine to read another card. Otherwise, it is assumed that the REFTBL entry was created as a result of previously encountered external references to the entry. The DMSLSBC routine is called to resolve the previous external references and adjust the REFTBL entry. The entry point name and address are printed by calling DMSLIO.

5. If there is no REFTBL entry for the card-specified entry point name, the routine makes such an entry and branches to the DMSLIO routine.

ESD TYPE 2 CARD ROUTINE - C3AH1

Function

This routine creates the proper ESID table entry for the card-specified external name and places the name's assigned address (ORG2) in the reference table relocation factor for that name.

Entry

This routine has two entry points: location C3AH1 and location ESD00. Location C3AH1 is entered from the ESD card analysis routine; this occurs when an ESD type 2 card is being processed. Location ESD00 is entered from:

- The ESD card analysis routine, when the card being processed is an ESD type 2, and an absolute loading process is indicated.

- The ESD type 0 card routine and ESD type 1 card routine, as the last operation in each of these routines.

Operation

1. When this routine is entered at location C3AH1, it first links to the REFTBL search routine to determine whether there is a REFTBL entry for the card-specified external name. If none is found, the REFTBL search routine sets the undefined flag for the new loader table entry.

2. The routine resets a possible WEAK EXTRN flag. The routine next places the REFTBL entry's position-key in the ESID table. If the entry has already been defined by means of an ESD type 0, 1, 5, or 6, processing continues at operation 4. Otherwise, it continues at operation 3.

3. The relocated address is placed in the RELFAC entry in the external name's REFTBL entry.

4. The ESD type 2 card routine then determines (at location ESD00) whether there is another entry on the ESD card. If there is another entry, the routine branches to location CA3A1 in the ESD card analysis routine for further processing of this card; otherwise, the routine branches to location RD in the initial and resume loading routine.

Exits
This routine exits to location CA3A1 in the ESD card analysis routine if there is another entry on the ESD card being processed, and exits to location RD in the initial and resume loading routine if the ESD card requires no further processing.

ESD TYPE 4 ROUTINE - PC

Function
This routine makes loader table and ESIDTAB entries for private code CSECT.

Operation
The ESD Type 4 Card Routine:

1. The routine LDRSYM is called to generate a unique character string number of the form 00000001, which is left in the external data area NXTSYM; it is greater in value than previously generated symbol.

2. The CSECT is then processed as a normal type 0 ESD with the above assigned name.

ESD TYPES 5 AND 6 CARD ROUTINE - PRVESD AND COMESD

Function
This routine creates reference table and ESIDTAB entries for common and pseudo-register ESDs.

Operation
The ESD type 5 and 6 card routine:

1. Links to ESIDINC in the ESD type 0 card routine, to update the number of ESIDTB entries.

2. Links to the REFTBL search routine to determine whether a reference table (REFTBL) entry has already been created. If there is no entry, the REFTBL search routine places the name of the item in the REFTBL.

3. If the REFTBL search routine had to create an entry for the item, the ESD type 5 and 6 card routine indexes it in the ESIDTB, enters the length and alignment in the entry, indicates whether it is a PR or common, and branches to ESD00 in the ESD type 2 card routine to determine whether the card contains additional ESD's to be

processed. If the entry is a PR, the ESD type 5 and 6 card routine enters its displacement and length in the REFTBL before branching to ESD00.

4. If the REFTBL already contained an entry, the ESD type 5 and 6 card routine indexes it in the ESIDTB, checks alignment and branches to ESD00.

Note: The PR alignment is coded and placed into the REFTBL. It is an error to encounter more restrictive alignment PR than previously defined. A blank alignment factor is translated to fullword alignment.

ESD TYPE 10 ROUTINE - WEAK EXTRN

The WEAK EXTRN routine calls the search routine to find the EXTRN name in the loader table. If not found, set the WEAK EXTRN flag in the new loader table entry. Exit to ESD00.

TXT CARD ROUTINE - C4AA1

Function
This routine has two functions: address inspection and placing text in storage.

Entry
This routine has three entry points: location C4AA1, which is entered from the ESD card analysis routine, and locations REPENT and APR1, which are entered from the REP card routine for address inspection.

Operation
1. This routine begins its operation with a test of card type. If the card being processed is not a TXT card, the routine branches to the REP card routine; otherwise, processing continues in this routine.

2. The routine then determines how many bytes of text are to be placed in storage, and finds whether the loading process is absolute or relocating. If the loading process is absolute, the routine performs operation 4, below; if relocating, the routine performs operation 3.

3. If the ESIDTB entry was negative, this is a duplicate to CSECT and processing branches to RD. Otherwise, the routine links to the REFADR routine to obtain the relocation factor of the current control segment.

4. The routine then adds the relocation factor (0, if the loading process is absolute) and the card-specified storage address. The result is the address at which the text must be stored. This routine also determines whether the address is such that the text, when loaded starting at that address, overlays the loader or the reference table. If a loader overlay or a reference table overlay is found, the routine branches to the LDRIO routine. If neither condition is detected, the routine proceeds with address inspection.

5. The routine then determines whether an address has already been saved for possible use as the end-of-load branch address. If an address has been saved, the routine performs operation 7; if not, the routine performs operation 6.

6. The routine determines whether the text address is below location 128. If the address is below location 128, it should not be saved for use as a possible end-of-load branch address, and the routine performs operation 7; otherwise the routine saves the address and then performs operation 7.

7. The routine then stores the text at the address specified (absolute or relocated) and branches to location RD in the initial and resume loading routine to read another card.

Exits
The routine exits to two locations, as follows:

1. The routine exits to location RD in the initial and resume loading routine if it is being used to process a TXT card.

2. The routine exits to location APRIL in the REP card routine if it is being used for REP card address inspection.

REP CARD ROUTINE - C4AA3

Function
This routine places text corrections in storage.

Entry
This routine has one entry point, location C4AA3. The routine is entered from the TXT card routine.

Operation
1. This routine begins its operation with a test of card type. If the card being processed is not a REP card, the routine branches to the RLD card routine; otherwise, processing continues in this routine.

2. The routine then links to the HEXB conversion routine to convert the REP card-specified correction address from hexadecimal to binary.

3. The routine then links to the HEXB conversion routine again to convert the REP card-specified ESID from hexadecimal to binary.

4. The routine then determines whether the 2-byte correction being processed is the first such correction on the REP card. If it is the first correction, the routine performs operation 5; otherwise, the routine performs operation 6.

5. When the routine is processing the first correction, it links to location REPENT in the TXT card routine, where the REP card-specified correction address is inspected for loader overlay and for end-of-load branch address saving; in addition, if the loading process is relocating, the relocated address is calculated and checked for reference table overlay. The routine then performs operation 7.

6. When the correction being processed is not the first such correction on the REP card, the routine branches to location APR1 in the TXT card routine for address inspection.

7. The routine then links to the HEXB conversion routine to convert the correction from hexadecimal to binary, places the correction in storage at the absolute (card-specified) or relocated address, and determines whether there is another correction entry on the REP card. If there is another entry, the routine repeats its processing from operation 4, above; otherwise, the routine branches to location RD in the initial and resume loading routine.

Exits

When all the REP-card corrections have been processed, this routine exits to location RD in the initial and resume loading routine.

RLD Card Routine - C5AA1

Function

This routine processes RLD cards, which are produced by the assembler when it encounters address constants within the program being assembled. This routine places the current storage address (absolute or relocated) of a given defined symbol or expression into the storage location indicated by the assembler. The routine must calculate the proper value of the defined symbol or expression and the proper address at which tc store that value.

Entry

This routine has two entry points, locations C5AA1 and PASSTWO.

Operation

1. Location C5AA1 writes each RLD card into a work file (DMSLDR CMSUT1). Exit to RD to process the next card.

   Location PASSTWO reads an RLD card from the work file. At EOF got to C6AB6 to finish this file.

2. The routine uses the relocation header (RH ESID) on the card to obtain the current address (absolute or relocated) of the symbol referred to by the RLD card. This address is found in the relocation factor section of the proper reference table entry. If the RH ESID is 0, the routine branches to the LDRIO routine (invalid ESD).

3. The routine uses the position header (PH ESID) on the card to obtain the relocation factor of the control segment in which the DEFINE CONSTANT assembler instruction occurred. If the PH ESID is 0, the routine branches to BADCRD in the REFTBL search routine (invalid ESID). If the ESIDTAB entry is negative (duplicate CSECT), the RLD entry is skipped.

4. The routine next decrements the card-specified byte count by 4 and tests it for 0. If the count is now 0, the routine branches to location RD in the initial and resume loading routine; otherwise, processing continues in this routine.

5. The routine determines the length, in bytes, of the address constant referred to in the RLD card. This length is specified on the RLD card.

6. The routine then adds the relocation factor obtained in operation 3 (relocation factor of the control segment in which the current address of the symbol must be stored), and the card-specified address. The sum is the current address of the location at which the symbol address must be stored.

7. The routine then computes the arithmetic value (symbol address or expression value) that must be placed in storage at the address calculated in operation 6, above, and places that value at the indicated address. If the value is undefined, the routine branches to location DMSLSBB, where the constant is added to a string of constants that are to be defined later.

8. The routine again decrements the byte count of information on the RLD card and tests the result for zero. If the result is zero, go to operation 2; otherwise, processing continues in this routine.

9. The routine next checks the continuation flag, a part of the data placed on the RLD card by the assembler. If the flag is on, the routine repeats its processing for a new address only; the processing is repeated from operation 4. If the flag is off, the routine repeats its processing for a new symbol; the processing is repeated from operation 2.

Exits

This routine exits to location RD in the initial and resume loading routine.

END CARD ROUTINE - C6AA1

Function

This routine saves the END card address under certain circumstances, and initializes the loader to load another control segment.

Entry

This routine has one entry point, location C6AA1. The routine is entered from the RLD card routine.

Operation

1. This routine begins its operation with a test of card type. If the card being processed is not an END card, the routine branches to the LDT card routine; otherwise, processing continues in this routine.

2. The routine then determines whether the END card contains an address. If the card contains no address, the routine performs operation 7, below; otherwise, the routine performs operation 3.

3. The routine next checks the end-address-saved switch. If this switch is on, an address has already been saved, and the routine performs operation 7. If the switch is off, the routine performs operation 4.

4. The routine determines whether loading is absolute or relocated. If the loading process is absolute, the routine performs operation 6; otherwise, the routine performs operation 5.

5. The routine links to the REFADR routine to obtain the current relocation factor, and adds this factor to the card-specified address.

6. The routine stores the address (absolute or relocated) in area BRAD, for possible use at the end-of-load transfer of control to the problem program.

7. Goes to location PASSTWO (in RLD routine) to process RLD cards.

8. The routine then clears the ESID table, sets the absolute load flag on, and branches to the location specified in a general register (see "Exits").

<u>Exits</u>
This routine exits to the location specified in a general register. This may be either of two locations:

1. Location RD in the initial and resume loading routine. This exit occurs when the END card routine is processing an END card.

2. The location in the LDT card routine that is specified by that routine's linkage to the END card routine. This exit occurs when the LDT card routine entered this routine to clear the ESID table and set the absolute load flag on.

CONTROL CARD ROUTINE - CTLCRD1

<u>Function</u>
This routine handles the ENTRY and LIBRARY control cards.

<u>Entry</u>
This routine has one entry point, location CTLCRD1. The routine is entered from the LDT card routine.

<u>Operations</u>
1. The CMS function SCAN is called to parse the card.

2. If the card is not an ENTRY or LIBRARY card, the routine determines whether the NOINV option (no printing of invalid card images) was specified. If printing is suppressed, control passes to RD in the initial and resume loading routine, where another card is read. If printing is not suppressed, control passes to the disk and type output routine (DMSLIO), where the invalid card image is printed in the load map. If the card is a valid control card, processing continues.

<u>ENTRY Card</u>
3. If the ENTRY name is already defined in REFTBL, its REFTBL address is placed in ENTADR. Otherwise, a new entry is made in REFTBL, indicating an undefined external reference (to be resolved by later input or library search), and this REFTBL entry's address is placed in ENTADR.

4. The control card is printed by calling DMSLIO via CTLCRD; it then exits to RD.

<u>LIBRARY Card</u>
5. Only nonobligatory reference LIBRARY cards are handled; any others are considered invalid.

6. Each entry-point name is individually isolated and is searched for in the REFTBL. If it has already been loaded and defined, nothing is done and the next entry-point name is processed. Otherwise, the nonobligatory bit is set in the flag byte of the REFTBL entry.

7. Processing continues at operation 4.

REFADR ROUTINE (DMSLDRB)

Function

This routine computes the storage address of a given entry in the reference table.

Entry

This routine has one entry point, location REFADR. The routine is entered for several of the routines within the loader.

Operation

1. Checks to see if requested ESDID is zero. If so, uses LOCCNT as requested location; branches to the return location + 44; otherwise continues this routine.

2. The routine first obtains, from the indicated ESID table entry, the position (n) of the given entry within the reference table (where the given entry is the nth REFTBL entry).

3. The routine then multiplies n by 16 (the number of bytes in each REFTBL entry) and subtracts this result from the starting address of the reference table. The starting address of the reference table is held in area TBLREF; this address is the highest address in storage, and the reference table is always built downward from that address.

4. The result of the subtraction in operation 2, above, is the storage address of the given reference table entry. If there is no ESD for the entry, goes to operation 5; otherwise, this routine returns to the location specified by the calling routine.

5. Adds an element to the chain of waiting elements. The element contains the ESD data item information to be resolved when the requested ESDID is encountered.

PRSERCH ROUTINE (DMSLDRD)

Function

This routine compares each reference table entry name with the given name determining (1) whether there is an entry for that name and (2) what the storage address of that entry is.

Entry

This routine is initially entered at PRSERCH, and subsequently at location SERCH. The routine is entered from several routines within the loader.

Operation

1. This routine begins its operation by obtaining the number of entries currently in the reference table (this number is contained in area TBLCT), the size of a reference table entry (16 bytes), and the starting address of the reference table (always the highest address in storage, contained in area TBLREF).

2. The routine then checks the number of entries in the reference table. If the number is zero, the routine performs operation 5; otherwise, the routine performs operation 3.

3. The routine next determines the address of the first (or next) reference table entry to have its name checked, increments by one the count it is keeping of name comparisons, and compares the given name with the name contained in that entry. If the names are identical, PRSERCH branches to the location specified in the routine that linked to it. PRSERCH then returns the address of the REFTBL entry; else PRSERCH performs operation 4.

4. The routine then determines whether there is another reference table entry to be checked. If there is none, the routine performs operation 5; if there is another, the routine decrements by one the number of entries remaining and repeats its operation starting with operation 3.

5. If all the entries have been checked, and none contains the given name for which this routine is searching, the routine increments by one the count it is keeping of name comparisons, places that new value in area TBLCT, moves the given name to form a new reference table entry, and returns to the calling program.

Exits

This routine exits to either of two locations, both of which are specified by the routine that linked to this routine. The first location is that specified in the event that an entry for the given name is found; the second location is that specified in the event that such as entry is not found.

LOADER DATA BASES

ESD Card Codes (col. 25...)

Code	Meaning
00	SD (CSECT or START)
01	LD (ENTRY)
02	ER (EXTRN)
04	PC (Private code)
05	CM (COMMON)
06	XD (Pseudo-register)
0A	WX (WEAK EXTERN)

ESIDTB ENTRY

The ESD ID table (ESIDTB) is constructed separately for each text deck processed by the loader. The ESIDTB produces a correspondence between ESD ID numbers (used on RLD cards) and entries in the loader reference table (REFTBL) as specified by the ESD cards. Thus, the ESIDTB is constructed while processing the ESD cards. It is then used to process the TXT and RLD cards in the text deck.

The ESIDTB is treated as an array and is accessed by using the ID number as an index. Each ESIDTB entry is 16 bits long.

Bits	Meaning
0	If 1, this entry corresponds to a CSECT that has been previously defined. All TXT cards and RLD cards referring to this CSECT in this text deck should be ignored.
1	If 1, this entry corresponds to a CSECT definition (SD).
2	Waiting ESD items exist for this ESDID.
3	Unused.
4-15	REFTBL entry number (for example 1, 2, 3, etc.)

Bit 1 is very crucial because it is necessary to use the VALUE field of the REFTBL if the ID corresponds to an ER, CM, or PR; but, the INFO field of the REFTBL entry must be used in the ID corresponds to an SD.

REFTBL Entry

```
┌───────────────────────────────────────────────┐
|0(0)                                           |
|- - - - - - - - - -  NAME  - - - - - - - - - - |
|                                               |
|-----------------------------------------------|
|8(8)          |9(9)                            |
|    FLAG1     |              INFO              |
|--------------|--------------------------------|
|12(C)         |13(D)                           |
|    NOTE1     |              VALUE             |
|--------------|--------------------------------|
|16(10)        |17(11)                          |
|    FLAG2     |              ADDRESS           |
└───────────────────────────────────────────────┘
```

A REFTBL entry is 20 bytes. The fields have the following uses:

NAME Field: Contains the symbolic name from the ESD data item.

FLAG1 BYTE

Loader Code	ESD Code	Routine Label	Meaning
7C	00	XBYTE	PR - byte alignment
7D	01	XHALF	PR - halfword alignment
7E	03	XFULL	PR - fullword alignment
7F	07	XDBL	PR - doubleword alignment
80	05	XUNDEF	Undefined symbol
81	04	XCXD	Resolve CXD
82	02	XCOMSET	Define common area
83	05	WEAKEXT	Weak external reference
90	06	CTLLIB	TXTLIBs not to be used to resolve names

INFO Field: Depends upon the type of the ESD item.

ESD Item Type	INFO Field Meaning
SD (CSECT or START)	Relocation factor
LD (ENTRY)	Zero
CM (COMMON)	Maximum length
PR (Psuedo Register)	-

VALUE Field: depends upon the type of the ESD item, as does the INFO field.

ESD Item Type	VALUE Field Meaning
SD (CSECT or START)	Absolue address
LD (ENTRY)	Absolue address
CM (COMMON)	Absolue address
PR (Pseudo register)	Assigned value (starting from 0)

FLAG2 Byte

Bit	Meaning	Bit	Meaning
0	Unused	4	Unused
1	Unused	5	Name was located in a TXTLIB
2	Unused	6	Section definition entry
3	Unused	7	Name specifically loaded from command line.

ADDRESS Field: Unused

Entries may be created in the loader reference table prior to the actual defining of the symbol. For example, an entry is created for a symbol if it is referenced by means of an EXTRN (ER) even if the symbol has not yet been defined or its type known. Furthermore, common (CM) is not assigned absolute addresses until prior to the start of execution by the START command.

These circumstances are determined by the setting of the flag byte; if the symbol's value has not yet been defined, the value field specifies the address of a patch control block (PCB).

PATCH CONTROL BLOCK (PCB)

These are allocated from free storage and pointed at from REFTBL entries or other PCBs.

Byte	Meaning
0-3	Address of next PCB
5-7	Location of ADCON in storage
4	Flag byte

All address constant locations in loaded program for undefined symbols are placed on PCB chains.

LOADER INPUT RESTRICTIONS

All restrictions which apply to object files for the OS linkage editor apply to CMS loader input files.

Processing Commands that Manipulate the File System

Figure 9 lists the CMS modules that perform either general file system support functions or that perform data manipulation.

Managing the CMS File System

A description of the structure of the CMS file system and the flow of routines that access and update the file system follows.

How CMS Files Are Organized in Storage

CMS files are organized in storage by three types of data blocks: the file status table (FST), chain links, and file records. Figure 12 shows how these types of data blocks relate to each other; the following text and figures describe these relationships and the individual data blocks in more detail.

FILE STATUS TABLES

CMS files consist of 800-byte records whose attributes are described in the file status table (FST). The file status table is defined by DSECT FSTSECT. The FST consists of such information as the filename, filetype, and filemode of the file, the date on which the file was last written, and whether the file is in fixed-length or variable format. Also, the FST contains a pointer to the first chain link. The first chain link is a block that contains addresses of the data blocks that contain the actual data for the file.

The FSTs are grouped into 800-byte blocks called FST Blocks (these are sometimes referred to in listings as hyperblocks). Each FST block contains 20 FST entries, each describing the attributes of a separate file. Figure 13 shows the structure of an FST block and the fields defined in the FST.

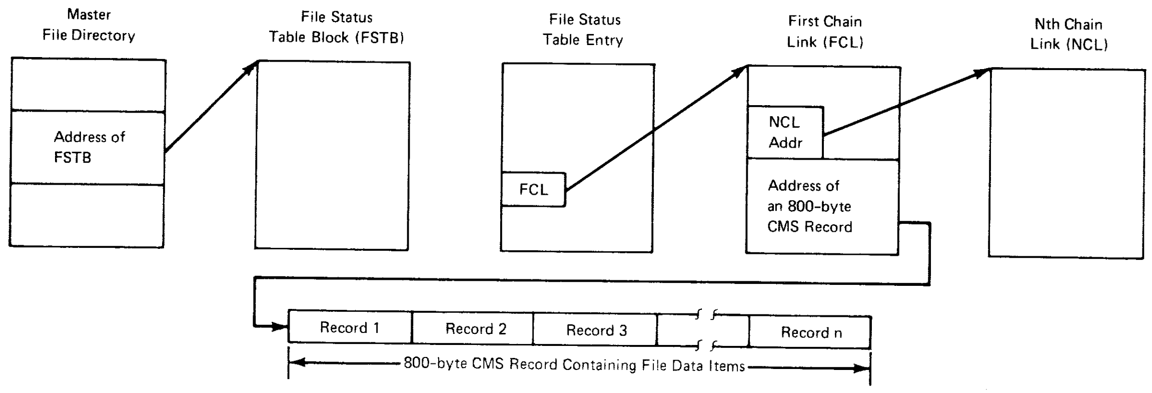


Figure 12. How CMS File Records Are Chained Together

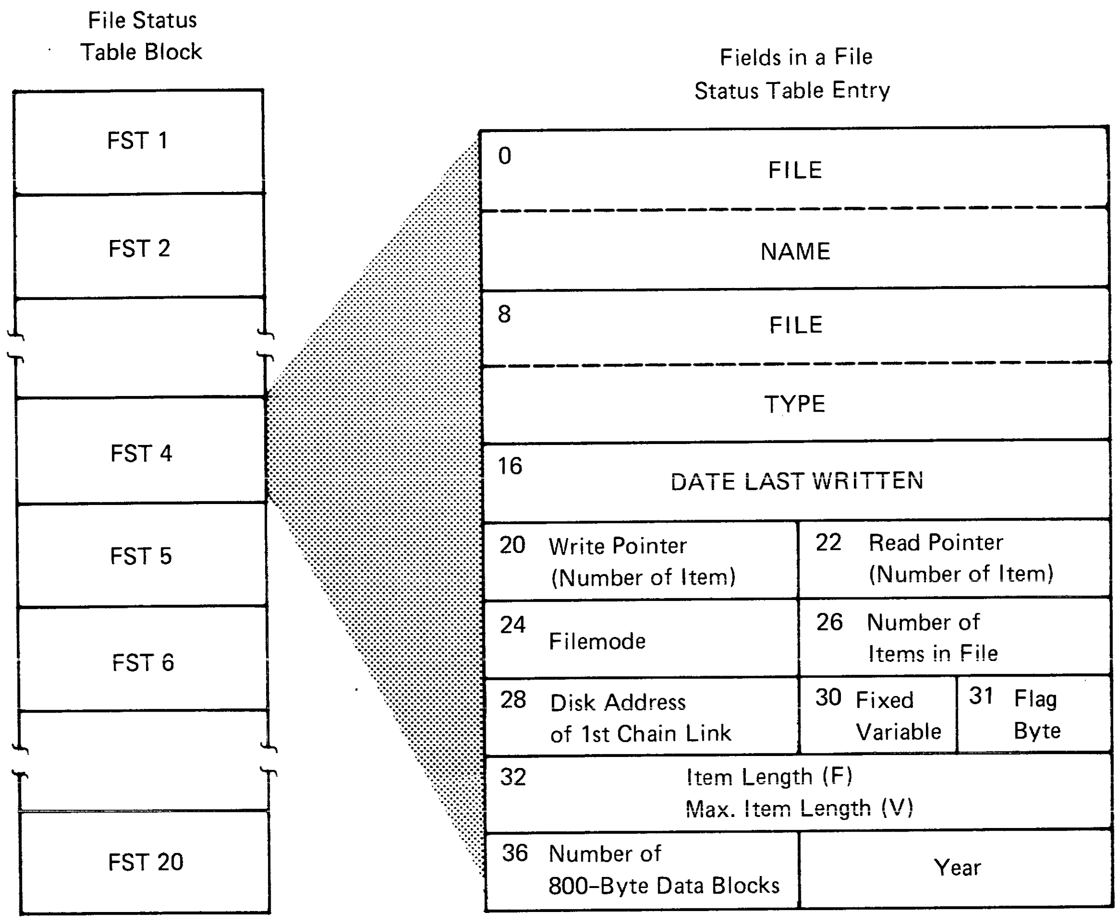

File Status Table Block
FST 1
FST 2
FST 4
FST 5
FST 6
FST 20
Fields in a File Status Table Entry
0 FILE
NAME
8 FILE
TYPE
16 DATE LAST WRITTEN
20 Write Pointer (Number of Item)
22 Read Pointer (Number of Item)
24 Filemode
26 Number of Items in File
28 Disk Address of 1st Chain Link
30 Fixed Variable
31 Flag Byte
32 Item Length (F) Max. Item Length (V)
36 Number of 800-Byte Data Blocks
Year

Figure 13. Format of a File Status Block; Format of a File Status Table

CHAIN LINKS

Chain links are 200- or 800-byte blocks of storage that chain the records of a file in storage. There are two types of chain links: first chain links and Nth chain links.

The first chain link points to two kinds of data. The first 80 bytes of the first chain link contain the halfword addresses of the remaining 40 chain links used to chain the records of the file. The next 120 bytes of the file are the halfword addresses of the first 60 records of the file.

The Nth chain links contain only halfword addresses of the records contained in the file.

Because there are 41 chain links (of which the first contains addresses for only 60 records), the maximum size for any CMS file is 16,060 800-byte records.

CMS RECORD FORMATS

CMS records are 800-byte blocks containing the data that comprises the file. For example, the CMS record may contain several card images or print images, each of which is referred to a record item. Figure 14 shows how chain links are chained together.

CMS records can be stored on disk in either fixed-length or variable-length format. However, the two formats may not be mixed in a single file.

Regardless of their format, the items of a file are stored by CMS in sequential order in as many 800-byte records as are required to accommodate them. Each record (except the last) is completely filled and items that begin in one record can end on the next record. Figure 15 shows the arrangement of records in files for files containing fixed-length records and files containing variable-length records.

The location of any item in a file containing fixed-length records is determined by the formula:

$$\text{locations} = \frac{(\text{Item Number} - 1) \times \text{Record Length}}{800}$$

where the quotient is the number of the item and the remainder is the displacement of the item into the file.

For variable-length records, each record is preceded by a 2-byte field specifying the length of the record.

Disk Organization

CMS virtual disks (also referred to as minidisks) are blocks of data designed to externally parallel the function of real disks. Several virtual disks may reside on one real disk.

A CMS virtual machine may have up to 10 virtual disks accessed during a terminal session, depending on user specifications. Some disks, such as the S-disk, are accessed during CMS initialization; however, most are accessed dynamically as they are needed during a terminal session.

PHYSICAL ORGANIZATION OF VIRTUAL DISKS

Virtual disks are physically organized in 800-byte records. Records 1 and 2 of each user disk are reserved for IPL. Record 3 contains the disk label. Record 4 contains the master file directory. The remaining records on the disk contain user file-related information such as the FSTs, chain links, and the individual file records discussed above.

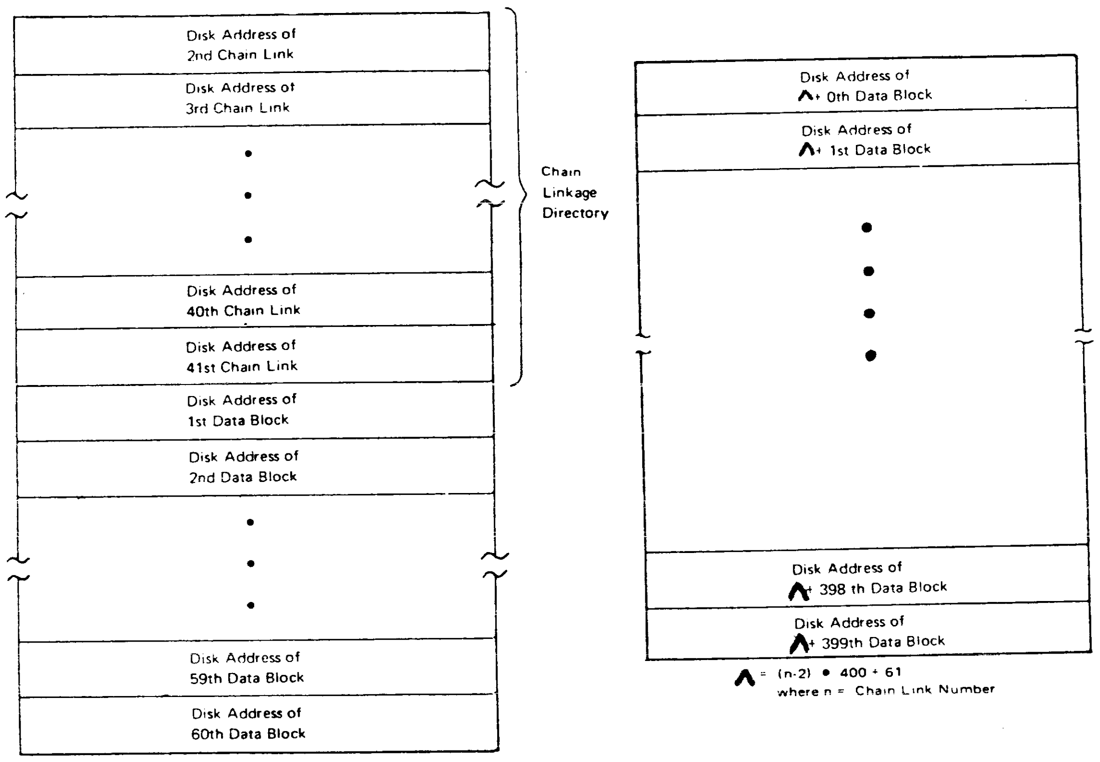

Disk Address of 2nd Chain Link
Disk Address of 3rd Chain Link
Disk Address of 40th Chain Link
Disk Address of 41st Chain Link
Chain Linkage Directory
Disk Address of 1st Data Block
Disk Address of 2nd Data Block
Disk Address of 59th Data Block
Disk Address of 60th Data Block
Disk Address of ʌ+ 0th Data Block
Disk Address of ʌ+ 1st Data Block
Disk Address of ʌ+ 398 th Data Block
Disk Address of ʌ+ 399th Data Block
ʌ = (n-2) • 400 + 61
where n = Chain Link Number

Figure 14. Format of the First Chain Link and Nth Chain Links

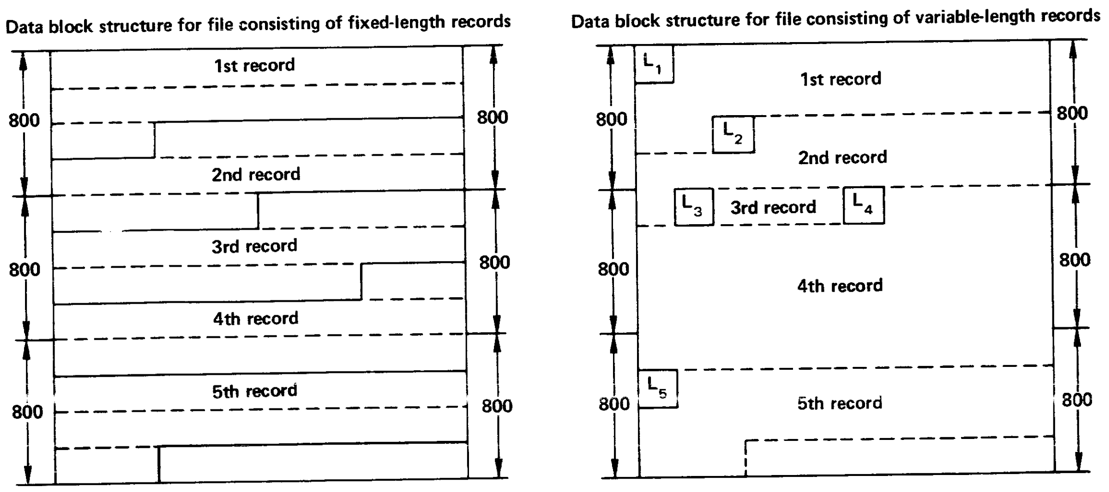


Figure 15. Arrangement of Fixed-Length Records and Variable-Length Records in Files

THE MASTER FILE DIRECTORY

The master file directory (MFD) is the major file management table for a virtual disk. As mentioned earlier, it resides on cylinder 0, track 0, record 4 of each virtual disk. Six types of information contained in the master file directory:

- The disk addresses of the FST entries describing user files on that disk.
- A 4-byte "sentinel," which can be either FFFD or FFFF. FFFD specifies that extensions of the QMSK (described below) follow. FFFF specifies that no QMSK extensions follow.
- Extensions to the QMSK, if any.
- General information describing the status of the disk:
  - ADTNUM -- The total number of 800-byte blocks on the user's disk.
  - ADTUSED -- The number of blocks currently in use on the disk.
  - ADTLEFT -- Number of blocks remaining for use (ADTNUM - ADTUSED).
  - ADTLAST -- Relative byte address of the last record in use on the disk.
  - ADTCYL -- Number of cylinders on the user's disk.
  - Unit Type -- A 1-byte field describing the type of the disk: 08 for a 2314, 09 for a 3330.
  - A bit mask called the QMSK, which keeps track of the status of the records on disk. The QMSK is described in more detail below.
  - Another bit map, called the QQMSK, which is used only for 2314 disks and performs a function similar to that of QMSK.

Figure 16 shows the structure of the master file directory. Figure 12 shows the relationship of the Master File Directory, which resides on disk, to data blocks brought into storage for file management purposes, for example, FSTs and chain links.

KEEPING TRACK OF READ/WRITE DISK STORAGE: QMSK AND QQMSK

Because large areas of disk space need not be contiguous in CMS, but are composed of 800-byte blocks chain-linked together, disk space management needs to determine only the availability of blocks, not extents. The status of the blocks on any read/write disk (which blocks are available and which are currently in use) is stored in a table called QMSK. The term QMSK is derived from the fact that a 2311 disk drive has four 800-byte blocks per track. One block is a "quarter-track", or QTRK, and a 200-byte area is a "quarter-quarter-track", or QQTRK. The bit mask for 2314, 2319, 3340, or 3330 records is called the QMSK, although each 800-byte block represents less than a quarter of a track on these devices.

On a 2314 or 2319 disk, the blocks are actually grouped fifteen 800-byte blocks per even/odd pair of tracks. An even/odd pair of tracks is called a track group. On a 3330 disk, the blocks are grouped fourteen 800-byte blocks per track. On a 3340 disk, the blocks are grouped into eight 800-byte blocks per track.

When the system is not in use, a user's QMSK resides on the Master File Directory; during a session it is maintained on disk, but also resides in real storage. QMSK is of variable length, depending on how many cylinders exist on the disk.

Each bit is associated with a particular block on the disk. The first bit in QMSK corresponds to the first block, the second bit to the second block, and so forth, as shown in Figure 17.

When a bit in QMSK is set to 1, it indicates that the corresponding block is in use and not available for allocation. A 0-bit indicates that the corresponding block is available. The data blocks are referred to by relative block numbers throughout disk space management, and the disk I/O routine, DMSDIO, finally converts this number to a CCHHR disk address.

A table called QQMSK indicates which 200 byte segments (QQTRK) are available for allocation and which are currently in use. QQMSK contains 100 entries, which are used to indicate the status of up to 100 QQTRK records. An entry in QQMSK contains either a disk address, pointing to a QQTRK record that is available for allocation, or zero. QQMSK is used only for 2314 files; for 3330, 3340, and 3350, the first chain link occupies the first 200-byte area of an 800-byte block.

The QMSK and QQMSK tables for read-only disks are not brought into storage, since no space allocation is done for a disk while it is read-only. They remain, as is, on the disk until the disk is accessed as a read/write disk.

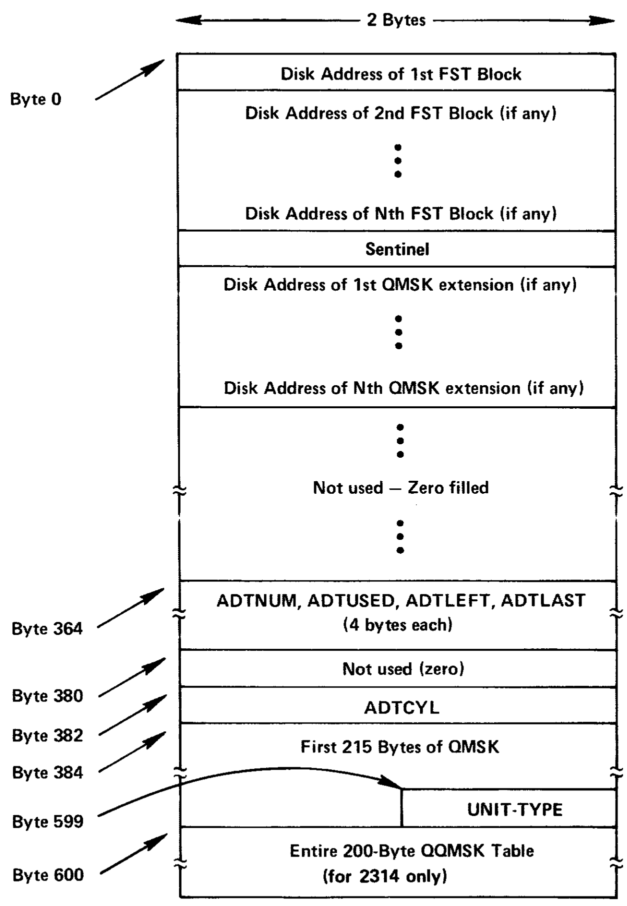
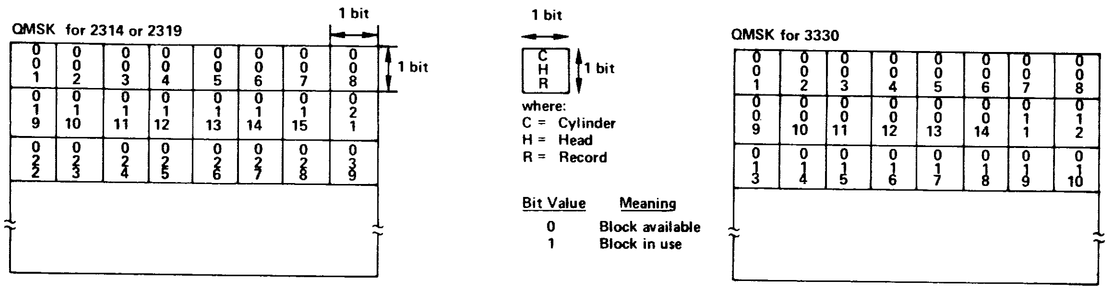

2 Bytes
Byte 0
Disk Address of 1st FST Block
Disk Address of 2nd FST Block (if any)
Disk Address of Nth FST Block (if any)
Sentinel
Disk Address of 1st QMSK extension (if any)
Disk Address of Nth QMSK extension (if any)
Not used — Zero filled
Byte 364
ADTNUM, ADTUSED, ADTLEFT, ADTLAST (4 bytes each)
Byte 380
Not used (zero)
Byte 382
ADTCYL
Byte 384
First 215 Bytes of QMSK
Byte 599
UNIT-TYPE
Byte 600
Entire 200-Byte QQMSK Table (for 2314 only)

Figure 16. Structure of the Master File Directory



Number of QMSK Extensions Required (if any)	Number of Cylinders on Disk			
	2314 or 2319	3330	3340	3350
0	1 - 11	1 - 6		
1	12 - 54	7 - 30		
2	55 - 96	31 - 54		
3	97 - 139	55 - 78		
4	140 - 182	79 - 102		
5	183 - 203	103 - 126		
6	-	127 - 150		
7	-	151 - 174		
8	-	175 - 198		
9	-	199 - 223		
10	-	224 - 246		

Figure 17. Disk Storage Allocation Using the QMSK Data Block

DYNAMIC STORAGE MANAGEMENT: ACTIVE DISKS AND FILES

CMS disks and files contained on disk are physically mapped using the data blocks described above: for disks, the QMSK, QQMSK, and the MFD; for files, the FST, chain links, and 800-byte file records. In storage, all of this data is accessed by means of two DSECTs whose addresses are defined in the DSECT NUCON, ADTSECT and AFTSECT.

Managing Active Disks: The Active Disk Table

The ADTSECT DSECT maps information in the active disk table (ADT). This information includes data contained in the MFD, FST blocks, the QMSK, and QQMSK. The DSECT comprises of ten "slots," each representing one CMS virtual disk. A slot contains significant information about the disk such as a pointer to the MFD for the disk, a pointer to the first FST block and pointers to the QMSK and QQMSK, if the disk is a R/W disk. Also contained in ADTSECT is information such as the number of cylinders on the disk, the number of records on the disk.

Managing Active Files: The Active File Table

Each open file is represented in storage by an active file table (AFT). The AFT (defined by the AFTSECT DSECT) contains data found on disk in FSTs, chain links, and data records. Also contained in the AFT is such information as the address of the first chain link for the file, the current chain link for the file, the address of the current data block, the fileid information for the file. Figure 1 shows the relationship between the AFT and other CMS data blocks.

CMS ROUTINES USED TO ACCESS THE FILE SYSTEM

DMSACC is the control routine used to access a virtual disk. In conjunction with DMSACM and DMSACF, DMSACC builds, in virtual storage, the tables CMS requires for processing files contained on the disk. The list below shows the logical flow of the main function of DMSACC.

ACCESS A VIRTUAL DISK: DMSACC

DMSACC: Scans the command line to determine which disk is specified.

DMSLAD: Looks up the address of the ADT for the disk specified on the command line.

DMSACC: Determines whether an extension to a disk has been specified on the command line and ensures that it is correctly specified.

DMSLAD: In the case where an extension has been specified, calls DMSLAD to ensure that the extension disk exists.

DMSLAD: Ensures that the specified disk is not already accessed as a R/W disk.

DMSFNS: In the case where the specified disk is replacing a currently accessed disk, closes any open files belonging to the duplicate disk.

DMSACC: Verifies the parameters remaining on the command line.

DMSALU: Releases any free storage belonging to the duplicate disk via a call to DMSFRE. Also, clears appropriate entries in the ADT for use by the new disk.

DMSACM: (Called as the first instruction by DMSACF) Reads, from the Master File Directory, QMSK, and the QQMSK for the specified disk; also, DMSACM updates the ADT for the specified disk using information from the MFD.

DMSACF: Reads into storage all the FST blocks associated with the specified disk.

DMSACC: Handles error processing or processing required to return control to DMSINT.

Handling I/O Operations

CMS input/output operations for disk, tape, and unit record devices are always synchronous. Disk and tape I/O is initiated via a privileged instruction, DIAGNOSE, whose function code requests CP to perform necessary error recovery. Control is not returned to CMS until the operation is complete, except for tape rewind or rewind and unload operations, which return control immediately after the operation is started. No interruption is ever received as the result of DIAGNOSE I/O. The CSW is stored only in the event of an error.

Input/output operations to a card reader, card punch, or printer are initiated via a normal START I/O instruction. After starting the operation, CMS enters the wait state until a device end interruption is received from the started device. Because the I/O is spooled by CP, CMS does not handle any exceptional conditions other than not ready, end-of-file, or forms overflow.

CMS input/output operations to the terminal may be either synchronous or asynchronous. Output to the terminal is always asynchronous, but a program may wait for all terminal input/output operations to complete by calling the console wait routine. Input from the terminal is usually synchronous but a user may cause CMS to issue a read by pressing the attention key. A program may also asynchronously stack data to be read by calling the console attention routine.

UNIT RECORD I/O PROCESSING

Seven routines handle I/O processing for CMS: DMSRDC, DMSPUN, and DMSPRT handle the READCARD, PUNCH, and PRINT commands and pass control to te actual I/O processors, DMSCIO (for READCARD and PUNCH) or DMSPIO (for PRINT). DMSCIO and DMSPIO issue the SIO instructions that cause I/O to take place. Two other routines, DMSIOW and DMSITI, handle synchronization processing for I/O operations. Figure 18 shows the overall flow of control for I/O operations.

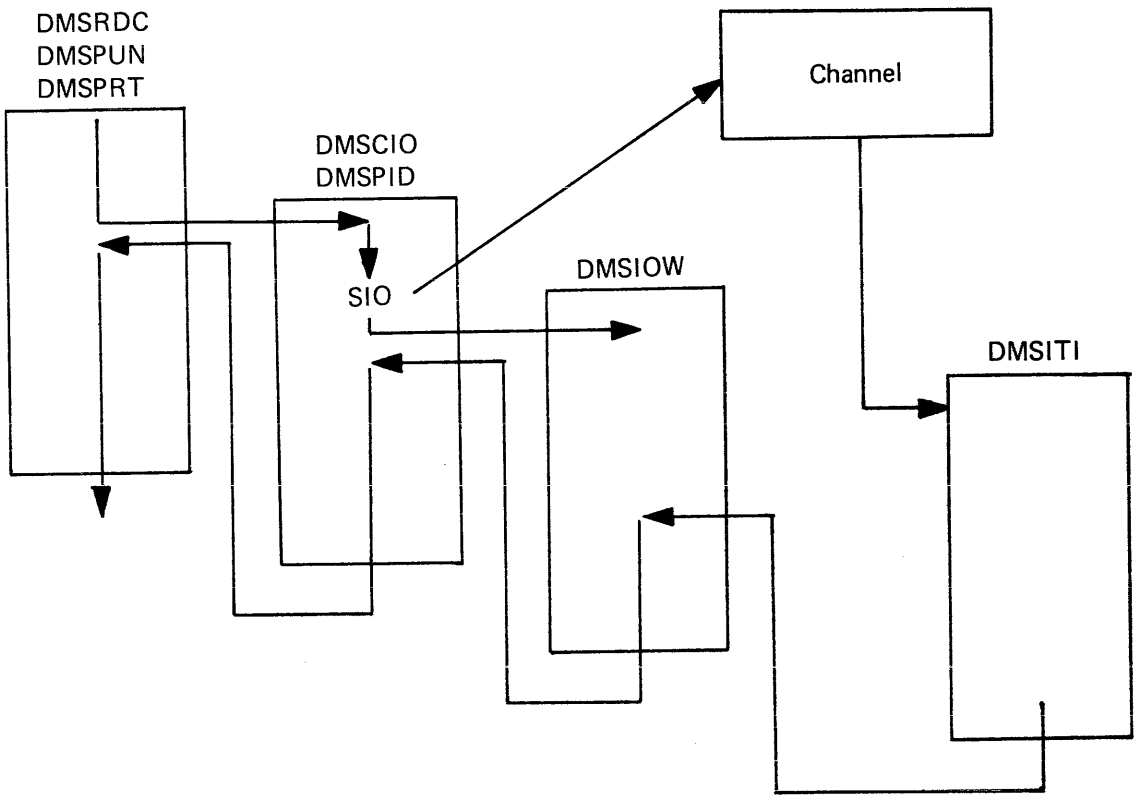


Figure 18. Flow of Control for Unit Record I/O Processing

The following are more detailed descriptions of the flow of control for the read, punch, and print unit record control functions.

Read a Card

DMSRDC: Initializes block length and unit record size.

DMSCIO: Initializes areas to read records.

DMSCIO: Issues an SIO command to read a record.

DMSIOW: Sets the wait bit for the virtual card reader and load the I/O old PSW from NUCON. This causes CMS to enter a wait state until the read I/O is complete.

DMSITI: Ensures that this interrupt is for the virtual reader. If not, the I/O old PSW is loaded, returning CMS to a wait state. If the interrupt is for the reader, DMSITI resets the wait bit in the I/O old PSW and loads it, causing control to return to DMSIOW.

DMSIOW: Places the symbolic name of the interrupting device in the PLIST and passes control to the calling routine.

DMSCIO: Checks for SENSE information and handle I/O errors, if necessary.

DMSCWR: Displays a control record at the console.

DMSSCN: If another control record is encountered, formats it via DMSSCN.

DMSCWR: Displays the new control record at the console.

DMSFNS: Closes the file when end-of-file occurs.

DMSRDR: Issues a CP CLOSE command to close the card reader.

Punch a Card

DMSPUN: Ensures that a virtual punch is available; processes PUNCH command options.

DMSSTT: Verifies the existence of the file and returns its starting address.

DMSPUN: If requested, sets up a header record and calls DMSCWR to write it to the console.

DMSBRD: Reads a block of data into the read buffer; continues reading until the buffer is filled.

DMSCIO: Initializes areas to punch records.

DMSCIO: Issues the SIO instruction to punch the contents of the buffer.

DMSCIO: Issues a call to DMSIOW to wait for completion of the punch I/O operation.

DMSIOW: Sets the wait bit on for the virtual punch device and loads the I/O old PSW from NUCON. This causes CMS to enter a wait state until the punch operation completes.

DMSITI: Ensures that this interrupt is for the punch. If not, the I/O old PSW is loaded returning CMS to a wait state. If the interrupt is for the punch, DMSITI resets the wait bit in the I/O old PSW and then loads the PSW, returning control to DMSIOW.

DMSIOW: Places the symbolic name of the interrupting device in the PLIST and passes control to DMSCIO.

DMSCIO: Checks for SENSE information and handles I/O errors, if any.

DMSPUN: Handles error returns and resets constants for the next punch operation.

DMSFNS: Closes the file and returns control to the command handler, DMSINT.

Print a File

DMSPRT: Determines the device type of the printer. Checks out the specified fileid. Checks out the options specified on the PRINT command line.

DMSSCN: Verifies the existence of the file and returns its starting address.

DMSPRT: Determines the record size to be printed and sets up an appropriate buffer area via a call to DMSFRE.

DMSFRE: Obtains storage space to be used as a buffer.

DMSPRT: Determines whether the file to be printed is a library member or an input file.

DMSBRD: Reads a record; continues reading until the buffer is filled. When the buffer is filled, calls DMSPIO to issue the SIO instruction to begin the print operation.

DMSPIO: Issues the print SIO instruction and then calls DMSIOW to wait until the the I/O operation completes.

DMSIOW: Sets the wait bit for the virtual printer device and load the I/O old PSW from NUCON. This causes CMS to enter a wait state until the print operation completes.

DMSITI: Ensures that the interrupt is for the printer. If not, the I/O old PSW is reloaded, returning CMS to a wait state. If the interrupt is for the printer, DMSITI resets the WAIT bit in the I/O old PSW and loads that PSW, returning control to DMSIOW.

DMSIOW: Places the symbolic name of the device in the last word of the PLIST and passes control to DMSPIO.

DMSPIO: Performs channel testing and handles errors. TIO instructions and sense SIO instructions are issued during the test processing. These operations are synchronized using DMSIOW and DMSITI in the manner described above. When the I/O completes successfully, control returns to DMSPRT.

DMSPRT: Determines whether all file records have been printed. If so, control returns to the caller. Otherwise, the address of the buffer is updated and more print operations are performed.

Printer Carriage Control Characters Used by DMSPIO

CMS supports the use of ASCII control characters and machine carriage control characters for the printed output. Part of the CMS implementation depends upon the fact that the set of ASCII control characters has almost nothing in common with the set of machine control characters. There are two exceptions to this, the characters X'C1' and X'C3'. These two characters, when interpreted as ASCII control characters, have the following meanings:

C1 = Skip to channel 10 before print.

C3 = Skip to channel 12 before print.

The same characters, when interpreted as machine control characters, have the following meanings:

C1 = Write, then skip to channel 8 after print.

C3 = Do not write, but skip to channel 8 immediately.

In printing lines containing carriage control characters, CMS has the capability of operating in two modes. In the first mode, which may be called ASCII control characters or machine control characters of either type are recognized and properly interpreted, except that the two conflicting characters are always interpreted as ASCII control characters. In the second mode, which may be called machine-only, only machine control characters are recognized, and the two conflicting characters are treated as machine.

The DMSPIO function uses a bit in the plist to indicate which of the two modes is in effect for printing.

The PRINTL macro always uses ASA control character or machine control character mode.

The PRINT command with the CC option always runs in ASCII control character or machine control character mode.

OS simulation output, which is used, for example, by the MOVEFILE command, uses the RECFM field in the DCB or in the FILEDEF command to determine which mode is to be used. If FA, VA, or UA is specified, then ASCII control character or machine control character mode is used. If FM, VM, or UM is specified, then machine-only mode is used. If no control character specification is included with the RECFM, then it is assumed that the output line begins with a valid data character, rather than with a control character, and single spacing is always used.

Handling Interruptions

Figure 9 lists the CMS modules that process interruptions for CMS. CMS modules are described briefly in "CMS Module Description." SVC 9 interruption processing is described in "Maintaining an Interactive Console Environment."

Disk I/O in CMS

Files residing on disk are read and written using DMSDIO. DMSDIO has two entry points: DMSDIOR, which is entered for a read I/O operation, and DMSDIOW, which is entered for a write operation.

The actual disk I/O operation is performed using the DIAGNOSE code 18 instruction. A return code of 0 from CP indicates a successful completion of the I/O operation. If the I/O is not successful, CP performs error recording, retry, recovery, or ABEND procedures for the virtual machine.

READ OR WRITE DISK I/O

<u>DMSDIO</u>: Initializes the CCW to perform read operations.

<u>DMSLAD</u>: Obtains the address of the disk from which to read or write.

<u>DMSDIO</u>: Determines the size of the record to be read or written.

<u>DMSFRE</u>: Gets enough storage to contain the record if the request is for a record longer than 800 bytes.

<u>DMSDIO</u>: Reads records continually until all records for the file have been read.

DMSFRE: Returns the buffer to free storage if the record was longer than 800 bytes.

DMSDIO: Returns to the caller.

Managing CMS Storage

DMSFRE handles requests for CMS free storage. The sections of CMS storage have the following uses:

- DMSNUC (X'00000' to approximately X'03000') - This is the nucleus constant area. It contains pointers, flags, and other data maintained by the various system routines.

- Low-core DMSFREE free storage area (approximately X'03000' to X'0E000') - This area is a free storage area, from which requests from DMSFREE are allocated. The top part of this area contains the file directory for the system disk (SSTAT). If there is enough room (as there will be in most cases), the FREETAB table also occupies this area, just below the SSTAT.

- Transient program area (X'0E000' to X'10000') - Because it is not essential to keep all nucleus functions resident in storage all the time, some of them are made "transient." This means that when they are needed, they are loaded from the disk into the transient program area. Such programs may not be longer than two pages, because that is the size of the transient area. (A page is 4096 bytes of virtual storage.)

- CMS nucleus (X'10000' to X'20000') - Segment 1 of storage contains the reentrant code for the CMS nucleus routines. In shared CMS systems, this is the protected segment. That is, this segment must consist only of reentrant code, and may not be modified under any circumstances. This fact implies certain system restrictions for functions which require that storage be modified, such as the fact that DEBUG breakpoints or CP ADSTOP commands cannot be placed in this segment, in a saved system.

- User program area (X'20000' to loader tables) - User programs are loaded into this area by the LOAD command. Storage allocated by means of the GETMAIN macro instruction is taken from this area, starting from the high address of the user program. In addition, this storage area can be allocated from the top down by DMSFREE, if not enough storage is available in the low-core DMSFREE storage area. Thus, the effective size of the user program area is reduced by the amount of free storage which has been allocated from it by DMSFREE.

- Loader tables (top pages of storage) - The top of storage is occupied by the loader tables, which are required by the CMS loader. These tables indicate which modules are currently loaded in the user program area (and the transient program area after a LOAD command). The size of the loader tables can be varied by the SET LDRTBLS command.

TYPES OF ALLOCATED FREE STORAGE

Free storage can be allocated by means of the GETMAIN or DMSFREE macros.

Storage allocated by means of the GETMAIN macro is taken from the user program area, beginning with the high address of the user program.

Storage allocated by means of the DMSFREE macro can be taken from several areas.

First, DMSFREE requests are allocated from the low-address free storage area. If requests cannot be satisfied from there, they will be satisfied from the user program area.

In addition, requests are further broken down between requests for user storage and nucleus storage, as specified in the TYPE parameter of the DMSFREE macro. These two types of storage are kept in separate 4K pages. It is possible, if there are no 4K pages completely free in low storage, for no storage of one type to be available in low storage, while there is storage of the other type available there.

GETMAIN FREE STORAGE MANAGEMENT POINTERS

All GETMAIN storage is allocated in the user program area, starting from the end of the user's actual program. Allocation begins at the location pointed to by NUCON pointer MAINSTRT. The location MAINHIGH in NUCON is the pointer to the highest address of GETMAIN storage.

When the STRINIT macro is executed, both MAINSTRT and MAINHIGH are initialized to the end of the user's program, in the user program area. As storage is allocated from the user program area to satisfy GETMAIN requests, the MAINHIGH pointer is adjusted upward. Such adjustments are always in multiples of doublewords, so that this pointer is always on a doubleword boundary. As the allocated storage is released, this pointer is adjusted downward.

The pointer MAINHIGH can never be higher than FREELOWE, the pointer to the lowest address of DMSFREE storage allocated in the user program area. If a GETMAIN request cannot be satisfied without extending MAINHIGH above FREELOWE, GETMAIN takes an error exit, indicating that insufficient storage is available to satisfy the request.

The area between MAINSTRT and MAINHIGH may contain blocks of storage that are not allocated, and that are therefore available for allocation by a GETMAIN instruction. These blocks are chained together, with the first one pointed to by the NUCON location MAINLIST.

The format of an element on the GETMAIN free element chain is as follows:

```
            <------------ 4 bytes ------------>
            ┌─────────────────────────────────┐
            | FREPTR -- pointer to next free  |
    0(0)    |      element in the chain, or 0 |
            |      if there is no next element|
            |─────────────────────────────────|
            | FRELEN -- length, in bytes, of  |
    4(4)    |    this element                 |
            |                                 |
            |─────────────────────────────────|
            |  Remainder of this free element |
            .                                 .
            .                                 .
            .                                 .
```

DMSFREE FREE STORAGE POINTERS

The pointers FREEUPPR and FREELOWE in NUCON indicate the amount of storage which DMSFREE has allocated from the high portion of the user program area. These pointers are initialized to the beginning of the system loader tables.

The pointer FREELOWE is the pointer to the lowest address of DMSFREE storage in the user program area. As storage is allocated from the user program area to satisfy DMSFREE requests, this pointer is adjusted downward. Such adjustments are always in multiples of 4K, so that this pointer is always on a 4K boundary. As the allocated storage is released, this pointer is adjusted upward when whole 4K pages are completely free.

The pointer FREELOWE can never be lower than MAINHIGH, the pointer to the highest address of GETMAIN storage. If a DMSFREE request cannot be satisfied without extending FREELOWE below MAINHIGH, then DMSFREE takes an error exit, indicating that insufficient storage is available to satisfy the request.

The FREETAB free storage table is kept in free storage, usually just below the master file directory for the system disk. If there was no space available there, then FREETAB was allocated from the top of the user program area. This table contains one byte for each page of virtual storage. Each such byte contains a code indicating the use of that page of virtual storage. The codes in this table are as follows:

USERCODE (1): If the page is assigned to user storage.

NUCCODE (2): If the page is assigned to nucleus storage.

TRNCODE (3): If the page is part of the transient program area.

USARCODE (4): If the page is part of the user program area.

SYSCODE (5): If the page is none of the above.

In these cases, the page is assigned to system storage, system code, or the loader tables.

Other DMSFREE storage pointers are maintained in the DMSFRT control section, in NUCON. The most important fields there are the four chain header blocks.

Four chains of elements are not allocated to be associated with DMSFREE storage: The low-storage nucleus chain, the low-storage user chain, the high-storage nucleus chain, and the high-storage user chain. For each of these chains, exists a control block consisting of four words, with the following format:

```
        <------------ 4 bytes ------------>
        ┌─────────────────────────────────┐
        |POINTER -- pointer to the first  |
  0 (0) |   free element on the chain, or |
        |   zero, if the chain is empty.  |
        |─────────────────────────────────|
        | NUM -- the number of elements on|
  4 (4) |        the chain.               |
        |                                 |
        |─────────────────────────────────|
        | MAX -- the value in this word is|
  8 (8) |    the size of the largest free |
        |    element on the chain.        |
        |─────────────────────────────────|
        | FLAGS- | SKEY - | TCODE -| Unused|
 12 (C) | Flag   |Storage |FREETAB |       |
        | byte   | key    |  code  |       |
        └─────────────────────────────────┘
```

These fields have the following meanings and uses:

POINTER	This field points to the first element on this chain of free elements. If there are no elements on this free chain, then the POINTER field contains a zero.
NUM	This field contains the number of elements on this chain of free elements. If there are no elements on this free chain, then this field contains a zero.
MAX	This field is used for the purpose of avoiding searches which will fail. It contains the size, in bytes, of the largest element on the free chain. Thus, a search for an element of a given size will not be made if that size exceeds the MAX field.
FLAGS	The following flags are used:

FLCLN (X'80')
Clean-up flag - This flag is set if the chain must be cleaned up. This is necessary in the following circumstances:

- If one of the two high-core chains contains a 4K page that is pointed to by FREELOWE, then that page can be removed from the chain, and FREELOWE can be increased.

- All completely non-allocated 4K pages are kept on the user chain, by convention. Thus, if one of the nucleus chains (low-core or high-core) contains a full page, then this page must be transferred to the corresponding user chain.

FLCLB (X'40')
Clobbered flag - Set if the chain has been destroyed.

FLHC (X'20')
High-core chain - Set for both the nucleus and user high-core chains.

FLNU (X'10')
Nucleus chain - Set for both the low-core and high-core nucleus chains.

FLPA (X'08')
Page available - This flag is set if there is a full 4K page available on the chain. Note that this flag may be set even if there is no such page available.

SKEY This one-byte field contains the storage key assigned to storage on this chain.

TCODE This one-byte field contains the FREETAB table code for storage on this chain.

Each element on the free chain has the following format:

```
           <------------- 4 bytes ------------->
           ┌───────────────────────────────────┐
           |  POINTER -- pointer to the next   |
     0(0)  |      element in the free chain    |
           |                                   |
           |───────────────────────────────────|
           |  SIZE -- size of this free        |
     4(4)  |     element, in bytes             |
           |                                   |
           |───────────────────────────────────|
           |  Remainder of this free element   |
           .                                   .
           .                                   .
           .                                   .
```

When the user issues a variable length GETMAIN, the control program reserves 6 1/2 pages for CMS usage; this is a designed and set value. If the user wants more space, for example, for more directories, he should free (from the high end of storage) some of the variable GETMAIN area.

As indicated in the illustration above, the POINTER field points to the next element in the chain, or contains the value zero if there is no next element. The SIZE field contains the size of this element, in bytes.

All elements within a given chain are chained together in order of descending storage address. This is done for two reasons:

1. Because the allocation search is satisfied by the first free element that is large enough, the allocated elements are grouped together at the top of the storage area, and prevent storage fragmentation. This is particularly important for high-storage free storage allocations, because it is desirable to keep FREELOWE as high as possible.

2. If free storage does become somewhat fragmented, the search causes as few page faults as possible.

As a matter of convention, completely nonallocated 4K pages are kept on the user chain rather than the nucleus chain. This is because requests for large blocks of storage are made, most of the time, from user storage rather than from nucleus storage. Nucleus requests need to break up a full page less frequently than user requests.

DMSFRE METHOD OF OPERATION

A description of the algorithms which allocate and release blocks follows. The descriptions are based on the assumption that neither AREA=LOW nor AREA=HIGH was specified in the DMSFREE macro call. If either was specified, then the algorithm must be appropriately modified.

ALLOCATING USER FREE STORAGE: When DMSFREE with TYPE=USER (the default) is called, the following steps are taken to satisfy the request. As soon as one of the steps succeeds, then processing can terminate. DMSFRE:

1. Searches low-storage user chain for a block of the required size.
2. Searches the high-storage user chain for a block of the required size.
3. Extends high-storage user storage downward into the user program area, modifying FREELOWE in the process.
4. For fixed requests, there is nothing more to try. For variable requests, DMSFRE puts all available storage in the user program area onto the high-storage user chain, and then allocates the largest block available on either the high-storage user chain or the low-storage user chain. The allocated block is not satisfactory, if it is not larger then the minimum requested size.

ALLOCATING NUCLEUS FREE STORAGE: When DMSFREE with TYPE=NUCLEUS is called, the following steps are taken in an attempt to satisfy the request, until one succeeds. DMSFREE:

1. Searches the low-storage nucleus chain for a block of the required size.
2. Gets free pages from low-storage user chain, if any are available, and removes them to the low-storage nucleus chain.
3. Searches the high-storage nucleus chain for a block of the required size.
4. Gets free pages from the high-storage user chain, if they are available, and removes them to the highstorage nucleus chain.
5. Extends high-storage nucleus storage downward into the user program area, modifying FREELOWE in the process.
6. For fixed requests, there is nothing more to try. For variable requests, DMSFRE puts all available pages from the user chains and the user program area onto the nucleus chains, and allocates the largest block available on either the low-storage nucleus chains or the high-storage nucleus chains.

RELEASING STORAGE: When DMSFRET is called, the block being released is placed on the appropriate chain. At that point, the cleanup operation is performed, if necessary, to advance FREELOWE, or to move pages from the nucleus chain to the corresponding user chain.

Similar cleanup operations are performed, when necessary, after calls to DMSFREE, as well.

RELATIVE EFFICIENCY OF DMSFREE REQUESTS

The types of DMSFREE request in decreasing order of efficiency, are as follows:

1. User fixed storage requests, any size.

2. Nucleus fixed storage requests, for small blocks (less than one page in size).

3. Nucleus fixed storage request, for large blocks.

4. User variable storage requests. (Variable requests are no less efficient than fixed requests, if the maximum block size requested can be allocated.)

5. Fixed variable storage requests, if the maximum block size requested cannot be allocated.

RELEASING ALLOCATED STORAGE

STORAGE ALLOCATED BY GETMAIN: Storage allocated by the GETMAIN macro instruction may be released in any of the following ways:

- A specific block of such storage may be released by means of the FREEMAIN macro instruction.

- The STRINIT macro instruction releases all storage allocated by any previous GETMAIN requests.

- Almost all CMS commands call the STRINIT routine. Thus, executing almost any CMS command causes all GETMAIN storage to be released.

STORAGE ALLOCATED BY DMSFREE: Storage allocated by the DMSFREE macro instruction may be released in either of the following ways:

- A specific block of such storage may be released by means of the DMSFRET macro instruction.

- Whenever any user routine or CMS command abends (so that the routine DMSABN is entered), and the ABEND recovery facility of the system is invoked, all DMSFREE storage with TYPE=USER is released automatically.

Except in the case of ABEND recovery, storage allocated by the DMSFREE macro is never released automatically by the system. Thus, storage allocated by means of this macro instruction should always be released explicitly by means of the DMSFRET macro instruction.

DMSFRE SERVICE ROUTINES

The system uses the DMSFRES macro instruction to request certain free storage management services. The options and their meanings are as follows:

- INIT1--DMSINS calls this option to invoke the first free storage initialization routine, to allow free storage requests to access the system disk. Before this routine is invoked, no free storage requests may be made. After this routine has been invoked, free storage requests may be made, but these are subject to the following restraints until the second free storage management initialization routine has been invoked:

-- All requests for user storage are changed to requests for nucleus storage.

-- Only partial error checking is performed by the DMSFRET routine. In particular, it is possible to release a block that was never allocated.

-- All requests that are satisfied in high storage must be temporary, because all high storage allocated is released when the second free storage initialization routine is invoked.

When CP's saved system facility is used, the CMS system is saved at the point just after the system disk has been accessed. This means that it is necessary for DMSFRE to be used before the size of virtual storage is known, because the saved system can be used on any size virtual machine. Thus, the first initialization routine initializes DMSFRE so that limited functions can be requested, while the second initialization routine performs the initialization necessary to allow the full functions of DMSFRE to be requested.

- INIT2--This option is called by DMSINS to invoke the second initialization routine. This routine is invoked after the size of virtual storage is known, and it performs the initialization necessary to allow all the functions of DMSFRE to be used. The second initialization routine performs the following steps:

  -- Releases all storage that has been allocated in the highstorage area.

  -- Allocates the FREETAB free storage table. This table contains one byte for each 4096-byte page of virtual storage, and so cannot be allocated until the size of virtual storage is known. It is allocated in the low-address free storage area, if there is enough room available. If not, then it is allocated in the higher free storage area. For a 256K virtual machine, FREETAB contains 64 bytes; for a 16 million byte machine, it contains 4096 bytes.

  -- The FREETAB table is initialized, and all storage protection keys are initialized.

  -- All completely non-allocated 4K pages on the nucleus free storage chain are removed to the user chain. Any other necessary cleaning up operations are performed.

- CHECK--This option can be called at any time for system debugging purposes. It invokes a routine that performs a thorough check of all free storage chains for consistency and correctness. Thus, it checks to see whether any free storage pointers have been destroyed.

- CKON--This option turns on a flag which causes the CHECK routine described in the preceding paragraph to be invoked each time any call is made to DMSFREE or DMSFRET. This can be useful to pinpoint a problem that is, for example, destroying free storage management pointers. Care should be taken when using this option, because the CHECK routine is coded to be thorough rather than efficient.

Thus, after the CKON option has been invoked, each call to DMSFREE or DMSFRET takes many times as long to be completed as before. This can impact the efficiency of system functions.

- CKOFF--Use of this option turns off the flag that was turned by the CKON option, described in the preceding paragraph.
- UREC--This option is called by DMSABN during the ABEND recovery process to release all USER storage.
- CALOC--This option is called by DMSABN after the ABEND recovery process has been completed. It invokes a routine that returns, in register 0, the number of doublewords of free storage that have been allocated. This figure is used by DMSABN to determine whether ABEND recovery has been successful.

STORAGE PROTECTION KEYS

In general, the following rule applies: system storage is assigned the storage key of X'F', while user storage is assigned the key of X'E'. This is the storage key associated with the protected areas of storage, not to be confused with the PSW or CAW key used to access that storage.

The specific key assignments are as follows:

- The NUCON area is assigned the key of X'F', with the exception of a half-page containing the OPSECT and TSOBLOKS areas, which has a key of X'E'.
- Free storage allocated by DMSFREE is broken up into user storage and nucleus storage. The user storage has a protection key of X'E', while the nucleus storage has a key of X'F'.
- The transient program area has a key of X'E'.
- The CMS nucleus code has a storage key of X'F'. In saved systems, this entire segment is protected by CP from modification even by the CMS system, and so must be entirely reentrant.
- The user program area is assigned the storage key of X'E', except for those pages which contain Nucleus DMSFREE storage. These latter pages are assigned the key of X'F'.
- The loader tables are assigned the key of X'F'.

CMS SYSTEM HANDLING OF PSW KEYS

The CMS nucleus protection scheme protects the CMS nucleus from inadvertent destruction by a user program. This mechanism, however, does not prevent a user from writing in system storage intentionally. Because a CMS user can execute privileged instructions, he can issue a LOAD PSW (LPSW) instruction and load any PSW key he wishes. If a user defeats nucleus protection in this way there is nothing to prevent his program from:

- Modifying nucleus code

- Modifying a table or constant area
- Losing files by modifying a CMS file directory

In general, user programs and disk-resident CMS commands run with a PSW key of X'E', while nucleus code runs with PSW key of X'0'.

There are, however, some exceptions to this rule. Certain disk-resident CMS commands run with a PSW key of X'0', because they need to modify nucleus pointers and storage. On the other hand, the nucleus routines called by the GET, PUT, READ and WRITE macros run with a user PSW key of X'E', to increase efficiency.

Two macros, DMSKEY and DMSEXS, are available for changing the PSW key. The DMSKEY macro changes the PSW key to the user value or the nucleus value. DMSKEY NUCLEUS causes the current PSW key to be placed in a stack, and a value of 0 to be placed in the PSW key. DMSKEY USER causes the current PSW key to be placed in a stack, and a value of X'E' to be placed in the PSW key. DMSKEY RESET causes the top value in the DMSKEY stack to be removed and re-inserted into the PSW.

It is a CMS requirement when a routine terminates, that the DMSKEY stack must be empty. This means that a routine should execute a DMSKEY RESET macro instruction for each DMSKEY NUCLEUS macro instruction and each DMSKEY USER macro instruction executed by the routine.

The DMSKEY key stack has a maximum depth of seven for each routine. In this context, a "routine" is anything invoked by an SVC call. The DMSEXS ("execute in system mode") macro instruction is useful in situations where a routine is running with a user PSW key, but wishes to execute a single instruction with the nucleus PSW key. The single instruction may be specified as the argument to the DMSEXS macro, and that instruction is executed with a system PSW key.

CP HANDLING FOR SAVED SYSTEMS

The explanation of saved system nucleus protection depends on the VSK, RSK, VPK and RPK:

1. Virtual Storage Key (VSK) - This is the storage key assigned by the virtual machine using the virtual SSK instruction.
2. Real Storage Key (RSK) - This is the actual storage key assigned by CP to the 2K page.
3. Virtual PSW Key (VPK) - This is the PSW storage key assigned by the virtual machine, by means of an instruction such as LPSW (Load PSW).
4. Real PSW Key (RPK) - This is the PSW storage key assigned by CP, which is in the real hardware PSW when the virtual machine is running.

When there are no shared segments in the virtual machine, then storage protection works as it does on a real machine. RSK=VSK for all pages, and RPK=VPK for the PSW.

However, when there is a shared segment (as in the case of segment 1 of CMS in the saved system), it is necessary for CP to protect the shared segment. For non-CMS shared systems, it does this by, essentially, ignoring the values of the VSKs and VPK, and assigning the real values as follows: RSK=0 for each page of the shared segment, RSK=F for all other pages, and RPK=F, always, for the real PSW. The SSK instruction is ignored, except to save the key value in a table in case the virtual machine later does an ISK to get it back.

For the CMS saved system, the RSKs and RPK are initialized as before, but resetting the virtual keys has the following effects:

- If the virtual machine uses an SSK instruction to reset a VSK, CP does the following: If the new VSK is nonzero, CP resets the RSK to the value of the VSK; if the new VSK is zero, CP resets RSK to F.

- If the virtual machine uses a LPSW (or other) instruction to reset the VPK, CP does the following: If the new VPK is zero, CP resets the RPK to the value of the VPK; if the new VPK is zero, CP resets RPK to F.

- If the VPK=0 and the RPK=F, storage protection may be handled differently. In a real machine, a PSW key of 0 would allow the program to store into any storage location, no matter what the storage key. But under CP, the program gets a protection violation, unless the RPK of the page happens to be F.

  Because of this, there is extra code in the CP program check handling routine. Whenever a protection violation occurs, CP checks to see if the following conditions hold:

  -- The virtual machine running is the saved CMS system, running with a shared segment.

  -- The VPK = 0. The virtual machine is operating as though its PSW key is 0.

  -- The RSK of the page into which the store was attempted is nonzero, and different from the RPK.

If any one of these three conditions fails to hold, then the protection violation is reflected back to the virtual machine.

If all three of these conditions hold, then the RPK (the real protection key in the real PSW) is reset to the RSK of the page into which the store was attempted.

<u>EFFECT ON CMS</u>: In CMS, this works as follows: CMS keeps its system storage in protect key F (RSK = VSK = F), and user storage in protect key E (RSK = VSK = E).

When the CMS supervisor is running, it runs in PSW key 0 (VPK = 0, RPK = F), so that CMS gets a protection violation the first time it tries to store into user storage (VSK = RSK = E). At that point, CP changes the RPK to E, and lets the virtual machine re-execute the instruction which caused the protection violation. There is not another protection violation until the supervisor goes back to storing into system-protected storage.

<u>RESTRICTIONS ON CMS</u>: There are several coding restrictions which must be imposed on CMS if it is to run as a saved system.

The first and most obvious one is that CMS may never modify segment 1, the shared segment, which runs with a RSK of 0, although the VSK = F.

A less obvious, but just as important, restriction, is that CMS may never modify with a single machine instruction (except MVCL) a section of storage which crosses the boundary between two pages with different storage keys. This restriction applies not only to SS instructions, such as MVC and ZAP, but also to RS instructions, such as STM, and to RX instructions, such as ST and STD, which may have nonaligned addresses on the System/370. An exception is the MVCL instruction which can be restarted after crossing a page boundary because the registers are updated when the paging exception occurs.

This restriction also applies to I/O instructions. If the key specified in the CCW is zero, then the data area for input may not cross the boundary between two pages with different storage keys.

OVERHEAD: It can be seen that this system is most inefficient when "storage-key thrashing" occurs -- when the virtual machine with a VPK of 0 jumps around, storing into pages with different VSK's.

ERROR CODES FROM DMSFREE, DMSFRES, AND DMSFRET

A nonzero return code, upon return from DMSFRES, DMSFREE or DMSFRET, indicates that the request could not be satisfied. Register 15 contains this return code, indicating which error has occurred. The codes below apply to the DMSFRES, DMSFREE and DMSFRET macros.

Code	Error
1	DMSFREE -- Insufficient storage space is available to satisfy the request for free storage . In the case of a variable request, even the minimum request could not be satisfied.
2	DMSFREE or DMSFRET -- User storage pointers destroyed.
3	DMSFREE or DMSFRET -- Nucleus storage pointers destroyed.
4	DMSFREE -- An invalid size was requested. This error exit is taken if the requested size is not greater than zero. In the case of variable requests, this error exit is taken if the minimum request is greater than the maximum request. However, the error is not detected if DMSFREE is able to satisfy the maximum request.
5	DMSFRET -- An invalid size was passed to the DMSFRET macro. This error exit is taken if the specified length is not positive.
6	DMSFRET -- The block of storage which is being released was never allocated by DMSFREE. Such an error is detected if one of the following errors is found: a. The block is not entirely inside either the free storage area in low storage or the user program area between FREELOWE and FREEUPPR. b. The block crosses a page-boundary which separates a page allocated for user storage from a page allocated for nucleus type storage. c. The block overlaps another block already on the free storage chain.
7	DMSFRET -- The address given for the block being released is not a doubleword boundary.
8	DMSFRES -- An illegal request code was passed to the DMSFRES routine. Because all request codes are generated by the DMSFRES macro, this error code should never appear.

9 DMSFRE, DMSFRET, or DMSFRES -- An unexpected internal error occurred.

THE DMSFRES MACRO

CMS uses the DMSFRES macro to request special internal free storage management services. Use of this macro by non-system routines causes unpredictable results. The format is:

```
┌──────────────────────────────────────────────┐
|   label  | DMSFRES |   option                 |
└──────────────────────────────────────────────┘
```

where "option" is one of the following:

INIT1 Performs the CMS system first initialization routine.

INIT2 Performs the CMS system second initialization routine.

CHECK Invokes a routine that checks the validity of all current free storage management pointers.

CKON Sets a flag that causes the CHECK to be invoked for each call to DMSFREE or DMSFRET.

CKOFF Turns off the above flag.

UREC Assists ABEND recovery, by releasing all USER-type DMSFREE storage allocations.

CALOC Assist ABEND recovery, by computing the total amount of allocated storage, excluding the system disk MFD and the FREETAB table.

For a full discussion of the meanings of these options, refer to "DMSFRE Service Routines."

THE DMSKEY MACRO

CMS uses the DMSKEY macro to modify the PSW storage protection key so that the nucleus code can store data into protected storage. The format is:

```
┌──────────────────────────────────────────────┐
| [label] | DMSKEY | {NUCLEUS[,NOSTACK]|         |
|         |        |  USER[,NOSTACK]|           |
|         |        |  LASTUSER[,NOSTACK]|       |
|         |        |  RESET}                    |
└──────────────────────────────────────────────┘
```

<u>where</u>:

NUCLEUS The nucleus storage protection key is placed in the PSW, and the old contents of the second byte of the PSW is saved in a stack. Use of this option allows the program to store into system storage, which is ordinarily protected.

USER The user storage protection key is placed in the PSW, and the old contents of the second byte of the PSW is saved in a stack. Use of this option prevents the program from inadvertently modifying nucleus storage, which is protected.

LASTUSER	The SVC handler traces back through its system save areas for the active user routine closest to the top of the stack, and the storage key in effect for that routine is placed in the PSW. The old contents of the second byte of the PSW is saved in a stack. This option should be used only by system routines that should enter a user exit routine.
NOSTACK	This option may be used with any of the above options to prevent the system from saving the second byte of the current PSW in a stack. If this is done, then no DMSKEY RESET need be issued later.
RESET	The second byte of the PSW is changed to the value at the top of the PSW key stack, and removed from the stack. Thus, the effect of the last DMSKEY NUCLEUS or USER or LASTUSER request is reversed. This option should may not be used to reverse the effect of a DMSKEY macro for which the NOSTACK option was specified. A DMSKEY RESET macro must be executed for each DMSKEY NUCLEUS, USER or LASTUSER macro that was executed and that did not specify the NOSTACK option. Failure to observe this rule results in program abnormal termination.

THE DMSEXS MACRO

System commands running in user protect status use the DMSEXS macro to execute a single instruction with a system protect key in the PSW. This macro instruction can be used in lieu of two DMSKEY macros. The format is:

[label]	DMSEXS	op-code,operands

The op-code and the operands of the instruction to be executed must be given as arguments to the DMSEXS macro.

For example, execution of the sequence,

```
USING  NUCON,0
DMSEXS OI,OSSFLAGS,COMPSWT
```

would cause the OI instruction to be executed with a zero protect key in the PSW. This sequence would turn on the COMPSWT flag in the nucleus. It would be reset with

```
DMSEXS NI,OSSFLAGS,255-COMPSWT
```

The instruction to be executed may be an EX instruction.

Register 1 cannot be used in any way in the instruction being executed.

Simulate Non-CMS Operating Environments

The following contains descriptions for: access method support for non-CMS operating systems, CMS simulation of OS functions, and CMS implementation of DOS/VS functions.

Access Method Support for Non-CMS Operating Environments

OS ACCESS METHOD SUPPORT

An access method governs the manipulation of data. To make the execution of OS generated code easier under CMS, the processing program must see data as OS would present it. For instance, when the processors expect an access method to acquire input source records sequentially, CMS invokes its sequential access method and passes data to the processors in the format that the OS access methods would have produced. Therefore, data appears in storage as if it had been manipulated using an OS access method. For example, block descriptor words (BDW), buffer pool management, and variable records are maintained in storage as if an OS access method had processed the data. The actual writing to and reading from the I/O device is handled by CMS file management.

The work of the volume table of contents (VTOC) and the data set control block (DSCB) is done by a master file directory (MFD) to maintain disk contents and a file status table (FST) for each data file. All disks are formatted in physical blocks of 800 bytes.

CMS continues to maintain the OS format, within its own format, on the auxiliary device, for files whose filemode number is 4. That is, the block and record descriptor words (BDW and RDW) are written along with the data. If a data set consists of blocked records, the data is written to and read from the I/O device in physical blocks, rather than logical records. CMS also simulates the specific methods of manipulating data sets.

To accomplish this simulation, CMS supports certain essential macros for the following access methods:

- BDAM (direct)--identifying a record by a key or by its relative position within the data set.
- BPAM (partitioned)--seeking a named member within an entire data set.
- BDAM/QSAM (sequential)--accessing a record in a sequence relative to
- VSAM (direct or sequential)--accessing a record sequentially or directly by key or address. CMS support of OS VSAM files is based on DOS/VS access method services and the virtual storage access method (VSAM). Therefore, the OS user is restricted to those services available under DOS/VS AMS and VSAM.

CMS Support for the Virtual Storage Access Method

CMS simulation of OS and DOS includes support for the virtual storage access method (VSAM). The description of this support is in three parts:

- A description of the access method services program (AMSERV), which allows you to create and update VSAM files.
- A description of support for VSAM functions under CMS/DOS.
- A description of support for VSAM functions for the CMS OS simulation routines.

  The routines that support VSAM reside in three discontiguous shared segments (DCSSs).

  -- The CMSAMS DCSS, which contains the DOS/VS AMS code to support AMSERV processing.

  -- The CMSVSAM DCSS, which contains actual DOS/VS VSAM code, and the CMS/VSAM OS interface program for processing OS VSAM requests.

  -- The CMSDOS DCSS, which contains the code that supports DOS requests under CMS.

Note: DMSVSR, which performs completion processing for CMS/VSAM support, resides in the CMS nucleus.

CREATING THE DOSCB CHAIN

The DLBL command creates a control block called a DOSCB in CMS free storage. The ddname specified in this DLBL command is associated with the ddname parameter in the program's ACB.

The DOSCB contains information defining the file for the system. The information in the DOSCB parallels the information written on the label information cylinder of a real DOS SYSRES unit, e.g. the name, and mode (volume serial number) of the data set, its logical unit specification, and its data set type (SAM or VSAM). The anchor for this chain is at location DOSFIRST in NUCON.

Executing an AMSERV Function

The CMS AMSERV command invokes the module DMSAMS, which is the CMS interface to the DOS/VS access method services (AMS) program. Module DMSAMS loads DOS/VS AMS code contained in the CMSAMS DCSS by means of the LOADSYS DIAGNOSE 64. The AMS code requires the services of DOS/VS code that resides in the CMSVSAM DCSS so that DCSS is also loaded via LOADSYS DIAGNOSE 64 when the VSAM master catalog is opened. Figure 19 shows the relationship in storage between the interface module DMSAMS and the CMSAMS and CMSVSAM DCSSs.

The following is a general description of the DMSAMS method of operation.

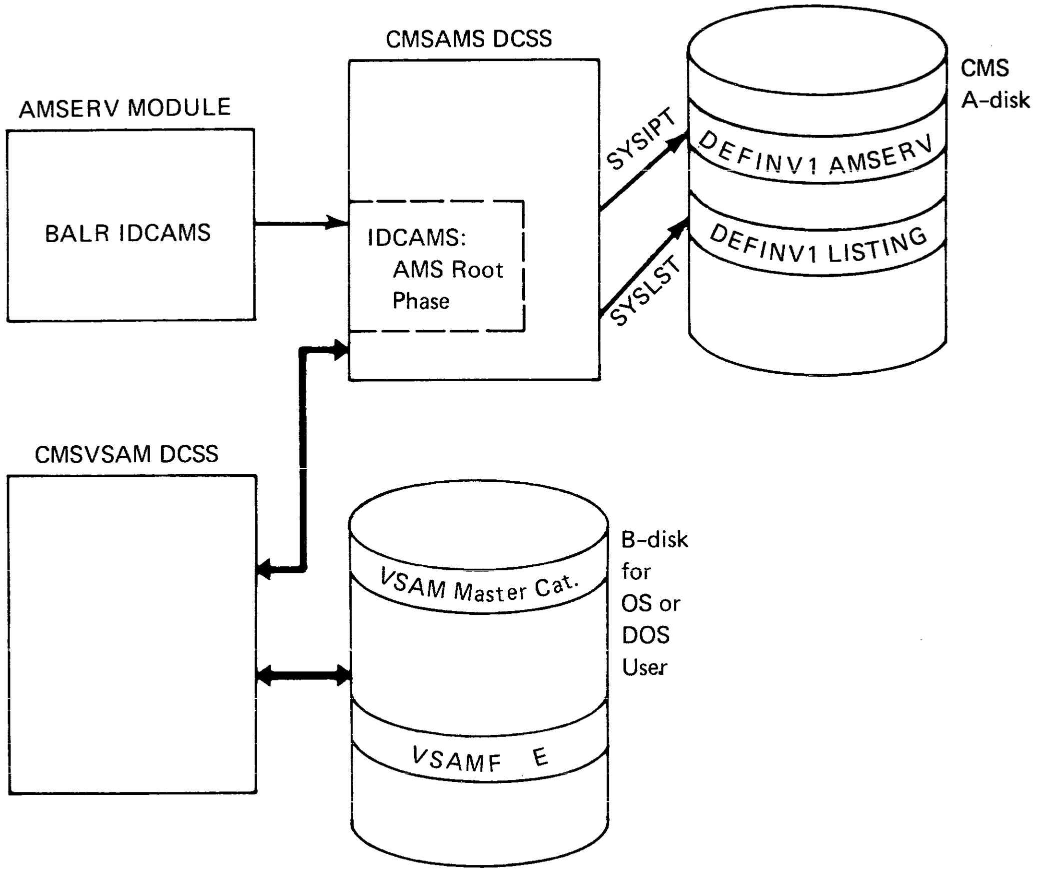


Figure 19. Relationship in Storage between the CMS Interface Module DMSAMS and the CMSAMS and CMSVSAM DCSSs

DMSAMS first determines whether the user is in the CMS/DOS environment. If not, a SET DOS ON (VSAM) command is issued to load the CMSDOS segment and initialize the CMS/DOS environment. In this case, DMSAMS must also issue ASSGN commands for the disk modes in the DOSCB chain created by the OS user's DLBL commands. An ASSGN is also issued for SYSCAT, the VSAM master catalog.

DMSAMS then issues the ASSGN command for the SYSIPT and SYSLST files, assigning them to the user's A-disk. DLBL commands are then issued associating these units with files on the user's A-disk. Input to the AMSERV processor is the SYSIPT file, which has the filetype AMSERV. Output from AMSERV processing is placed in the SYSLST file, which has a filetype of LISTING.

DIAGNOSE 64 (LOADSYS) is then issued to load the CMSAMS DCSS, which contains the DOS/VS AMS code. A DOS/VS SVC 65 is issued to find the address of the DOS/VS AMS root phase, IDCAMS. When the SVC returns with the address of IDCAMS, a branch is made to IDCAMS, giving control to "live" DOS/VS routines.

IDCAMS expects parameters to be passed to it when it receives control. DMSAMS passes dummy parameters in the list labeled AMSPARMS.

After the root phase IDCAMS receives control, the functions in the file specified by the filename on the AMSERV command are executed.

In performing the functions requested in this file, AMS may require execution of DOS/VS VSAM phases located in the CMSVSAM DCSS. The CMSVSAM DCSS is loaded when AMS opens the VSAM catalog for processing.

On return from DOS/VS code, DMSAMS purges the CMSAMS DCSS, and issues DLBL commands for the SYSIPT and SYSLST files to clear the DOSCB's for these ddnames.

Control is then passed to DMSVSR, which purges the CMSVSAM DCSS. If the user program was not in the CMS/DOS environment when DMSAMS was entered, the SET DOS OFF command is issued by DMSVSR. Upon return from DMSVSR, DMSAMS performs minor housekeeping tasks and returns control to CMS.

Executing a VSAM Function for a DOS User

When a VSAM function, such as an OPEN or CLOSE macro, is requested from a DOS program, CMS routes control through the CMSDOS DCSS to the CMSVSAM DCSS, thus giving control to DOS/VS VSAM phases. Figure 20 shows the relationships in storage between the user program, the CMSDOS DCSS, and the CMSVSAM DCSS. The description below illustrates the overall logic of that control flow.

CMS/DOS SVC HANDLING

There are four CMS/DOS routines that handle VSAM requests: DMSDOS, DMSBOP, DMSCLS, and DMSXCP. Within DMSDOS, several SVC functions support VSAM requests. These are described in "Simulating a DOS Environment Under CMS."

DMSDOS VSAM Processing

DMSDOS VSAM processing involves handling of SVC 65 (CDLOAD), which returns the address of a specified phase to the caller. DMSDOS searches both the shared segment table and the nonshared segment table for the CMSDOS and CMSVSAM segments, because both could be in use. Both of these segment tables contain the name of each phase comprising that segment followed by the fullword address of that phase within the segment.

During SVC 65 processing, DMSDOS checks to see if the address of IKQLAB is being requested. IKQLAB is the VSAM routine that returns the label information generated by DLBLs and EXTENT cards in DOS/VS systems. If this is the case, DMSDOS saves the address of IKQLAB in NUCON for later use by DMSXCP.

If VSAM has not been loaded, a DIAGNOSE 64 (LOADSYS) is issued to load the CMSVSAM DCSS.

DMSBOP VSAM Processing

When DMSBOP is entered to process ACBs, it checks to see if CMSVSAM is loaded. If VSAM has not been loaded, DIAGNOSE 64 is issued to load the

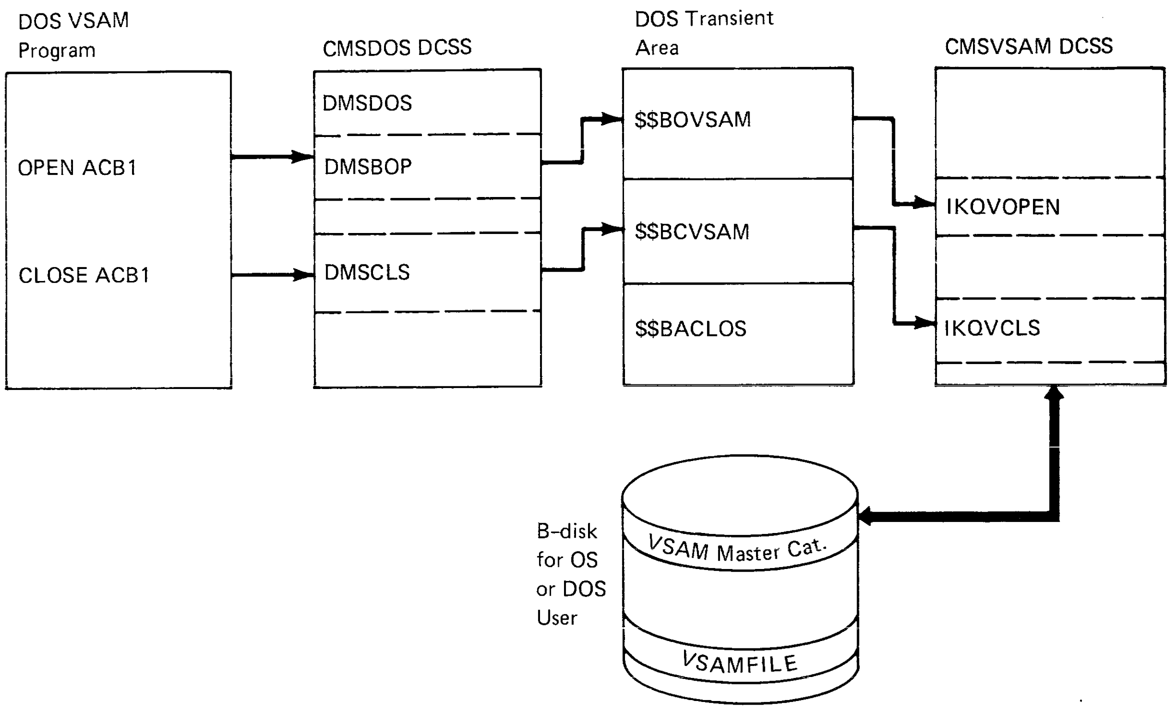


Figure 20. The Relationships in Storage between the User Program and the CMSDOS and CMSVSAM DCSSs

CMSVSAM DCSS. DMSBOP then initializes the transient work area and issues a DOS OPEN via SVC 2 to bring the VSAM OPEN $$BOVSAM transient into the DOS transient area.

When VSAM processing completes, control returns to the user program directly.

DMSCLS VSAM Processing

DMSCLS processing is nearly the same as processing for DMSBOP. When DMSCLS is entered, it checks for an ACB to process. If there is one, the $$BCVSAM transient work area is initialized and SVC 2 is issued to FETCH the VSAM CLOSE transient $$BCVSAM into the DOS transient area. When the VSAM CLOSE routines complete processing, control returns to the user program, as in the case of OPEN.

DMSXCP VSAM Processing

When DMSXCP processes an EXCP request, it determines if the request is from IKQLAB (that is, to read the SYSRES label information). If so, the label information area record is filled in from the appropriate DOSCB. (DMSXCP determines that the caller is IKQLAB by comparing the address of the caller with the address stored in NUCON by DMSDOS, as described above.)

Executing a VSAM Function for an OS User

OS user requests for VSAM services are handled by DOS/VS VSAM code that resides in the CMSVSAM DCSS. To access this code, OS VSAM requests are intercepted by the CMS module DMSVIP, the interface between the OS VSAM requests and the CMS/DOS and DOS/VS VSAM routines.

Because DMSVIP is in the CMSVSAM segment, it is available only when that segment is loaded. Module DMSVIB, which resides in the CMS nucleus, is a bootstrap routine to load the CMSVSAM segment and pass control to DMSVIP.

DMSVIP receives control from VSAM request macros in three ways: via SVC (e.g. OPEN and CLOSE), via a direct branch using the address of DMSVIP in the ACB, and via a direct branch to the location of DMSVIP whose address is 256 bytes into the CMSCVT (CMSCVT is a CMS control block that simulates the OS CVT control block).

This last technique is used by the code generated from the OS VSAM control block manipulation macros (GENCB, SHOWCB, TESTCB, MODCB). That is, the address at 256 into CVT is assumed to be that of a control block that is at displacement X'12' has the address of the VSAM control block manipulation routine. To ensure that DMSVIP receives control from these requests, the address of DMSVIP is stored at 256 bytes into CMSCVT. However, until the CMSVSAM segment is loaded, the address at CMSCVT+256 is the address of module DMSVIB rather than the address of DMSVIP. The address of DMSVIP replaces that of DMSVIB when CMSVSAM is loaded. Both DMSVIB and DMSVIP have pointers to themselves at 12 bytes into themselves to ensure that this technique works.

Figure 21 shows the relationships in storage between the user program, the OS simulation and interface routines, and the CMSDOS and CMSVSAM DCSSs.

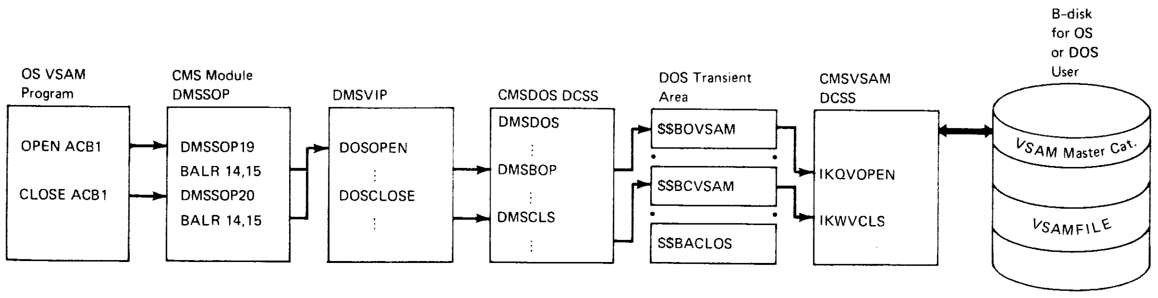


Figure 21. Relationship in Storage between the User Program, the CS Simulation and Interface Routines, and the CMSDOS and CMSVSAM DCSSs

The following description illustrates the overall logic of that control flow.

DMSVIP Processing

DMSVIP gains control from DMSSOP when an OS SVC 19, 20 or 23 (CLOSE TYPE=T) is issued. It also gains control on return from execution of a VSAM function, as described below. DMSVIP performs five main functions:

- Initializes the CMS/DOS environment for OS VSAM processing.
- Simulates an OS VSAM OPEN macro.
- Simulates an OS VSAM CLOSE macro.
- Simulates an OS VSAM control block manipulation macro (GENCB, MODCB, SHOWCB, or TESTCB).
- Processes OS VSAM I/O macros.

Initializing the CMS/DOS Environment for OS VSAM Processing

DMSVIP gets control when the first VSAM macro is encountered in the user program. Initialization processing begins at this time. The CMSDCS DCSS is loaded by issuing the command SET DOS ON (VSAM). ASSGN commands are also issued at this time according to the user-issued DLBL's as indicated in the DOSCB chain. Once this initialization completes, DMSVIP processes the VSAM request.

After the initialization, DMSVIP first checks to determine which VSAM function is being requested, OPEN, CLOSE, or a control block manipulation macro.

Simulate an OS VSAM OPEN

For OPEN processing, the DOSSVC bit in NUCON is set on and control passes to DMSBOP via SVC 2. Once the CMS/DOS routines are in control, execution of the VSAM function is the same as for the DOS VSAM functions described above.

On return from executing the OPEN routine, the address of another entry point to DMSVIP, at label DMSVIP2, is placed in the ACB for the data set just opened, the DOSSVC bit is turned off, and control is passed to DMSSOP, which returns to the user program. DMSVIP2 is the entry point for code that performs linkage to the VSAM data management phase IKQVSM. This is done after the first OPEN because it is assumed that, once opened, the user performs I/O for the phase, e.g., a GET or PUT operation.

When the linkage routine is entered, the DOSSVC bit is set on and control is given to the VSAM data management routine IKQVSM. On return from IKQVSM DMSVIP turns off the DOSSVC bit and returns control to the user program. (Refer to Simulate OS VSAM I/O Macros in this section.)

Simulate an OS VSAM CLOSE

For CLOSE processing, the DOSSVC bit is set on and control is passed to the CMS/DOS routine DMSCLS via SVC 2. As in the case of OPEN, once control passes to the CMS/DOS routine, execution of the VSAM function is the same as for the DOS VSAM functions described above.

On return from executing the VSAM CLOSE, the DOSSVC bit is turned off and control passes to DMSSOP, which returns to the user program.

Simulate OS VSAM Control Block Manipulation Macros

DMSVIP simulates the GENCB, MODCB, SHOWCB, and TESTCB control block manipulation macros.

GENCB PROCESSING: When a GENCB macro is issued with BLK=ACB or BLK=EXLST specified, the GENCB PLIST is passed unmodified to IKQGEN for execution. If GENCB is issued with BLK=RPL and ECB=address specified, the PLIST is rearranged to exclude the ECB specification, because DOS/VS does not support ECB processing. The GENCB PLIST is then passed to IKQGEN for execution.

MODCB, SHOWCB, AND TESTCB PROCESSING: When MODCB, SHOWCB, or TESTCB is issued, the OS ACB, RPL, and EXLST control blocks are reformatted, if necessary, to conform to DOS/VS formats.

For MODCB and SHOWCB, the requests are passed to IKQTMS for processing. When MODCB is issued with EXLST= specified, ensure that the exit routines return control to entry point DMSVIP3.

For TESTCB, check for any error routines the user may have specified. If the TESTCB specified RPL= and IO=COMPLETE, a not equal result is passed to the user. All other TESTCB requests are passed to DOS and the new PSW condition code indicates the results of the test.

If an error return is provided for TESTCB, the address of DMSVIP4 is substituted in the PLIST. This allows DMSVIP to regain control from VSAM so that the DOSSVC bit can be turned off. The error routine is then given control after the address is returned to the PLIST.

Simulate OS VSAM I/O Macros

DMSVIP simulates the OS GET, PUT, POINT, ENDREQ, ERASE, and CHECK I/O macros.

GET, PUT, POINT, ENDREQ, and ERASE Processing:

First, the OS request code in register 0 is mapped to a DOS/VS request code. The RPL or chain of RPLs is rearranged to DOS format (unless that has already been done).

If there is an ECB address in the OS RPL, a flag is set in the new DOS RPL and the ECB address is saved at the end of the RPL.

Asynchronous I/O processing is simulated by setting active exit returns inactive in the user EXLST. The exception to this is the JRNAD exit which need not be set inactive since it is not an error exit. Setting error exits to be inactive prevents VSAM from taking an error exit, thus allowing such an exit to be deferred until a CHECK can be issued for it.

The DOS macro is then issued via a BALR to IKQVSM.

DOS error codes returned in the RPL FDBK field that do not exist in OS are mapped to their OS equivalents. If the user has specified synchronous processing, this return code is passed unchanged in register 15.

For asynchronous processing, return codes are cleared before return and any exit routines set inactive are reactivated in the EXLST. Also, all ECBs are set to WAITING status.

CHECK PROCESSING: For CHECK processing, return codes in the RPL FDBK field are checked to determine the results of the I/O operation. If there is an active exit routine provided for the return code, control is passed to that routine. Also, all WAITING ECBs are posted with an equivalent completion code.

If no active exit routine is provided or if the exit routine returns to VSAM, the return code is placed in register 15 and control is returned to the instruction following the CHECK.

CMS/VSAM Error Return Processing

Two types of support for error routine processing are provided in DMSVIP. Entry point DMSVIP3 provides support for user exit routines; entry point DMSVIP4 provides support for ERET error returns.

USER EXIT ROUTINE PROCESSING: DMSVIP provides support for OS VSAM I/O error exits at entry point DMSVIP3. At this entry point the DOSSVC bit is turned off and the user storage key is restored.

The address of the user routine is recovered from VIP's saved exit list (either the primary exit list in the work area or the overflow exit list, OEXLSA).

Control then passes to the appropriate exit routine. If the routine is one that returns to VSAM, the DOSSVC flag is set ON and VSAM processing continues.

DMSVIP can save the addresses of up to 128 exit routines during execution of a user program.

ERET ERROR ROUTINE PROCESSING: DMSVIP provides support for OS VSAM ERET exit routines used in conjunction with the TESTCB macro. This support is located at entry point DMSVIP4. At DMSVIP4, the DOSSVC bit is turned off and the user storage key is restored. The address of the ERET routine is recovered from the work area and control passes to that routine.

The ERET routine may not return control to VSAM.

COMPLETION PROCESSING FOR OS AND DOS VSAM PROGRAMS

When an OS or DOS VSAM program completes, control is passed to module DMSVSR, which "cleans up" after VSAM. DMSVSR can be called from three routines after OS processing:

- DMSINT, if processing completes without system errors or serious user errors.

- DMSEXT, if the user program is used as part of an EXEC file.
- DMSABN, if there are system errors or the user program abnormally terminates.

After DOS VSAM processing completes, DMSVSR is called by DMSDOS.

DMSVSR issues an SVC 2 to execute the DOS transient routine $$BACLOS. $$BACLOS first checks for any OPEN VSAM files. If any are open, SVC 2 is issued to $$BCLOSE (DMSCLS) to close the files.

If there are no open files or if all ACB's have been closed, $$BACLOS issues SVC 2 to $$BEOJ4, an entry point in DMSVSR. At $$BEOJ4, a PURGESYS DIAGNOSE 64 is issued to purge the CMSVSAM DCSS. DMSVSR then checks to see if an OS program has completed processing. If this is the case, the SET DOS OFF command is issued and control returns to the caller.

OS Simulation by CMS

When in a CMS environment, a processor or a user-written program is executing and utilizing OS-type functions, OS is not controlling this action, CMS is in control. Consequently, it is not OS code that is in CMS, but routines to simulate, in terms of CMS, certain OS functions essential to the support of OS language processors and their generated code.

These functions are simulated to yield the same results as seen from the processing program, as specified by OS program logic manuals. However, they are supported only to the extent stated in CMS documentation and to the extent necessary to successfully execute OS language processors. The user should be aware that restrictions to OS functions as viewed from OS exist in CMS.

Certain TSO Service routines are provided to allow the Program Products to run under CMS. The routines are the Command Scan and Parse Service Routines and the Terminal I/O Service Routines. In addition the user must provide some initialization as documented in TSO TMP Service Routine initialization. The OS functions that CMS simulates are shown in Figure 22.

TSO Service Routine Support

TSO macros that support the use of the terminal monitor program (TMP) service routines are contained in TSOMAC MACLIB. The macro functions are as described in the TSO TMP documentation with the exception of PUTLINE, GETLINE, PUTGET, and TCLEARQ.

Before using the TSO service routines, the calling program performs the following initialization:

1. Stores the address of the command line as the first word in the command processor parameter list (CPPL). The TSOGET macro puts the address of the CPPL in register 1.
2. Initializes CMS storage using the STRINIT macro.
3. Clears the ECT field that contains the address of the I/O work area (ECTIOWA).

SVC Number	OS Macro Function	Simulation Routine	Comments
00	XDAP	DMSSVT	Reads or writes direct access volumes
01	WAIT	DMSSVN	Waits for an I/O completion
02	POST	DMSSVN	Posts the I/O completion
03	EXIT	DMSSLN	Returns from linked phase
04	GETMAIN	DMSSMN	Conditionally acquires user free storage
05	FREEMAIN	DMSSMN	Releases user-acquired free storage
06	LINK	DMSSLN	Links control to another load phase
07	XCTL	DMSSLN	Deletes, then links control to another load phase
08	LOAD	DMSSLN	Reads another load phase into storage
09	DELETE	DMSSLN	Deletes a loaded phase
10	GETMAIN/ FREEMAIN	DMSSMN	Manipulates free user storage
	GETPOOL	DMSSMN	Simulates an SVC10
11	TIME	DMSSVT	Gets the time of day
13	ABEND	DMSSAB	Terminates processing
14	SPIE	DMSSVT	Processes program interruptions
17	RESTORE	DMSSVT	Effective NOP
18	BLDL/FIND	DMSSVT	Manipulates simulated partitioned data files
19	OPEN	DMSSOP	Activates a data file
20	CLOSE	DMSSOP	Deactivates a data file
21	STOW	DMSSVT	Manipulates partitioned directories
22	OPENJ	DMSSOP	Activates a data file
23	TCLOSE	DMSSOP	Temporarily deactivates a data file
24	DEVTYPE	DMSSVT	Obtains device-type physical characteristics
25	TRKBAL	DMSSVT	Effective NOP
31	FEOV	DMSSVT	Set forced EOV error code
35	WTO/WTOR	DMSSVT	Communicates with the terminal
40	EXTRACT	DMSSVT	Effective NOP
41	IDENTIFY	DMSSVT	Adds entry to loader table
42	ATTACH	DMSSVT	Effective LINK
44	CHAP	DMSSVT	Effective NOP
46	TTIMER	DMSSVT	Accesses or cancels timer
47	STIMER	DMSSVT	Sets timer interval and timer exit routine
48	DEQ	DMSSVT	Effective NOP
51	SNAP	DMSSVT	Dumps specified storage areas
56	ENQ	DMSSVT	Effective NOP
57	FREEDBUF	DMSSVT	Releases a free storage buffer
60	STAE	DMSSVT	Allows processing program to decipher abend condition
62	DETACH	DMSSVT	Effective NOP
63	CHKPT	DMSSVT	Effective NOP
64	RDJFCB	DMSSVT	Obtains information from FILEDEF command
68	SYNAD	DMSSVT	Handles data set error conditions
69	BACKSPACE	DMSSVT	Backs up to the beginning of the previous record
–	GET/PUT	DMSSQS	Manipulates data records
–	READ/WRITE	DMSSBS	Manipulates data blocks
–	NOTE/POINT	DMSSCT	Accesses or changes relative track address
–	CHECK	DMSSCT	Tests ECB for completion and errors
93	TGET/TPUT	DMSSVN	Terminal processing
94	TCLEARQ	DMSSVN	Clears input queue
96	STAX	DMSSVT	Adds or deletes an attention exit level

Figure 22. OS Functions that CMS Simulates

4. Issues the STACK macro to define the terminal as the primary source of input.

CMS Simulation of OS Control Block Functions

Most of the simulated supervisory OS control blocks are contained in the following two CMS control blocks:

CMSCVT	simulates the communication vector table (CVT). Location 16 contains the address of the CVT control section.
CMSCB	allocated from system free storage whenever a FILEDEF command or an OPEN (SVC 19) is issued for a data set. The CMS control block consists of the CMS file Control block (FCB) for the data file management under CMS, and simulation of the job file control block (JFCB), input/output block (IOB), and data extent block (DEB). The name of the data set is contained in the FCB, and is obtained from the FILEDEF argument list, or from a predetermined file name supplied by the processing problem program.

CMS also utilizes portions of the supplied data control block (DCB) and the data event control block (DECB). The TSO control blocks utilized are the command program parameters list (CPPL), user profile table (UPT), protected step control block (PSCB), and environment control table (ECT).

Operating System Simulation Routines

CMS provides a number of routines to simulate certain operating system functions used by programs such as the Assembler and the FORTRAN and PL/I compilers. Some of the SVC simulation routines are located in the disk resident transient module DMSSVT. Whenever one of the SVC routines in DMSSVT or is invoked, that routine is loaded into the transient area. The following paragraphs describe how these simulation routines work.

XDAP-SVC 0: Writes and reads the source code spill file, SYSUT1, during language compilation for PL/I Optimizer and ANS COBOL Compilers.

WAIT-SVC 1: Causes the active task to wait until one of more event control blocks (ECBs) have been posted. For each specified ECB that has been posted one is subtracted from the number of events specified in the WAIT macro. If the number of events is zero by the time the last ECB is checked control is returned to the user. If the number of events is not zero after the last ECB is checked and the number of events is not greater than the number of ECBs, the active task is put into a wait state until enough ECBs are posted to set the number of events at zero. When the event count reaches zero the wait bits are turn off in any ECBs that have not been posted and control is returned to the user. If the number of events specified is greater than the number of ECBs the system abnormally terminates with an error message. All options of WAIT are supported.

POST-SVC 2: Causes the specified event control block (ECB) to be set to indicate the occurrence of an event. This event satisfies the requirements of a WAIT macro instruction. All options of POST are supported. The bits in the ECB are set as follows:

Bit	Setting
0	0
1	1
2-7	Value of specified completion code

EXIT-SVC 3: This SVC is for CMS internal use only. It is used by the CMS routine DMSSLN to acquire an SVC SAVEAREA on return from an executing program that had been given control by LINK (SVC 6), XTCL (SVC 7) or ATTACH (SVC 42).

GETMAIN-SVC 4: Control is passed to the GETMAIN entry point in the DMSSMN storage resident routine. The mode is determined: VU, VC, EC. A call is made to GETBLK to obtain the block of storage. Control blocks of two fullwords precede each section of available storage: (1) the address of the next block, (2) the size of this block. The head of the pointer string is located at the words MAINSTRT - initial free block, and MAINLIST - address of first link in chain of free block pointers. All options of GETMAIN are supported.

FREEMAIN-SVC 5: Releases a block of free storage. If the block is part of segmented storage, a control block of two fullwords is placed at the beginning of the released area. Adjustment is made to include this block in the chain of available areas. All options of FREEMAIN are supported.

LINK-SVC 6: Program transfer is controlled by the nucleus routine, DMSSLN. The LINK macro causes program control to be passed to a designated phase. If the COMPSWT bit within the byte OSSFLAGS is on, loading is done by calling LOADMOD to bring a CMS MODULE file into storage. If this flag is off, dynamic loading is initiated by calling LOAD. A GETMAIN is issued to obtain enough storage so that the loader (DMSLDR) may relocate the phase in storage. A chain of link request blocks is built to record the old SVC PSW, and the location and size of the phase storage area. If the routine is already in storage, determined by scanning the load request chain, no LOAD or LOADMOD is done. Control is passed directly to the routine. CMS ignores the DCB and HIARCHY options; all other options of LINK are supported.

XCTL-SVC 7: XCTL first deletes the current phase from storage. Processing then continues as for LINK-SVC 6, as previously described. CMS ignores The DCB and HIARCHY options; all other options of XCTL are supported.

LOAD-SVC 8: Control is passed to DMSSLN8 located in DMSSLN when a LOAD macro is issued. If the requested phase is not in storage, a LOAD or LOADMOD is issued to bring it in. Control is then returned to the caller. CMS ignores the DCB and HIARCHY options; all other options of LOAD are supported.

DELETE-SVC 9: Control is passed to DMSSLN9 located in DMSSLN when a DELETE macro is issued. Upon entry, DELETE checks to see whether the module specified was loaded using LOADMOD or dynamically loaded by LOAD or INCLUDE. If it was loaded by LOADMOD control is returned to the user. If it was dynamically loaded, the responsibility count is decremented by one and if it reaches zero, the storage is released using FREEMAIN, and control is returned to the user. All options of DELETE are supported. Code 4 is returned in register 15 if the phase is not found.

GETMAIN/FREEMAIN-SVC 10: Control is passed to the SVC 10 entry point in DMSSMN. Storage management is analogous to SVC 4 and 5, respectively. All options of GETMAIN and FREEMAIN are supported. Subpool specifications are ignored.

<u>GETPOOL</u>: Gets control via an OS LINK macro to IECQBFGI. IECQBFGI allocates an area of free storage using GETMAIN, sets up a buffer control block in the free storage, stores the address of the buffer control block in the DCB, and then returns control to the caller.

<u>TIME-SVC 11</u>: This routine (TIME) located in DMSSVT receives control when a TIME macro instruction is issued. A call is made (by SIO or DIAGNOSE) to the RPQ software chronological timer device, X'OFF'. The real time of day and date are returned to the calling program in a specified form: decimal (DEC) binary (BIN), or timer units (TU). All options of TIME except hundredths of a second MIC are supported.

<u>ABEND-SVC 13</u>: This routine (DMSSAB) receives control when either an ABEND macro or an unsupported OS/360 SVC is issued. If an SVC 13 was issued with the DUMP option and either a SYSUDUMP or SYSABEND ddname had been defined via a call to DMSFLD (FILEDEF), a SNAP (SVC 51) specifying PDATA=ALL is issued to dump user storage to the defined file. A check is made to see if there are any outstanding STAE requests. If not, or if an unsupported SVC was issued, DMSCWR is called to type a descriptive error message at the terminal. Next, DMSCWT is called to wait until all terminal activity has ceased, and then, control is passed to the ABEND recovery routine. If a STAE macro was issued, a STAE work area is built and control is passed to the STAE exit routine. After the exit routine is complete, a test is made to see if a retry routine was specified. If so, control is passed to the retry routine. Otherwise, control passes to DMSABN unless the task that had the ABEND was a subtask. In that case, the resume PSW in the link block for the subtask is adjusted to point to an EXIT instruction (SVC 3). The EXIT frees the subtask, and the attaching task is redispatched.

<u>SPIE-SVC 14</u>: This routine (SPIE) receives control when a SPIE macro instruction is issued. When it gets control, SPIE inserts the new program interruption control area (PICA) address into the program interruption element (PIE). The program interruption element resides in the program interruption handler (DMSITP). It then returns the address of the old PICA to the calling program, sets the program mask in the calling program's PSW, and returns to the calling program. All options of SPIE are supported.

<u>RESTORE-SVC 17</u>: RESTORE is a NOP located in DMSSVT.

<u>BLDL/FIND(Type D)-SVC 18</u>: SVC to entry points in DMSSOP. If an OS disk is specified, DMSSVT branches and links to DMSROS. See BLDL and FIND under description of BPAM routines in DMSSVT.

<u>STOW-SVC 21</u>: See STOW under description of BPAM routines in DMSSVT.

<u>OPEN/OPENJ-SVC 19/22</u>: OPEN simulates the data management function of opening one or more files. It is a nucleus routine and receives control from DMSITS when an executing program issues an OPEN macro instruction. The OPEN macro causes an SVC to DMSSOP. DMSSOP simulates the OPEN macro. The DISP and RDBACK options are ignored by CMS; all other options of OPEN and OPENJ are supported.

<u>CLOSE/TCLOSE-SVC 20/23</u>: CLOSE and TCLOSE are simulated in the nucleus routine DMSSOP. It receives control whenever a CLOSE or TCLOSE macro instruction is issued. The CLOSE macro causes an SVC to DMSSOP. DMSSCP simulates the CLOSE macro. CMS ignores the DISP option; all other options of CLOSE and TCLOSE are supported.

<u>DEVTYPE-SVC 24</u>: This routine (DEVTYPE), located in DMSSVT, receives control when a DEVTYPE macro is issued. Upon entry, DEVTYPE moves Device Characteristic Information for the requested data set into a user specified area, and then returns control to the user. All options of DEVTYPE are supported, except RPS, which is ignored.

TRKBAL-SVC 25: TRKBAL is a NOP located in DMSSVT.

FEOV-SVC 31: Returns control to CMS with an error code of 4 in register 15.

WTO/WTOR-SVC 35: This routine (WTO), located in DMSSVT, receives control when either a WTO or a WTOR macro instruction is issued. For a WTO, it constructs a calling sequence to the DMSCWR function program to type the message at the terminal. (The address of the message and its length are provided in the parameter list that results from the expansion of the WTO macro instruction.) It then calls the DMSCWT function program to wait until all terminal I/O activity has ceased. Next, it calls the DMSCWR function program to type the message at the terminal and returns to the calling program. All options of WTO and WTOR are supported except those concerned with multiple console support.

For a WTOR macro instruction, this routine proceeds as described for WTO. However, after it has typed the message at the terminal it calls the DMSCRD function program to read the user's reply from the terminal. When the user replies with a message, it moves the message to the buffer specified in the WTOR parameter list, sets the completion bit in the ECB, and returns to the calling program.

EXTRACT-SVC 40: This routine (EXTRACT), located in DMSSVT receives control when an EXTRACT macro is issued. Upon entry, EXTRACT clears the user provided answer area and returns control to the user with a return code of 4 in register 15.

IDENTIFY-SVC 41: Located in DMSSVT, this routine creates a new load request block with the requested name and address if both are valid. The new entry is chained from the existing load request chain. The new name may be used in a LINK or ATTACH macro.

ATTACH-SVC 42: Located in DMSSLN, ATTACH operates like a LINK (SVC 6), with additional capabilities. The user is allowed to specify an exit address to be taken upon return from the attached phase; also, an ECB is posted when the attached phase has completed; and a STAI routine can be specified in case the attached phase abends. The DCB, LPMOD, DPMOD, HIARCHY, GSPV, GSPL, SHSPV, SHSPL, SZERO, PURGE, ASYNCH, and TASKLIB options are ignored; all other options of ATTACH are supported. Because CMS is not a multitasking operating system, a phase requested by the ATTACH macro must return to CMS.

CHAP-SVC 44: CHAP is a NOP located in DMSSVT.

TTIMER-SVC 46: Checks to ensure that the value in the timer (hex location 50) was set by an STIMER macro. If it was, the value is converted to an unsigned 32 bit binary number specifying 26 microsecond units and is returned in register 0. If the timer was not set by an STIMER macro a zero is returned in register 0, after setting register 0, the CANCEL option is checked. If it is not specified, control is returned to the user. If it is specified, the timer value and exit routine set by the STIMER macro are cancelled and control is returned to the user. All options of TTIMER are supported.

STIMER-SVC 47: Checks to see if the WAIT option is specified. If so, control is returned to the user. If not, the specified timer interval is converted to 13 microsecond units and stored in the timer (hex location 50). If a timer completion exit routine is specified, it is scheduled to be given control after completion of the specified time interval. If not, no indication of the completion of the time interval is scheduled. After checking and handling any specified exit routine address, control is returned to the user. All options of STIMER are supported. The TASK option is treated as though the REAL option had been specified.

DEQ-SVC 48: DEQ is a NOP located in DMSSVT.

SNAP-SVC 51: Control is passed to SNAP in DMSSVT when a SNAP macro is issued. SNAP fills in a PLIST with a beginning and ending address and calls DMPEXEC. DMPEXEC dumps the specified storage along with the registers and low storage to the printer. Control is then returned to SNAP and SNAP checks to see if any more addresses are specified. It continues calling DMPEXEC until all the specified addresses have been dumped to the printer. Control is then returned to the user. Except for SDATA, PDATA, and DCB, all options of the SNAP macro are processed normally. SDATA and PDATA are ignored. Processing for the DCB option is as follows: The DCB address specified with SNAP is used to verify that the file associated with the DCB is open. If it is not open, control returns to the caller with a return code of 4. If the file is open, the FCB associated with the file is checked for a device type of DUMMY. If the device type is DUMMY, control returns to the caller with a return code of 0 and storage is not dumped.

ENQ-SVC 56: ENQ is a NOP located in DMSSVT.

FREEDBUF-SVC 57: This routine (FREEDBUF) located in DMSSVT receives control when a FREEDBUF macro is issued. Upon entry, FREEDBUF sets up the correct DSECT registers and calls the FREEDBUF routine in DMSSBD. This routine returns the dynamically obtained buffer (BDAM) specified in the DECB to the DCB buffer control block chain. Control is then returned to the DMSSVT routine which returns control to the user. All the options of FREEDBUF are supported.

STAE-SVC 60: This routine (STAE) located in DMSSVT receives control when a STAE macro is issued. Upon entry, STAE creates, overlays or cancels a STAE control block (SCB) as requested. Control is then returned to the user with one of the following return codes in register 15:

Code	Meaning
00	An SCB is successfully created, overlaid or cancelled.
08	The user is attempting to cancel or overlay a nonexistent SCB.

Format of SCB

```
 0(0)  ┌──────────────────────────┐
       │0 or pointer to next SCB│
 4(4)  ├──────────────────────────┤
       │exit address              │
 8(8)  ├──────────────────────────┤
       │parameter list address    │
12(C)  └──────────────────────────┘
```

DETACH-SVC 62: DETACH is a NOP located in DMSSVT.

CHKPT-SVC 63: CHKPT is a NOP located in DMSSVT.

RDJFCB-SVC 64: This routine (RDJFCB) receives control when a RDJFCB macro instruction is issued. When it gets control, RDJFCB obtains the address of the JFCB from the DCBEXLST field in the DCB and sets the JFCB to zero. It then reads the simulated JFCB located in CMSCB that was produced by issuing a FILEDEF into the closed area. RDJFCB calls the STATE function program to determine if the associated file exists. If it does, RDJFCB returns to the calling program. If the file does not exist, RDJFCB sets a switch in the DCB to indicate this and then returns to the calling program. RDJFCB is located in DMSSVT. All the options of RDJFCB are supported.

Note: The switch set by the RDJFCB is tested by the FORTRAN object-time direct-access handler (DIOCS) to determine whether or not a referenced disk file exists. If it does not, DIOCS initializes the direct access file.

SYNAD-SVC 68: Located in DMSSVT, SYNAD attempts to simulate the functions SYNADAF and SYNADRLS. SYNADAF expansion includes an SVC 68 and a high-order byte in register 15 denoting an access method. SYNAD prepares an error message line, swap save areas and register 13 pointers. The message buffer is 120 bytes: bytes 1-50, 84-119 blank; bytes 51-120, 120S INPUT/OUTPUT ERROR nnn ON FILE: "dsname"; where nnn is the CMS RDBUF/WRBUF error code. All the options of SYNAD are supported.

SYNADRLS expansion includes SVC 68 and a high order byte of X'FF' in register 15. The save area is returned, and the message buffer is returned to free storage.

BACKSPACE-SVC 69: Also in DMSSVT. For a tape, a BSR command is issued to the tape. For a direct access data set, the CMS write and read pointers are decremented by one. Control is passed to BACKSPACE in DMSSVT when a BACKSPACE macro is issued. BACKSPACE decrements the read write pointer by one and returns control to the user. No physical tape or disk adjustments are made until the next READ or WRITE macro is issued. All the options of BACKSPACE are supported.

TGET/TPUT-SVC 93: Located in DMSSVN, this routine receives control when a TGET or TPUT macro is issued. It is provided to support TSO service routines needed by program products. TGET reads a terminal line; TPUT writes a terminal line. The return code is zero if the operation was successful and a four if an error was encountered.

TCLEARQ-SVC 94: TCLEARQ is located in DMSSVN and causes the terminal input queue to be cleared via a call to DESBUF. At completion a return is made to the user.

STAX-SVC 96: Located in DMSSVT, STAX gets and chains a CMSTAXE control block for each STAX SVC issued with an exit routine address specified. The chain is anchored by TAXEADDR in DMSNUC. If no exit address is specified the most recently added CMSTAXE is cleared from the chain. If an error occurs during STAX SVC processing, a return code of eight is placed in register 15. The only option of STAX which may be specified is EXIT ADDRESS.

GET/PUT: See the DMSSQS prolog for description.

READ/WRITE: OS READ and WRITE macros branch and link to DMSSBS. DMSSBS branches and links to DMSSEB and, if the disks is an OS disk, DMSSEB branches and link to DMSROS. See DMSSBS for description.

NOTE/POINT/FIND(type C): OS NOTE, POINT, and FIND (type c) macros branch and link to entry points in DMSSCT. If the disk is an OS disk, DMSSCT branches and links to DMSROS. See DMSSCT for descriptions.

CHECK: See the DMSSCT prolog for description.

Notes on using the OS simulation routines:

- CMS files are physically blocked in 800-byte blocks, and logically blocked according to a logical record length. If the filemode of the file is not 4, the logical record length is equal to the DCBLRECL and the file must always be referenced with the same DCBLRECL, whether or not the file is blocked. If the filemode of the file is 4, the logical record length is equal to the DCBBLKSI and the file must always be referenced with the same DCBBLKSI.

- When writing CMS files with a filemode number other than four, the OS simulation routines deblock the output and write it on a disk in unblocked records. The simulation routines delete each 4-byte block descriptor word (BDW) and each 4-byte record descriptor word (RDW) of variable length records. This makes the OS-created files compatible with CMS-created files and CMS utilities. When CMS reads a CMS file with a filemode number other than four, CMS blocks the record input as specifies and restores the BDW and RDW control words of variable length records.

  If the CMS filemode number is four, CMS does not unblock or delete BDWs or RDWs on output. CMS assumes on input that the file is blocked as specified and that variable length records contain block descriptor words and record descriptor words.

- To set the READ/WRITE pointers for a file at the end of the file, a FILEDEF command must be issued for the file specifying the MCD option.

- A file is erased and a new one created if the file is opened and all the following conditions exist:

  -- The OUTPUT or OUTIN option of OPEN is specified.

  -- The TYPE option of OPEN is not J.

  -- The dataset organization option of the DCB is not direct access or partitioned.

  -- A FILEDEF command has not been issued for data set specifying the MOD option.

- The results are unpredictable if two DCBs read and write to the same data set at the same time.

Command Flow of Commands Involving OS Access

ACCESS COMMAND FLOW: The module DMSACC gets control first when you invoke the ACCESS command. DMSACC verifies parameter list validity and sets the necessary internal flags for later use. If the disk you access specifies a target mode of another disk currently accessed, DMSACC calls DMSALU to clear all pertinent information in the old active disk table. DMSACC then calls DMSACF to bring in the user file directory of the disk. As soon as DMSACF gets control, DMSACF calls DMSACM to read in the master file directory of the disk. Once DMSACM reads the label cf the disk, and determines that it is an OS disk, DMSACM calls DMSROS (ROSACC) to complete the access of the OS disk. Upon returning from DMSROS, DMSACM returns immediately to DMSACF, bypassing the master file directory logic for CMS disks. DMSACF then checks to determine if the accessed disk is an OS disk. If it is an OS disk, DMSACF returns immediately to DMSACC, bypassing all the user file directory logic for OS disks. DMSACC checks to determine if the accessed disk is an CS disk; if it is, another check determines if the accessed disk replaces another disk to issue an information message to that effect. Another check determines if you specified any options or fileid and, if you did, a warning message appears on the terminal. Control now returns to the calling routine.

FILEDEF COMMAND FLOW: DMSFLD gets control first when you issue a CMS FILEDEF command. DMSFLD adds, changes, or deletes a FILEDEF control block (CMSCB) and returns control to the calling routine.

LISTDS COMMAND FLOW: The module DMSLDS gets control first when you invoke the LISTDS command. DMSLDS verifies parameter list validity and calls module DMSLAD to get the active disk table associated with the specified mode. DMSLDS reads all format 1 DSCB and if you specified the PDS option and the data set is partitioned, DMSLDS calls DMSROS (ROSFIND) to get the members of the data set. After displaying the DSCB (or DSCB) on you console, DMSLDS returns to the calling routine.

MOVEFILE COMMAND FLOW: The module DMSMVE gets control first when you issue a CMS MOVEFILE command. DMSMVE calls DMSFLD to get an input and output CMSCB and, if the input DMSCB is for a disk file, DMSMVE calls DMSSTT to verify the existence of the input file and get default DCB parameters in absence of CMSCB DCB parameters. DMSMVE uses OS OPEN, FIND, GET, PUT, and CLOSE macros to move data from the input file to the output file. After moving the specified data, control returns to the calling routine.

QUERY COMMAND FLOW: The module DMSQRY gets control first when you invoke the QUERY command. DMSQRY verifies parameter list validity and calls DMSLAD to get the active disk table associated with the specified mode. DMSQRY displays all the information that you requested on your console. When DMSQRY finishes, control returns to the calling routine.

RELEASE COMMAND FLOW: The module DMSARE gets control first when you invoke the RELEASE command. DMSARE verifies parameter list validity and checks to determine if the disk you want to release is accessed. If the disk you want to release is currently active, DMSARE calls DMSALU to clear all pertinent information associated with the active disk. DMSALU first checks the active disk table for any existing CMS tables kept in free storage. If the disk you want to release is an OS disk, DMSALU does not find any tables associated with a CMS disk. If the disk is an OS disk, DMSALU releases the OS FST blocks (if any) and clears any OS FST pointers in the OS file control blocks. DMSALU then clears the active disk table and returns to DMSARE. DMSARE then clears the device table address for the specified disk and returns to the calling routine.

STATE COMMAND FLOW: The module DMSSTT gets control first when you invoke the STATE command. DMSSTT verifies the parameter list validity and calls module DMSLAD to get the active disk table associated with the specified mode. Upon return from DMSLAD, DMSSTT calls DMSLFS to find the file status table (FST) associated with the file you specified. Once DMSLFS finds the associated FST, it checks to determine if the file resides on an OS disk. If it does, DMSLFS calls DMSROS (ROSSTT) to read the extents of the data set. Upon return from DMSROS, DMSLFS returns to DMSSTT. DMSSTT then copies the FST (or OS FST) to the FST copy in statefst and returns to the calling routine.

OS Access Method Modules--Logic Description

DMSACC MODULE: Once DMSACC determines that the disk you want to access is an OS disk, it bypasses the routines that perform LOGIN UFD and LOGIN ERASE.

If the disk you want to access replaces an OS disk, message DMSACC724I appears at your terminal.

If you specified any options or fileid in the ACCESS command to an OS disk, a warning message, DMSACC230W, appears to notify you that such options or fileid were ignored. DMSACC returns to the calling routine with a warning code of 4.

DMSACF MODULE: DMSACF verifies that the disk you want to access is an OS disk and, if it is, exits immediately.

DMSACM MODULE: DMSACM saves the disk label and VTOC address in the ADT block if the disk is an OS disk. DMSACM checks to determine if a previous access to an OS disk loaded DMSROS. If not, DMSACM calls DMSSTT to verify that DMSROS text exists. Upon successful return from STATE, DMSACM loads DMSROS text into the high storage area with the same protect key and calls the OS access routine (ROSACC) of DMSROS to read the format 4 DSCB of the disk. Upon successful return from DMSROS, control returns to the calling routine. Any other errors are treated as general logon errors.

DMSALU MODULE: If the disk is an OS disk, DMSFRET returns the OS FST blocks (if any) to free storage. DMSALU clears the OS FST pointer in all active OS file control blocks, decrements the DMSROS usage count and, if the usage count is zero, clears the address of DMSROS in the nucleus area. DMSALU also calls DMSFRET to returns to free storage the area which DMSROS occupies.

DMSARE MODULE: DMSARE ensures that the disk you want to relase is an OS disk. DMSARE calls DMSALU to release all OS FST blocks and, if necessary, to free the area DMSROS occupies. Upon return from DMSALU, DMSARE clears the common CMS and OS active disk table.

DMSFLD MODULE

- DSN -- If you specify the parameter DSN as a question mark (?), FILEDEF displays the message DMSFLD220R to request you to type in an OS data set name with the format Q1.Q2.QN. Q1, Q2, and QN are the qualifiers of an OS data set name. If you specify the parameter DSN as Q1.Q2.QN, FILEDEF assumes that Q1, Q2, and QN are the qualifiers of an OS data set name, and stores the qualifiers with the format Q1.Q2.QN in a free storage block and chains the block to the FCB.

- CONCAT -- If you specify the CONCAT option, FILEDEF assumes that the specified FILEDEF is unique unless a filedef is outstanding with a matching ddname, filename, and filetype. This allows you to specify more than one FILEDEF for a particular ddname. The CONCAT option also sets the FCBCATML bit in the FCB to allow the OS simulation routine to know the FCB is for a concatenated MACLIB.

- MEMBER -- If you specify the member option, filedef stores the member name in FCBMEMBR in the FCB to indicate that the OS simulation routine should set the read/write pointer to point to the specified BPAM file member when OPEN occurs.

DMSLDS MODULE: DMSLDS saves the return register, sets itself with the nucleus protection key, clears the dsname key, and initializes its internal flag.

DMSLDS verifies parameter list validity. The data set name must not exceed 44 characters, and the disk mode (the last parameter before the options) must be valid. DMSLDS joins the quailifiers with dots (.) to form valid data set names. If you specify the data set name as a question mark (?), DMSLDS prompts you to enter the dsname in exactly the same form as the dsname which appears on the disk.

DMSLDS calls DMSLAD to find the active disk table block. If you specify filemode as an asterisk (*), DMSLAD searches for all ADT blocks. If you specify the filemode as alphabetic, DMSLAD finds only the ADT block for the specified filemode.

If you specify the dsname (which is optional), DMSLDS sets the channel programs to read by key. If you did not specify a dsname, DMSLDS searches the whole VTOC for format 1 DSCBS and displays all the requested information contained in the DSCB on your console. If you specify the format option, the RECFM, LRECL, BLKSI, DSCRG, DATE, LABEL, FMODE, and data set name appear on you console; otherwise, only the FMODE and data set name appear.

If you specify the PDS option, DMSLDS calls the 'find' routine (rosfind) in DMSROS to read the member directory and pass back, one at a time, in the fcbmembr field of CMSCB the name of each member of the data set. This occurs if the data set is partitioned.

After processing finishes, DMSLDS resets the nucleus key to the same value as the user key, puts the return code in register 15, and returns to the calling routine.

DMSLFS MODULE: DMSLFS verifies that the FST being searched for has an CS disk associated with it. DMSLFS calls the DMSROS state routine (ROSSTT) to verify that the data set exists and CMS supports the data set attributes. Upon return from DMSROS, a return code of 88 indicates that the data set was not found, and DMSLDS starts the search again using the next disk in sequence. Any other errors, such as a return code 80, cause DMSLFS to exit immediately. A return code of 0 from DMSRCS indicates that the data set is on the specified disk. From this point on, execution occurs common to both CMS and OS disks.

DMSMVE MODULE: If you specify the PDS option and the input is from a disk, DMSMVE sets the FCBMVPDS bit and issues an OS FIND macro before opening an output DCB to position the input file at the next member. DMSMVE then stores the input member name in the output CMSCB for use as the output filename. After reaching end-of-file on a member, the message DMSMVE225I appears, DMSMVE closes the output DCB, and passes control to find the next member. After moving all the members to separate CMS files, movefile displays message DMSMVE226I, closes the input and output DCBS, and returns control to the calling routine.

DMSROS MODULE:

- ROSACC Routine -- ROSACC gets control from DMSACM after DMSACM determines that the label of the disk belongs to an OS disk. The ROSACC routine reads the format 4 DSCB of the disk to further verify the validity of the OS disk. ROSACC updates the ADT to contain the address of the high extent of the VTOC (if the disk is a DOS disk) or the address of the last active format 1 DSCB (if the disk is an CS disk), and the number of cylinders in the disk. If the disk is a DCS disk, ROSACC sets a flag in the ADT. Information messages appear to notify you that the disk was accessed in read-only mode. If the disk is already accessed as another disk, another information message appears to that effect. Finally ROSACC zeroes out the ADTFLG1 flag in the ADT, sets the ADRFLG2 flag to reflect that an OS disk was accessed, and returns control to the calling routine.

- ROSSTT Routine -- Verifies the existence of an CS data set and verifies the support of the data set attributes.

  Note: Within the ROSSTT description, any reference to FCB or CMSCB implies a DOSCB if DOS is active.

ROSSTT gets control from DMSSTT after DMSSTT determines that the STATE operation is to an OS disk. The ROSSTT routine searches for the correct FCB which a previous FILEDEF associated with the data set. If the DOS environment is active, ROSSTT locates the correct DOSCB that defines a data set described by a previous DLBL. If ROSSTT finds an active FST, control passes to ROSSTRET; otherwise, ROSSTT acquires the dsname block, places its address in the FCB, and moves the dsname in the FCB to the acquired block. ROSSTT acquires an FST block, chains it to the FST chain, and fills all general fields (dsname, disk address, and disk mode). ROSSTT now reads the format 1 DSCB for the data set and checks for unsupported options (BDAM, ISAM, VSAM, and read protect).

Errors pass control back to the calling routine with an error code. ROSSTT groups together all the extents of the data set (by reading the format 3 DSCB if necessary) and checks them for validity. ROSSTT bypasses any user labels that may exist and displays a message to that effect. Next, ROSSTT moves the DSCB1 BLKSIZE, LRECL, and RECFM parameters to the OS FST and passes control to rosstret.

- ROSSTRET Routine -- If the disk is not a DOS disk, rosstret passes control back to the caller. If the specified disk is a DOS disk, rosstret fills in the OS FST BLKSIZE, LRECL, and RECFM fields that were not specified in the DSCB1. If the CMSCB fields are zero, rosstret defaults them to BLKSIZE=32760, LRECL=32670, and RECFM=U. Control then returns to the calling routine.

- ROSRPS Routine -- ROSRPS reads the next record of an OS data set. Upon entry to the ROSRPS entry point, ROSRPS calls CHKXTNT and, if the current CCHHR is zero, SETXTNT to ensure the CCHHR and extent boundaries are correctly set. ROSRPS then calls DISKIO and, if necessary, CHKSENSE and GETALT to read the next record. If no errors exist or an unrecoverable error occurred, control returns to the user with either a zero (I/O OK) or an 80 (I/O error) in register 15. If an unrecoverable error occurs, ROSRPS updates the CCWS and buffer pointers as necessary and recalls CHKXTNT and DISKIO to read the next record.

- ROSFIND Routine -- ROSFIND sets the CCHHR to point to a member specified in FCBMEMBR or, if the FCBMVPDS bit is on, sets the CCHHR to point to the next member higher than FCBMEMBR and sets a new member name in FCBMEMBR.

  Upon entry at the ROSFND entry point, ROSFND sets up a CCW to search for a higher member name if the FCBMVPDS bit is on, or an equal member name if the FCBMVPDS bit is off. It then calls SETXTNT, DISKIO and, if needed, CHKSENSE and GETALT to read in the directory block that contains the member name requested. After reading the block, it is searched for the requested member name. If the member name is not found, an error code 4 returns to the calling routine. If an I/O error occurs while trying to read the PDS block, an error code 8 returns to the calling routine. If the member name is found, TTRCNVRT is called to convert the relative track address to a CCHH and pass the address of the member entry to the calling routine.

- ROSNTPTB Routine -- ROSNTPTB gets the current TTR, sets the current CCHHR to the value of the TTR, and backspaces to the previous record.

  Upon entry at the ROSNTPTB entry point, ROSNTPTB checks to determine if a NOTE, POINT, or BSP operation was requested.

  If register 0 is zero, NOTE is assumed. The note routine calls CHRCNVRT to convert the CCHH to a relative track and returns control to the calling routine with the TTR in register 0.

If register 0 is positive upon entry into DMSROS, POINT is assumed and ROSNTPTB loads a TTR from the address in register 0 and calls TTRCNVRT and SETXTNT to convert the TTR to a CCHHR. Then control returns to the calling routine.

If register 0 is negative upon entry into DMSROS, BSP (BACKSPACE) is assumed. The backspace code checks to determine if the current position is the beginning of a track. If not, the backspace code decrements the record number by one and control then returns to the calling routine. If the current position is the beginning of a track, the backspace code calls CHRCNVRT to get the current CCHH. The backspace code then calls rdcnt to get the current record number of the last record on the new track, calls setxtnt to set the new extent boundaries, and returns control to the calling routine.

__DMSSCT MODULE__:

- NOTE Routine -- Upon entry to note, DMSSCT checks to determine if the DCB refers to an OS disk. If it does, DMSSCT calls DMSROS (ROSNTPTB) to get the current TTR. Control then returns to the user.
- POINT Routine -- Upon entry to point, DMSSCT checks to determine if the DCB refers to an OS disk. If it does, DMSSCT calls DMSROS (ROSNTPTB) to reset the current TTR, calls CKCONCAT and returns control to the calling routine.
- CKCONCAT Routine -- Upon entry to CKCONCAT, DMSSCT checks to determine if the FCB MACLIB CONCAT bit is on. If it is on, DCBRELAD+3 sets the correct OS FST pointer in the FCB and returns control to the calling routine. If the FCB MACLIB CONCAT bit is off, control returns to the calling routine.
- FIND (type_C) Routine -- If the DCB refers to an OS disk, DMSSCT calls DMSROS (ROSNTPTB) to update the TTR and control returns to the calling routine.

__DMSSEB MODULE__:

- EOBROUTN Routine -- If the FCB OS bit is on, control passes to OSREAD. Otherwise, if no special I/O routine is specified in FCBPROC, control passes to EOB2 in DMSSEB.
- OSREAD Routine -- DMSSEB calls DMSROS to perform a read or write and then control passes to EOBRETRN which, in turn, passes control back to DMSSBS. DMSSBS passes control back to the routine calling the read or write macro operation.

__DMSSOP MODULE__ -- If the MACLIB CONCAT option is on in the CMSCB, OPEN checks the MACLIB names in the global list and fills in the addresses of OS FSTS for any MACLIBS on OS disks. The CMSCB of the first MACLIB in the global list merges and initializes CMSCBS.

If the CMSCB refers to a data set on an OS disk, DMSSOP checks to ensure that the data set is accessible and the DCB does not specify output, BDAM, or a key length. If any errors occur, error message DMSSOP036E appears and DMSSOP does not open the DCB. DMSSOP fills them in from the OS FST for the data set.

If the CMSCB fcbmembr field contains a member name (filled in by FILEDEF with the member option), DMSSOP issues an OS FIND macro to position the file pointer to the correct member. If an error occurs on the call to the FIND macro, error message DMSSOP036E appears and DMSSOP does not open the DCB.

DMSSVT MODULE:

- BSP (backspace) Routine -- Upon entry, backspace checks for the FCB OS bit. If it is on, the BSP routine calls DMSROS (ROSNTPTB) to backspace the TTR and control returns to the calling routine.

- FIND (type_D) Routine -- Upon entry to find, the find routine checks the FCB OS bit. If it is on, the FIND routine takes the OS FST address from the CMSCB or, if the CONCAT bit is on, from the global MACLIB list. The FIND routine then calls DMSROS (ROSFIND) to find the member name and TTR. DMSROS searches for a matching member name or, if the FCBMVPDS option is specified, a higher member name. If the DMSROS return code is 0 or 8, or if the FCBCATML bit is not on, control returns to the calling routine with the return code from DMSROS. If the return code is 4 and the FCBCATML bit is on, DMSSVT checks to determine if all the global MACLIBS were searched. If they were, control returns to the calling routine with the DMSROS return code. If they were not, DMSSVT issues the FIND on the next MACLIB in the global list.

- BLDL Routine--BLDL list = FF LL NAME TTR KZC DATA

  If the DCB refers to an OS disk, the BLDL routine fills in the TTR, C-byte and data field from the OS data set.

DMSQRY MODULE:

- SEARCH Routine -- The search routine ensures that any OS disk currently active is included in the search order of all disks currently accessible.

- DISK Routine -- The disk routine displays the status of any or all OS disks using the following form:

  'MODE(CUU): (NO. CYLS.), TYPE R/O - OS.'

DMSSTT MODULE -- DMSSTT verifies that the disk being searched is an OS disk. DMSSTT calls DMSLFS to get the FST associated with the data set. Upon return from DMSLFS, DMSSTT checks the return code to ensure that CMS supports the data set attributes. A return code of 81 or 82 indicates that CMS does not support the data set and message DMSSTT229E occurs to that effect. DMSSTT then clears the FST copy with binary zeros, and moves the filename, filetype, filemode, BLKSIZE, LRECL, RECFM, and flag byte to the FST copy. From this point on, common code execution occurs for both CMS and OS disks.

Routines Common to All of DMSROS

- CHRCNVRT Routine -- The CHRNCVRT routine converts a CCHH address to a relative track address.

- CHKSENSE Routine -- CHKSENSE checks sense bits to determine the recoverability of a unit check error if one occurs.

- CHKXTNT Routine -- CHKXTNT checks to determine if the end of split cylinder or the end of extent occurred, and, if so, updates to the next split cylinder or extent.

- DISKIO Routine -- DISKIO starts I/O operation on a CCW string via a DIAGNOSE X'20'.

- GETALT Routine -- GETALT switches reading from alternate track to prime track, and from prime track to alternate track.

- RDCNT Routine -- RDCNT reads count fields on the track to determine the last record number on the track.

- SETXTNT Routine -- SETXTNT sets OSFSTEND to the value of the end of the extent and, if a new extent is specified, sets CCHHR to the value of the start of the extent.

Simulating a DOS Environment under CMS

CMS/DOS is a functional enhancement to CMS that provides DOS installations with the interactive capabilities of a VM/370 virtual machine. CMS/DOS operates as the background DOS partition; the other four partitions are unnecessary, since the CMS/DOS virtual machine is a one-user machine.

CMS/DOS provides read access to real DOS data sets, but not write or update access. Real DOS private and system relocatable, source statement, and core-image libraries can be read. This read capability is supported to the extent required to support the CMS/DOS linkage editor, the DOS/PLI and DOS/VS COBOL compilers, the FETCH routine, and the RSERV, SSERV, and ESERV commands. No read or write capability exists for the DOS procedure library, except for copying procedures from the procedure library (via the PSERV command) or displaying the procedure library (via the DSERV command).

CMS/DOS does not support the standard label cylinder.

INITIALIZING DOS AND PROCESSING DOS SYSTEM CONTROL COMMANDS

Initialization of the CMS/DOS operating environment requires the setting of flags and the creation of certain data areas in storage. Once initialized, these flags and data areas may then be changed by routines invoked by the system control commands.

Five modules are described in this section:

- DMSSET Activates the CMS/DOS environment control blocks to be used during CMS/DOS processing.

- DMSOPT Sets or resets compiler execution-time options.

- DMSASN Relates logical units to physical units.

- DMSLLU Lists the assignments of CMS/DOS physical units.

- DMSDLB Associates a DTF with a logical unit for CMS/DOS processing.

DMSSET--Initializing the CMS/DOS Operating Environment

DMSSET initializes the CMS/DOS operating environment as follows:

- Verifies that the mode, if specified, is for a DOS formatted disk.
- Stores appropriate data in the SYSRES LUB and PUB.
- Locates and loads the CMS/DOS discontiguous shared segment. Saves (in NUCON) the addresses of the two major CMS/DOS data blocks, SYSCOM, BGCOM,and the address of the CMS/DOS discontiguous shared segment (CMSDOS).
- Sets the DOSMODE and DOSSVC bits in DOSFLAGS in NUCON.
- Assigns (via ASSGN) the SYSLOG logical unit as the CMS virtual console.

The CMS/DOS operating environment is entered when the CMS SET DOS ON command is issued, invoking the module DMSSET.

Data Areas Prepared for Processing during CMS/DOS Initialization

Several data areas are prepared for processing during initialization. The main CMS data area, NUCON, is modified to contain the addresses of two DOS data areas, SYSCOM and BGCOM.

The SYSCOM DSECT is the DOS system communications region. It consists mainly of address constants, including the addresses of the AB option table, the PUB ownership table, and the FETCH table. It also includes such information as the number of partitions (always one for CMS/DOS) and the length of the PUB table.

The BGCOM DSECT is the partition communication region. It includes such information as the date, the location of the end of supervisor storage, the end address of the last phase loaded, the end address of the longest phase loaded, bytes used to set the language translator and supervisor options, and the addresses of many other DOS data areas such as the LUB, PUB, NICL, FICL, PIB, PIB2TAB, and the PCTAB.

The LUB and PUB tables are also made available during initialization. The LUB is the logical unit block table. It acts as an interface between the user's program and the CMS/DOS physical units. It contains an entry for each symbolic device available in the system.

Each of the symbolic names in the LUB is mapped into an element in the PUB, the physical unit block table. The PUB table contains an entry for each channel and device address for all devices physically available to the system and also contains such information as device type code, CMS disk mode, tape mode setting, and 7-track indicator.

Two bits are set in DOSFLAGS in NUCON, DOSMODE and DOSSVC. DOSMODE specifies that this virtual machine is running in the CMS/DOS operating environment. DOSSVC indicates whether OS or DOS SVCs are operative in the operating environment. If DOSSVC is set, DOS SVCs are used; otherwise, OS SVCs are operative.

SETTING OR RESETTING SYSTEM ENVIRONMENT OPTIONS

Once the CMS/DOS environment is initialized, the flags and control blocks set during initialization can be modified and manipulated to perform the functions specified by commands entered at the console. This section describes the modules that set and reset the system environment options. That is, they set those options that control compiler execution and that control the configuration of logical and physical units in the system.

DMSOPT--Setting and Resetting Compiler Options

The CMS/DOS OPTION command invokes module DMSOPT, which sets either the default options for the compiler or the options specified on the command line. The nonstandard language translator options switch and the job duration indicator byte are altered. Options are set using two control words located in the partition communication region (BGCOM). Bits in bytes JCSW3 or JCSW4 are set, depending on the options specified.

DMSASN--Associate System or Programmer Logical Units with Physical Units

Module DMSASN is invoked when the ASSGN command is entered. DMSASN first scans the command line to ensure that the logical unit being assigned is valid for the physical unit specified (for example, SYSLOG must be assigned to either the virtual console or the virtual printer). Once the command line is checked, PUB and LUB entries are modified to reflect the specified assignment.

For the PUB entry, the device type is determined (via DIAG 24) and the device type code is placed in the PUB. Other modifications are made to the PUB depending on the specified assignment. The LUB entry is then mapped to its corresponding PUB.

DMSLLU--List the Assignments of CMS/DOS Logical Units

The function of DMSLLU is to request a list of the physical units assigned to logical units. It performs this function by referencing information located in the CMS/DOS data blocks, specifically SYSCOM, LUB, and PUB. Another data block, the next in class (NICL) table is also referenced.

The information on the command line is scanned and the appropriate items are displayed at the user's console. If an option (EXEC or APPEND) is specified, an EXEC file is created ($LISTIO EXEC A1) to contain the output. If EXEC is specified, any existing $LISTIO EXEC A1 file is erased and a new one is created. If APPEND is specified, the new file is appended to the existing file.

DMSDLB--Associate a DTF Table Filename with a Logical Unit

DMSDLB is invoked when the CMS/DOS DLBL command is entered. DMSDLB associates a DTF (Define The File) table filename with a logical unit. This function is performed by creating a control block called a DOSCB, which contains information defining a DOS file used during job execution. DLBL is valid only for sequential or VSAM disk devices.

This information parallels the label information written on a real DOS SYSRES unit under DOS/VS. The DOSCB contains such information as the name, type, and mode of the referenced dataset, its device type code, its logical unit specification, and its dataset type (SAM or VSAM).

A DOSCB is created for each file specified by the user during a terminal session. The DOSCBs are chained to each other and are anchored in NUCON at the field DOSFIRST. The chain remains intact for the entire session, unless an abend occurs or the user specifically clears an entry in the the DOSCB chain. A given DOSCB is accessed when an OPEN macro is issued from an executing user program.

The overall logic flow for DMSDLB is as follows:

1. Scans the command line to ensure that any options entered are valid (that is, anything to the right of the open parenthesis).
2. Processes the first operand (ddname or *). When ddname is specified, loop through the DOSCB chain to find a matching ddname. If none is found, DMSDLB calls DMSFRE to get storage to create a new DOSCB for this file. The old copy of the DOSCB is then saved so that, in case of errors during processing, it can be retrieved intact. The new copy of the DOSCB contains updates and DOSCB replaces the old copy if there are no errors.
3. The mode specification is checked to ensure that it is a valid mode letter; if the file is a CMS file, the mode letter must specify a CMS disk. If DSN has been specified, the mode letter must be for a non-CMS disk.
4. Process each option on the command line appropriately.
5. If EXTENT or MULT is specified, a separate block of free storage is obtained to contain information about the extent, for example, a block is obtained to contain the DOS data set name.
5. Check for errors. If there are errors, any blocks created during processing are purged and an error message is issued. If there are no errors, restore the old block, which has been modified to reflect current processing, and return control to DMSITS.

PROCESS CMS/DOS OPEN AND CLOSE FUNCTIONS

The CMS/DOS OPEN routines are invoked in response to DOS OPEN macros. They operate on DTF (define the file) tables and ACB (access method control block) tables created when the DTFxx and ACB macros are issued from an executing user program. These tables contain information such as the LOG unit specification for the file, the DTF type of the file, the device code for the file, and so forth. The information in the tables varies depending upon the type of DTF specified (that is, the table generated by a unit record DTF macro is slightly different from the table generated by a DTF disk macro).

Five routines are invoked to perform OPEN functions, DMSOPL, DMSOR1, DMSOR2, DMSOR3, and DMSBOP. DMSCLS performs the CLOSE function.

Opening Files Associated With DTF Tables

Depending on the type of OPEN macro issued from a user program, one of five CMS/DOS OPEN routines could be invoked. OPENR macros give control to DMSOR1 and, depending on the DTF type specified, DMSOR2 or DMSOR3 may be invoked. These three routines (DMSOR1,DMSOR2, and DMSOR3) request the relocation of a specified file. DMSOPL is invoked by the DOS/VS compilers when they need access to a source statement library. These routines are mainly interface routines to DMSBOP, which performs the main function of opening the specified file. Each of the routines calls DMSBOP.

DMSBOP is the CMS/DOS routine that simulates the DOS/VS OPEN function. The basic function of DMSBOP is the initialization of DTF tables (that is, setting fields in specified DTFs for use by the DOS/VS LIOCS routines).

When a DOS problem program is compiling, a list of DTFs and ACBs is built. At execution time, this list is passed to DMSBOP. The logic flow of DMSBOP is as follows:

1. Scans the list of DTF and ACB addresses, handling each iteam in the list in line. When the OPEN macro expands, register 1 points to the name of the $$B transient to receive control ($$BOPEN) and register 0 points to the list of DTF/ACB addresses to be opened.

2. When an ACB is encountered in the table, control is passed directly to the VSAM OPEN routine, $$BOVSAM. The VSAM routine is responsible for opening the file and returning control to DMSBOP.

3. When a DTF is encountered in the table, DMSBOP itself handles the OPEN:

   a. For reader/punch files (DTFCD), the OPEN bit in the DTF table is turned on.

   b. For printer files (DTFPR), if two IOAREAs are specified, the IOREG is loaded with the address of the appropriate IOAREA. Next, the PUB index byte associated with the logical unit specified in the DTF is checked to ensure that a physical device has been assigned and the PUB device code is then analyzed. The OPEN bit in the DTF table is then turned on.

   c. For console files (DTFCN), no OPEN logic is required.

   d. For tape files (DTFMT), the PUB device type code must specify TAPE. If an IOREG is specified (for output tapes only), the address of the appropriate IOAREA is placed in it. For input files, there is separate processing for tapes with standard label, nonstandard label, and no label. For output tapes, both tape data files and work tape files are treated as no label tapes.

   e. For disk files (DTFxx), the LUB is verified to ensure that the logical unit has been assigned. A check is made to ensure that the DOSCB exists for the DTF filename. For disk output files, the address of the appropriate IOAREA is placed in IOREG. For disk input files, the existence of the file is verified via a call to DMSSST. Also, EXTENT information is initialized and the OPEN bit is posted.

   f. DTFDT and DTFCP are separate DTF types that could describe any of the above devices.

4. After all files in the table have been opened, DMSBOP returns control to the problem program via SVC 11.

5. If errors are encountered during DMSBOP processing, an error message is issued and return is made via SVC 6.

<u>Closing Files Associated With DTFs</u>

The CMS/DOS routine that processes CLOSE requests is DMSCLS, whose logic is analogous to that of DMSBOP, the OPEN routine described above: when CLOSE expands, register 1 points to $BCLOSE and register 0 points to the list of DTF/ACB addresses. The same table containing DTFs and ACBs used to open files is also used to close those files. Each entry in the table is processed as it occurs, with control passing to a VSAM CLOSE routine ($$BCVSAM) when an ACB is encountered. The OPEN bit is then turned off.

PROCESS CMS/DOS EXECUTION-RELATED CONTROL COMMANDS

The CMS/DOS FETCH and DOSLKED commands simulate the operation of the DOS/VS fetch routines and the DOS/VS Linkage Editor. The three CMS modules that perform this simulation are:

- DMSFET--Provide an interface to interpret the DOS FETCH command line and execute the phase, if START is specified on the command line.
- DMSFCH--Bring into storage a specified phase from a system or private core-image library or from a CMS DOSLIB library.
- DMSDLK--Link edit the relocatable output of the CMS/DOS language translators to create executable programs.

<u>DMSFET and DMSFCH--Bring a Phase into Storage for Execution</u>

The DOS/VS FETCH function is simulated by CMS modules DMSFET and DMSFCH. The main control block used during a FETCH operation is FCHSECT, which contains addressing information required for I/O operations.

The FETCH command line invokes module DMSFET. This module first validates the command line and issues a FILEDEF for the DOSLIB file. It then issues a FILEDEF for a DOSLIB file. DMSFET then issues a DOS SVC 4, which invokes the module DMSFCH to perform the actual FETCH operation.

DMSFCH first determines where the phase to be fetched resides. The search order is private core-image library, DOSLIB, system core-image library. If the phase is not found in any of these libraries, DMSFCH assumes that the FETCH is for a phase in a system or private core-image library. To find a DOSLIB library member, OS OPEN and FIND macros are issued (SVC 19 and 18).

When the member is found, OS READ and CHECK macros are issued to read the first record of the file (the member directory). This record contains the number of text blocks and the length of the member.

All addressing information is stored in FCHSECT and the text blocks that the phase are read into storage. If the read is from a CMS disk, issue the OS READ and CHECK macros to read the data. If the read is from a DOS disk, first determine whether this is the first read for the DOS discontiguous shared segment (DCSS). If this is the case, CCW information is relocated to ensure that the DCSS code is reentrant. For all reads for a DOS disk, a CP READ DIAG instruction is issued. When the entire file is read, it is relocated (if it is relocatable).

If a DOSLIB is open, close it using an OS SVC 20 and return control to DMSFET. DMSFET then checks to see whether START is specified and, if so, an SVC 202 is issued for the CMS START command to execute the loaded file.

When all FETCH processing is complete, control returns to the CMS command handler, DMSITS.

Simulate the Functions of the DOS/VS Linkage Editor: DMSDLK

CMS simulation of the DOS/VS Linkage Editor function directly parallels the DOS/VS implementation of that function. For detailed information on the logic of the function, see the publication *DOS/VS Linkage Editor Logic*, Order No. SY33-8556.

Note that the modules comprising the DOS/VS Linkage Editor are prefixed by the letters IJB and are separate CSECTs. ALL of these CSECTs have counterparts contained within the one CMS module, DMSDLK. They are treated as subroutines within that module, but perform the same functions as their independent DOS/VS counterparts and have been named using the same naming conventions as for the DOS/VS CSECTs. For example, the IJBESD CSECT in DOS/VS is paralleled by the CMS DMSDLK subroutine DLKESD.

A brief dscription of the logic follows. The CMS/DOS DOSLKED command invokes the module DMSDLK, which is entered at subroutine DLKINL. DLKINL performs initialization and is later overlaid by the text buffer and the linkage editor tables. DLKINL starts to read from a DOSLNK file and processes ACTION statements, if there are any.

On encountering the first non-ACTION card (or if there is no DOSLNK file), the main flow is entered. Depending on the input on the DOSLNK or the TEXT file, records from either of those files may be read or records from a relocatable library may be read. The type of card image read determines the subroutine to which control is given for further processing.

An ENTRY card indicates the end of the input to the linkage editor. At this point, a map is produced by subroutine DLKMAP. DLKRLD is then entered to finish the editing of object modules by relocating the address constants. If the phases are to be relocatable, relocation information is added to the output on the DOSLIB. Updating of the DOSLIB library is performed by DLKCAT using the OS STOW macro.

A significant deviation from DOS/VS code is the use of OS macros, in some instances, rather than DOS/VS macros. To take advantage of CMS support of partitioned data sets, the OS OPEN, FIND, READ, CHECK, and CLOSE macros are issued rather then their DOS/VS counterparts.

SIMULATE DOS SVC FUNCTIONS

All SVC functions supported for CMS/DOS are handled by the CMS module DMSDOS. DMSDOS receives control from DMSITS (the CMS SVC handler) when that routine intercepts a DOS SVC code and finds that the DOSSVC flag in DOSFLAGS is set in NUCON.

DMSDOS acquires the specified SVC code from the OLDPSW field of the current SVC save area. Using this code, DMSDOS computes the address of the routine where the SVC is to be handled.

Many CMS/DOS routines (including DMSDOS) are contained in a discontiguous shared segment (DCSS). Most SVC codes are executed within DMSDOS, but some are in separate modules external to DMSDOS. If the SVC code requested is external to DMSDOS, its address is computed using a table called DCSSTAB; if the code requested is executed within DMSDOS, the table SVCTAB is used to compute the address of the code to handle the SVC.

The items below show the SVCs supported by CMS/DOS simulation routines, the name of the macro that invokes a given SVC code, the CMS module that executes the code, and a brief statement describing how the SVC function is performed.

<u>SVC 0</u>: <u>EXCP</u> -- Handled by module DMSXCP...reads from CMS or DOS/VS formatted disks. CCWs are converted to appropriate CMS I/O requests, for example, RDBUF/WRBUF, CARDRD/CARDPH. The CCB is posted (indicating I/O completion) using CMS return information. If a non-zero return code is returned, a CANCEL is performed. I/O requests to DOS disks are handled using CP DIAGNOSE instructions.

<u>SVC 1</u>: <u>FETCH</u> -- Handled by DMSFCH...loads a problem program phase into core and executes it, if execution is requested. For details on how FETCH works, see the section "Bring a Phase into Storage for Execution: DMSFET and DMSFCH."

<u>SVC 2</u>: <u>FETCH</u> -- Handled by DMSFCH...loads a $$$$B-Transient phase into core and executes it, if execution is requested. For details on how FETCH works, see the section "Bring a Phase into Storage for Execution: DMSFET and DMSFCH."

<u>SVC 4</u>: <u>FETCH</u> -- Handled by DMSFCH...loads a problem program phase into user storage and executes it, if execution is requested. For details on how FETCH works, see the section "Bring a Phase into Storage for Execution: DMSFET and DMSFCH."

<u>SVC 5</u>: <u>MVCOM</u> -- Handled by DMSDOS...provides the user with a way of altering bytes 12 through 23 of the partition communication region (BGCOM). Checks to ensure that the specified field is correct length and then moves the information to the specified field.

<u>SVC 6</u>: <u>CANCL</u> -- Handled by DMSDOS...cancels a CMS/DOS session. Processing depends on value in register 15 on entry; if above 256 the request is from a system program. If below 256, request is from a user program. Processing continues with control passing to EOJ code, described below.

SVC 7: WAIT -- Handled by DMSDOS...informs system programs to wait for a system event to take place before processing can continue. WAIT is an effective NOP for CMS/DOS.

SVC 8: Handled by DMSDOS...temporarily returns control to a problem program. The address of the problem to which control is being passed is contained in register 0. This address is stored in the SVC save area OLDPSW field and control is passed to the CMS SVC handler (DMSITS).

SVC 9: Handled by DMSDOS...returns control to system program (i.e. a user program has been given control, as in the case of SVC 8, and must return control to the system routine, a $$$$B-Transient routine, that called it).

SVC 11: Handled by DMSDOS...returns control to a problem program from a $$$$-B transient routine. Uses the SVC save area OLDPSW field to return to the calling program.

SVC 12: Handled by DMSDOS...resets flags in the linkage control byte of the Partition Communication Region (BGCOM) to zero; also, provides the user the capability to use a mask to set the value of this same byte. In both cases, the SVC routine that handles the request performs an AND operation to accomplish the function.

SVC 14: EOJ -- Handled by DMSDOS...normally terminates execution of a problem program. Clears control blocks and resets control words.

SVC 16: Handled by DMSDOS...establishes linkage with or terminates linkage to a user's program check routine. Locates the appropriate PC option table entry. If contents of register 0 is zero, terminates linkage: stores a zero into the routine address field of the PC option table. If register 0 is non-zero, the address of the PC routine and the save area address is passed to the STXIT macro. If a STXIT PC routine is already active, the complement of the new routine address is placed in the PC option table; if no STXIT PC routine is active, both the new routine address and the save area address are placed in the PC option table.

SVC 17: Handled by DMSDOS...provides supervisory support for the EXIT macro. Locates appropriate PC option table entry and restores user's registers and PSW. Stores the address of the PC routine in the PC option table and returns to the next sequential address in the interrupted program.

SVC 26: Handled by DMSDOS...validates address limits. Checks the limits passed in registers 1 and 2 and either returns control to the caller or writes an error message.

SVC 33: COMRG -- Handled by DMSDOS...provides the address of the partition communication region (BGCOM). Returns the address of BGCOM in register 1.

SVC 34: Handled by DMSDOS...supports the GETIME macro. Updates the date field in the partition communications region (BGCOM).

SVC 37: Handled by DMSDOS...establishes linkage to or terminates linkage from a user's abnormal termination routine. Locate the AB table entry. If register 0 contains zeros, terminates linkage: if the AB routine is active, stores zeros into the routine address field of the AB option table. If the AB routine is not active, stores zeros into both the routine address field and the save area field of the AB option table.

If register 0 is non-zero, establishes linkage: passes the address of the AB routine and the save area address to the STXIT AB macro. If STXIT AB is active, the complement of the AB routine address is stored in the AB option table. If STXIT AB is not active, both the address of the new AB routine and the address of the save area are placed in the option table.

SVC 40: _POST_ -- Handled by DMSDOS...signals the completion of a system event.

SVC 50: Handled by DMSDOS...issues an error message and terminates the command. Issued by a LIOCS routine when that routine is requested to perform a function it could not perform.

SVC 61: _GETVIS_ -- Handled by DMSDOS...used by VSAM to obtain scratch storage; also, obtains storage for a relocatable VSAM routine. Storage is obtained from the user free storage area and the address of the storage is returned in Register 1.

SVC 62: _FREEVIS_ -- Handled by DMSDOS...returns storage obtained by a GETVIS. Address of the area to be returned is pointed to by Register 1.

SVC 63: _USE_ -- Handled by DMSDOS...VSAM uses SVC 63 to ensure that system resources are updated serially, so that two or more attempts to modify the same data at the same time do not succeed. A table of counters (RURTBL) is kept for system resources. These counters are posted when a request is made for system resources. If a resource is already in use, a return code of eight is placed in register 0. If the resource is available, a zero is returned in Register 0.

SVC 64: _RELEASE_ -- Handled by DMSDOS...VSAM uses SVC 64 to release a system resource obtained via USE SVC. The appropriate counter in RURTBL is decremented by one each time a resource is released.

SVC 65: _CDLOAD_ -- Handled by DMSDOS...loads a relocatable VSAM phase into storage unless that phase has already been loaded.

If an anchor table is available, it is searched for the phase. If the phase is found, its load point, entry point, and length are returned in registers 0, 1, and 14, respectively, and register 15 contains zeros.

If the phase is not found in the anchor table, DMSFCH is called to search for it. If the phase is found in the discontiguous shared segment, return is made to the requestor as above.

If the phase was found, but not loaded, storage is obtained for it via the GETVIS SVC. DMSFCH is called again to load the phase into the storage just obtained. An anchor table is then built in the user area (unless one already exists) and return to the caller is then made as described above.

SVC 66: _RUNMODE_ -- Handled by DMSDOS...determines whether the problem program is running in real or virtual mode. Register 0 contains zero on return if the program is running in virtual mode.

SVC 75: _SECTVAL_ -- Handled by DMSDOS...used by VSAM I/O routines to obtain a sector number for 3330 or 3340 devices. The appropriate sector value is calculated from input supplied in registers 1 and 0. The sector number (from 0 to 127) is returned in register 0.

Certain DOS SVCs are treated as no-ops by CMS/DOS and other DOS/VS SVCs are not supported. These are listed below.

SVC 95: Handled by DMSDOS...provides supervisory support for the EXIT macro. The AB option of the EXIT macro provides an exit from the abnormal task termination routine and continues the task.

The linkage to either the PC or AB routine is reestablished, and the cancel condition is reset by clearing the abnormal end indication in the partition PIB extension. Control is returned to the instruction following the EXIT AB macro.

SVCS TREATED AS NO-OP BY CMS/DOS

SVC	Action
10:	Sets timer interval
18:	STXIT (IT)
20:	Establishes linkage to OC
22:	Seizes (interruption enable/disable)
24:	Sets timer interval
35:	Holds a track
36:	Frees a track
41:	Dequeues a resource
42:	Enqueues a resource
52:	0 seconds returned as remaining timer interval in register 0
67:	PFIX, fixes pages in real storage
68:	PFREE, frees pages in real storage
71:	SETPFA
85:	RELPAG
86:	FCEPGOUT
87:	PAGEIN

__SVCS NOT SUPPORTED BY CMS/DOS__: The following SVCs cause an error message to be generated and are treated as a CANCL (SVC 6).

SVC	Action
3:	Forces dequeue
13:	Sets switches in BGCOM
15:	Heads queue and executes channel program
19:	Returns from user's IT
21:	EXIT(OC)
23:	Loads phase header
25:	Issues HIO
27:	Special HIO
28:	Returns from user's MR
29:	Multiple WAITM support
30:	Waits for a QTAM element
31:	Posts a QTAM element
32:	Reserved for IBM use
38:	Initializes a subtask
39:	Terminates a subtask
43:	Reserved for IBM use
44:	External unit checks record
45:	Emulator interface
46:	OLTEP in supervisor state
47:	Multiple WAITF support
48:	Fetches a CRT trans
49:	Reserved by IBM
51:	Returns phase header
53:	Reserved by IBM
54:	Frees real page frames
55:	Gets real page frames
56:	Gets or frees PUB of POWER device
57:	Makes POWER dispatchable
58:	Interface between JCL and supervisor
59:	Interface between EOJ and supervisor
60:	EREP and CRT I/O areas address
69:	REALAD
70:	VIRTAD

72: GETCBUF/FREECBUF
73: SETAPP
74: Fixes pages in real storage for restart
76: Initializes for recording of RMSR I/O error
77: TRANSCSW
78: Reserved for IBM use
79: Reserved for IBM use
80: Reserved for IBM use
81: Reserved for IBM use
82: Reserved for IBM use
83: Reserved for IBM use
84: Reserved for IBM use
88 and up:
Reserved for IBM use

PROCESS CMS/DOS SERVICE COMMANDS

DMSSRV--Copies books from a system or private source statement library to a specified output device.

DMSPRV--Copies DOS procedures from a DOS system procedure library to a specified output device.

DMSRRV--Copies modules from a system or private relocatable library to a specified output device.

DMSDSV--Lists the directories of DOS private or system libraries.

DMSDSL--Deletes members (phases) of a DOSLIB library; compresses a DOSLIB library; lists the members (phases) of a DOSLIB library.

ESERV--De-edits, displays or punches, verifies, and updates edit assembler macros from the source statement library.

TERMINATE PROCESSING THE CMS/DOS ENVIRONMENT

DMSBAB--Gives control to an abnormal termination routine once linkage to such a routine has been established via the STXIT AB macro.

DMSITP--Processes program interrupts and SPIE exits.

DMSDMP--Simulates the $$BDUMP and $$BPDUMP routines; issues a CP DUMP command directing the dump to an offline printer.

Performing Miscellaneous CMS Functions

The CMS Batch Facility and error printouts are described below.

CMS BATCH FACILITY

The CMS Batch Facility is a function of CMS. It provides a way of entering individual user jobs through an active CMS machine from the virtual card reader rather than from the console. The batch facility reissues the IPL command after each job.

The CMS Batch Facility consists of two modules: DMSBTB, the bootstrap routine (a nonrelocatable CMS module file) and DMSBTP, the processor routine (a relocatable CMS text file that runs free storage).

General Operation of DMSBTB

The bootstrap module, DMSBTB, loads the processor routine DMSBTP and the user exit routines BATEXIT1 and BATEXIT2 (if they exist) into free storage.

DMSBTB first ensures that DMSINS (CMS initialization) has set the BATRUN and BATLOAD flags on in the CMS nucleus constant area indicating that either an explicit batch initial program load command has been issued or that the CMSBATCH command has been issued immediately after initial program load has taken place. If not, error message DMSBTB101E is typed and the batch console returns to a normal CMS interactive environment. STATE (DMSSTT) is then called to confirm the existence of the processor file DMSBTP TEXT. If the file does not exist, error message DMSTBT100E is typed and the batch console returns to the CMS interactive environment.

Using the "state" copy of the file status table (FST) for DMSBTP, DMSBTB computes the size of DMSBTP TEXT file by multiplying the logical record length by the number of logical records (no DS constants). A free storage request is made for the size of DMSBTP and the address of the routine is then stored at ABATPROC in the NUCON area of the CMS nucleus.

The existence of the user exit routines is determined by STATE. If they exist, their sizes are included in the request for free storage.

The free storage address is translated into graphic hexadecimal format and the CMS LOAD command is issued to load the DMSBTP TEXT file into the reserved free storage area. The user exit routines, BATEXIT1 TEXT and BATEXIT2 TEXT are also loaded at this time. If these files do not exist, an unresolved external reference error code is returned by the loader, but is ignored by DMSBTB because these routines are optional. If an error (other than unresolved names) occurs, error message DMSBTB101E is typed and the batch console returns to the CMS interactive environment.

The loader tables are searched for the address of the ABEND entry point DMSBTPAB in the loaded batch processor. When the entry is found, its address and that of entry DMSBTPLM are stored in ABATABND and the ABATLIMT respectively, in the NUCON area of the CMS nucleus. If the ABEND entry point is not found in the tables, error message DMSBTB101E is typed and the batch console returns to the CMS interactive environment.

The BATLOAD flag is set off to show that DMSBTP has been loaded, the BATNOEX flag is set on to prevent user job execution until DMSBTP encounters a /JOB card and finally, control is returned to the command processor DMSINT.

If an error message is issued, DMSERR is called to type the message, and the BATRUN and BATLOAD flags are set off before control is returned to CMS. This allows the normal CMS interaction to resume.

General Operation of DMSBTP

The batch processor module DMSBTP simulates the function of the CMS console read module DMSCRD. This is accomplished by issuing reads to the virtual card reader, formatting the card-image record to resemble a console record and returning control to CMS to process the command (or data) request. DMSBTP also performs reads to the console stack if the stack is not empty, checks for and processes the /JOB card, ensuring that it is the first record in the user job, traps all CP commands to maintain system integrity and performs job initialization, cleanup, and job recovery.

Upon receiving control, DMSBTP checks the BATCPEX flag in NUCON. If the flag is set on, control was received from DMSCPF and a branch is made to the CP trap routine to verify that the command is allowable under batch. The function of that routine is described later. If the BATCPEX flag is off, control was received from DMSCRD (console read module) and DMSBTP checks for finished reads in the real batch console stack. If the number of finished reads is not zero, control is returned to DMSCRD to process the real console finished (stacked) reads. If the number of finished reads is zero, a record is read from the batch virtual card reader into the CARD buffer via an SVC call to CARDRD (DMSCIO). The record in the CARD buffer is typed on the console via the WRTERM macro. If the BATMOVE flag is set on (MOVEFILE executing from the console), the records in the file are not typed on the console.

The record in the reader buffer is scanned to compute its length with trailing blanks deleted. It is then moved to the CMS console read buffer and the computed length is stored in the original DMSCRD parameter list, whose address is passed by DMSCRD when it initially passes control to DMSBTP.

If the first user record is not a /JOB card, error message DMSBTP105E is typed and normal cleanup is performed with the BATTERM flag set on. This flag prevents another initial program load, since it is not needed at this time. Reads to the card reader are then issued until the next /JOB card is found.

If the first record is a /JOB card, DMSBTP branches to its /JOB card processing routine which calls DMSSCNN via a BALR. A check is made for the existence of the userid and account number on the card. If the fields exist, a CP DIAGNOSE X'4C' is issued to start accounting recording for that userid and account number. If an error is returned from CP denoting an invalid userid, or if the userid or account number fields were missing on the /JOB card, error message DMSBTP106E is typed and normal cleanup is performed with the BATTERM flag set on.

The jobname, if provided on the /JOB card, is saved and a message is issued via SVC to inform the source userid that the job has started. The spooling devices are closed and respooled for continuous output, a CP QUERY FILES command is issued for information purposes and the implied CP function under CMS is disabled and the protection feature set off via SVC calls to SET (DMSSET). The BATPROF EXEC is executed via an SVC to EXEC. The BATNOEX flag, which is set by DMSBTB to suppress user job execution until the /JOB card is detected, is set off. The BATUSEX flag is set on (for DMSCPF) to signal the start of the actual user job, and a branch is taken to read the next card from the reader file (user job).

After reading the /JOB card, DMSBTP continues reading and checks for a /* card, a /SET card, or a CP command. If a card is none of these, DMSBTP passes control back to the command processor DMSINT for processing of the command (or data).

If a /* card is read and it is the first card of the new job, it is assume to be a precautionary measure and thus ignored by DMSBTP which then reads the next card. If it is not the first card a check is made for the BATMOVE flag. If the flag is on, the /* card indicates an end-of-file condition for the MOVEFILE operation from the console (reader) and is consequently translated to a null line for the MOVEFILE command.

If the BATMOVE flag is not on, the /* card is and end-of-job indicator and an immediate branch is taken to the end-of-job routine for cleanup and reloading of CMS batch.

When a CP command is encoutered DMSBTP branches to a routine that first checks a table of CP commands allowable in batch. If the command is allowed, a check is made for a reader or other spool device in the command line. If the CP command is allowed but would alter the status of the batch reader or any spooling device or certain disks, or if the command is not allowed at all, error message DMSBTP107E is typed, and the next card is read.

If the CP command is LINK, the device address is stored in a table so that DMSBTP can detach all user disk devices at the end of the job.

A CP DETACH command is examined for a device address corresponding to the system disk, the IPL disk, the batch 195 work disk or any spool device. If the device to be detached is any of these, error message DMSBTP107E is displayed and the next card is read. Otherwise, DMSBTP returns control to DMSINT (or DMSCPF is the BATCPEX flag is set on) for processing of the command.

When a /SET control card is encountered, the card is checked for valid keywords, valid integer values (less than or equal to the installation default values), and if an error is detected, error message DMSBTP108E is typed. An abnormal termination message is also sent to the source userid and the job is terminated with normal cleanup performed. If the control card values are valid, the appropriate fields are updated in the user job limit table DMSBTPLM and the next card is read.

If DMSBTP detects a "not ready" condition at the reader, a message is typed at the console stating that batch is waiting for reader input. DMSBTP then issues the WAITD macro to wait for a reader interrupt. When first detecting the empty reader, DMSBTP calls the CP accounting routines via a CP diagnose '4C' to charge the wait time to the batch userid.

If a hard error is detected at the reader, DMSBTP sends an "intervention required" message to the system console and branches to its abnormal terminal routine and waits for an interruption for the reader by issuing the WAITD macro.

When a /* card is read (with the BATMOVE flag off) or when the end-of-file condition occurs at the reader, DMSBTP branches to the cleanup routine which sends the source userid a message stating that the job ended normally or abnormally (if cleaning up after an abnormal termination) and turns off the BATUSEX flag (for DMSCPF) to signal the end of the user job. CONWAIT (DMSCWT) is called via SVC to allow any console I/O to finish, the spooling devices are closed (including the console), and all disks that were made available by issuing the CP LINK command are returned by issuing the CP DETACH command.

DMSBTP then relinquishes control by issuing the CP IPL command with the PARM BATCH option which loads a new CMS nucleus and the next job is started when CMS attempts its first read to the console.

A branch is made to the CMSBTP routine when DMSBTP itself detects an I/O error at the reader. However, the primary purpose of the routine is to receive control not only from DMSABN when there is an abnormal termination during the user job, but also from DMSITE, DMSPIO, and DMSCIO when a user job exceeds one of the batch job limits (BATXLIM flag is on). This routine, entry point DMSBTPAB, calls the CP DUMP routine via SVC and then branches to the cleanup routine which reloads CMS Batch and treat the remainder of the current job as a new job with no /JOB card. This has the effect of flushing the remainder of the job. This technique is used because batch must keep its reader spooled "continuous." Entry point DMSBTPAB is also used by the CMS commands that are disabled in CMS batch. In this case (BATDCMS flag set on), an error message is displayed and control returned to CMS.

When a CP command is called via an SVC in DMSBTP, the CMS CP module (DMSCPF) is actually called to issue the DIAGNOSE instruction to invoke the CP command. DMSBTP calls DMSCPF by issuing a direct SVC 202 or by issuing the LINEDIT macro with the CPCOMM option that generates an SVC 203.

Other CMS Modules Modified in CMS Batch

Several CMS modules check whether CMS batch is running, and, if so, perform functions associated with batch operation. These are shown in the following list:

Module	Function Performed for CMS Batch
DMSINI	Passes batch parameters to DMSINS.
DMSINS	Uses batch IPL parameters to reload CMS Batch.
DMSLDR	Loads DMSBTP into free storage.
DMSCRD	Passes control to DMSBTP to read from the reader rather than from the console.
DMSITE	Accounts for virtual time used by batch job -- ABEND if over limit.
DMSPIO	Accounts for number of lines printed by batch job -- ABEND if over limit.
DMSCIO	Accounts for number of cards punched by batch job -- ABEND if over limit.
DMSABN	Passes control to batch ABEND routine in DMSBTP.
DMSERR	Passes control to batch ABEND routine instead of entering disabled wait state.
DMSMVE	Turns the BATMOVE flag on and off -- allows batch to treat moved blanks as data.

DMSSET	Disabled if batch running, except during batch initialization.
DMSRDC	Disabled if batch running.
DMSCPF	Distinguishes between CP command issued by user and by batch.
DMSFLD	Disallows reader device specification.
DMSDSK	Disk load not allowed in batch.

ERROR PRINTOUTS

VM/370 error recording records and records passed via the SVC 76 by virtual machines are accumulated in chronological order on the VM/370 error recording cylinders. The following modules are used by CMS CPEREP to edit and print error records compiled by VM/370 as well as SYS1.LOGREC data sets:

Module	Function
DMSIFC	Checks some of the operands invoked by CPEREP for validity and passes the operands to IFCEREP1 for further processing.
DMSREA	Reads pages from the error recording cylinder and makes the records available to IFCPEREP1.
IFCEREP1	Selects error records according to supplied CPEREP operands or default values, and formats the records for output.

Detailed descriptions of the CPEREP command, the DMSIFC and DMSREA modules, and EREP (IFCEREP1) are found in the _VM/370 CLTSEP and Error Recording Guide_ and the _VM/370 Service Routines Program Logic_ with appropriate referrals to _OS/VS Environmental Recording, Editing, and Printing (EREP) Program_.

CMS Directories

This section contains the following information:

- Module Entry Point Directory
- Module-to-Label Cross Reference
- Label-to-Module Cross Reference

Module Name	Entry Points	Function
DMSABN	DMSABN	Intercepts an abnormal termination (ABEND) and provides recovery from the ABEND. Entered by a DMKABN TYPCALL=BALR macro call.
	DMSABNKX	Entered by a KXCHK macro to halt execution after HX has been entered after signaling attention.
	DMSABNGO	Entered by any routine that sets up ABNPSW and ABNREGS in the work area beforehand.
	DMSABNSV	Entered as the result of a DMSABN TYPCALL=SVC macro call.
	DMSABNRT	Returns entry point from DEBUG.
DMSACC	ACCESS	Accesses data in the ADT and related information (such as AFT's and chain links) in virtual storage.
DMSACF	READFST	Reads all file status table blocks into storage for a read/write disk. Reads in file management tables for a read - only disk. For an O/S disk, control returns to to the caller after a successful return from DMSACM.
DMSACM	READMFD	Reads the ADT, QMSK, QQMSK, and first chain link into virtual storage from the master file directory on disk.
DMSALU	RELUFD	For a specified disk, releases all tables kept in free storage and clears appropriate information in the active disk table (ADT).
DMSAMS	DMSAMS	Provides an interface to DOS Access Method Utility programs (IDCAMS). Provided for support of CMS/VSAM.
DMSARD	DMSARD	Provides storage for the ASM3705 assembler auxiliary directory. DMSARD contains no executable code. It must be loaded with DMSARX and the GENDIRT command must then be issued to fill in the auxiliary directory entries. GENMOD must then be issued to create the ASSEMBLE module.
DMSARE	DMSARE	Releases storage used for tables pertaining to a given disk when that disk is no longer needed.
DMSARN	DMSARN	This is the ASM3705 command processor. It provides the interface between user and the 370x Assembler.
	ASMHAND	This is the SYSUT2 processing routine called from DMSSOB and used during the assembly whenever any I/O activity pertains to the SYSUT2 file.
DMSARX	DMSARX	Provide an interface for the ASM3705 command to the 3705 assembler program.
DMSASD	DMSASD	Provides storage for the assembler auxiliary directory. DMSASD contains no executable code. It must be loaded with DMSASM and the GENDIRT command must then be issued to fill in the auxiliary directory entries. The GENMOD command must then be issued to create the assemble module.

Module Name	Entry Points	Function
DMSASM	DMSASM	Processes the ASSEMBLE command. Provides the interface between the user and the system assembler.
	ASMPROC	This is the SYSUT1 processing routine (called from DMSSOB).
DMSASN	DMSASN	Associates logical units with a physical hardware device. (Interface for the ASSGN command used by CMS/DOS and CMS/VSAM.)
DMSAUD	DMSAUD	Reserves space on disk for writing a copy of disk and and file management tables on disk and then updates the master file directory.
	DMSAUDUP	Closes all CMS files, thereby updating the master file Directory for any disks that had an output file open.
DMSBAB	DMSBAB	Give control to an abnormal termination routine once linkage to such a routine has been established by STXIT AB macro.
DMSBOP	DMSBOP	Opens CMS/DOS files associated with the following DTF (Define The File) tables: DTFCN, DTFCD, DTFPR, DTFMT, DTFDI, DTFCP, DTFSD. Once the files are opened and initialized, I/O operations can be performed using the file.
DMSBRD	DMSBRD (RDBUF)	Reads one or more successive items from a specified file.
DMSBSC	BASIC	Processes the BASIC command. The BASIC command invokes the CALL-OS BASIC language processor to compile and execute the specified file of BASIC source code.
DMSBTB	DMSBTB	This is the CMS batch bootstrap routine. It loads the batch processor routine (DMSBTP) and user exit routine (if they exist) into free storage.
DMSBTP	DMSBTP	Main entry; reads from the virtual card reader each time CMS tries to execute a console read.
	DMSBTPAB	Entry point for abnormal conditions during user job: • Job exectuion ABEND (from DMSABN) • Job limit exceeded (from DMSITE, DMSCIO, DMSPIO) • Disabled CMS command (from the command)
	DMSBTPLM	Non-executable user job limit table referenced by DMSITE, DMSPIO, and DMSCIO.
DMSBWR	DMSBWR	Writes one or more successive items into a specified disk file.
DMSCAT	DMSCAT	Stacks a line of console input that DMSCRD reads later when it is called.
DMSCIO	DMSCIOR	Reads one card record.
	DMSCIOP	Punches one card record.
	DMSCIOSI	Punch caller's buffer.

Module Name	Entry Points	Function
DMSCIT	DMSCIT	Processes the interruptions for all CMS terminal I/O operations and starts the next I/O operation upon completion of the current I/O operation.
	DMSCITA	Processes terminal interruptions.
	DMSCITB	Starts next terminal I/O operation.
	DMSCITDB	Frees I/O buffers from stacks.
DMSCLS	DMSCLS	Closes CMS/DOS files associated with the following DTF (Define The File) tables: DMTCN, DTFCD, DTFPR, DTFMT, DTFDI, DTFCP, and DTFSD. For reader, printer, or punch files, a CP CLOSE command is issued. For disk files, DMSFNS is called to close the file. For a disk work file, DMSERS is called to erase the file, unless DELETFL=NO is specified.
DMSCMP	COMPARE	Compares the records contained in two disk files.
DMSCPF	DMSCPF	Passes a command line to CP for execution.
DMSCPY	DMSCPY	Processes the COPYFILE command to copy disk files.
DMSCRD	DMSCRD	Reads an input line and makes it available to the caller.
DMSCWR	DMSCWR	Writes an output line to the console.
DMSCWT	DMSCWT	Causes the calling program to wait until all terminal I/O operations have been completed.
DMSDBD	DMSDBD	Enables a user to dump his virtual storage from within an executing program.
DMSDBG	DMSDBG	Enables the user to debug his program from the terminal.
	DMSDBGP	Entry point for program interruptions.
	DMSDBG	Entry point for all other interruptions.
DMSDIO	DMSDIOR	Reads one or more 800-byte records (blocks) from disk, or reads one 200-byte record (sub-block) from disk.
	DMSDIOW	Writes one or more 800-byte records (blocks) on disk, or writes one 200-byte record (subblock) on disk.
DMSDLB	DMSDLB	Interface for the DOS DLBL command; allows the user to specify I/O devices extents, and certain file attributes for use by a program at execution time. DLBL can also be used to modify or delete proviously defined disk file descriptions.
DMSDLK	DMSDLK	Interface for the DOS user command. Link-edit the relocatable output of the language processors. Once link-edited, these core image phases are added to the end of the specified DOSLIB.
DMKDMP	DMKDMP	Simulates the DOS/VS $$BDUMP and $$BPDCMP functions. For both functions, a CP DUMP command is issued, directing the dump to an offline printer.
DMSDOS	DMSDOS	Provides DOS SVC support. Interprets DOS SVC codes and passes control to appropriate routines for execution (for example, OPEN, CLOSE, FETCH, EXCP).

Module Name	Entry Points	Function
DMSDSK	DMSDSK	Dumps a disk file to cards or loads files from card to disk.
DMSDSL	DMSDSL	Provides capability to delete members (phases) of a DOSLIB library; also, to compress a DOSLIB library; also, to list the members (phases) of a DOSLIB library.
DMSDSV	DMSDSV	Lists the directories of DOS private or system packs.
DMSEDC	DMSEDC	Arranges compound (overstruck) characters into an ordered form and disregards tab characters as special characters.
DMSEDF	DMSEDF	Provides the Editor with the proper settings (CASE, TAB, FORMAT, SERIAL, etc.) by filetype. Contains nonexecutable code for reference by DMSEDI.
DMSEDI	DMSEDI	Modifies the contents of an existing file or creates a new file for editing.
DMSEDX	DMSEDX	Performs initialization fcr the CMS Editor.
DMSERR	DMSERR	Builds a message to be written at the virtual console by DMSCWR.
DMSERS	DMSERS	Deletes a file or related group of files from read/write disks.
DMSEXC	DMSEXC	Bootstrap loader for disk version of EXEC.
DMSEXT	DMSEXT	Processes the EXEC command.
DMSFCH	DMSFCH	Bring a specified phase into storage from a system or private core image library or from a CMS DOSLIB library. DMSFCH is invoked via SVC 1, 2, or 4 or via the FETCH command.
DMSFET	DMSFET	Provides an interface for the FETCH command; also, provides the capability to start execution of a specified phase.
DMSFLD	DMSFLD	Interprets OS JCL DD parameters for use by CMS.
DMSFNC	DMSFNC	Nucleus resident command name table.
	DMSFNCSV	Standard SVC table.
DMSFNS	DMSFNSA	Closes one or more input or output disk files.
	DMSFNSE	Closes a particular file without updating the directory or removing it from the active file table.
	DMSFNST	Temporarily closes all output files for a given disk.
DMSFOR	DMSFOR	Physically initializes a disk space for the CMS data management routines. For an existing disk, any information on the disk may be destroyed. The label may be changed and the number of cylinders allowed may be changed.

Module Name	Entry Points	Function
DMSFRE	DMSFREB	Called as a result of the DMSFREE and DMSFRET macro calls. Allocates or releases a block of storage depending upon the code in NUCON location CODE203.
	DMSFREES	Called as a result of the SVCFREE macro call. The size of the block is loaded from the PLIST and a DMSFREE macro is executed. Upon return, the address of the allocated block is stored into the PLIST.
	DMSFRETS	Called as a result of the SVCFRET macro call. The size and address of the block to be released are loaded from the PLIST and a DMSFRET macro is executed.
	DMSFREEX	Called as a result of a BALR to the address in the NUCON location AFREE. Executes the DMSFREE macro.
	DMSFRETX	Called as a result of a BALR to the address in the NUCON location AFRET. Executes the DMSFRET macro.
	DMSFRES	Called as a result of executing the DMSFRES macro. DMSFRES processes the following service routines: CKOFF, INIT1, INIT2, CHECKS, UREC, and CALOC.
DMSGIO	DMSGIO	Creates the DIAGNOSE and CCWs for an I/O operation to a display terminal from a virtual machine.
DMSGLB	DMSGLB	Defines the macro libraries to be searched during assembler processing. Defines text libraries to be searched by the loader for any unresolved external references.
DMSGND	DMSGND	Generates auxiliary system status table.
DMSGRN	DMSGRN	Edits STAGE1 output (STAGE2 input), builds 3705 assembler files, link-edits text files and an EXEC macro file.
DMSHDI	DMSHDI (HNDINT)	Sets the CMS interruption handling functions to transfer control to a given location for an I/O device other than those normally handled by CMS, or clears previously initialized I/O interruption handling.
DMSHDS	DMSHDS	Initializes the SVCINT SVC interruption handler to transfer control to a given location for a specific SVC number (other than 202) or to clear such previous handling.
DMSIFC	DMSIFC	Scans and passes all non-special parameters to the IFCEREP1 module, initializing values to edit and print records from VM/370's error recording cylinders.
	DMSIFC76	Immediately reflects SVC76 back to the calling routine.
	DMSIFC18	BLDL handler for IFCEREP1.
	DMSIFC0	EXCP handler for IFCEREP1.
DMSINA	DMSINA	Handles either user-defined synonyms or abbreviations or system-defined synonyms for command names.
DMSINDEX	DMSINDEX	Index of CMS listings in the microfiche deck.
DMSINI	DMSINIR	Reads a nucleus into main storage.
	DMSINIW	Writes a nucleus onto a DASD device.

Module Name	Entry Points	Function
DMSINM	DMSINM (GETCLK) (CMSTIMER)	Obtains the time from the CP timer.
DMSINS	DMSINS	Controls initialization of the CMS nucleus.
DMSINS	DMSINS	Controls initialization of the CMS nucleus.
DMSINT	DMSINT	Reads CMS commands from the terminal and executes them. Entry is from DMSINS.
	DMSINTAB	Entry from DMSABN.
	SUBSET	CMS subset entry.
DMSIOW	DMSIOW, WAIT, DMSIOWR, WAITRTN	Places the virtual CPU in the wait state until the completion of an I/O operation on one or more devices.
DMSITE	DMSITE, EXTINT, DMSITET, TRAP,	Processes external interruptions.
DMSITI DMSITI	DMSITI, IOINT,	This module is entered when an I/O operation causes the I/O new PSW to be loaded. This module handles all I/O interruptions, passes control to the interruption processing routine, and returns control to the interrupted program.
DMSITP	DMSITP	Processes program interruptions and processes SPIE exits.
DMSITS	DMSITS	Avoids CP overhead due to SVC call.
	DMSITS1	Address pointed to by the CMS SVC new PSW. This point is entered whenever an SVC interruption occurs.
	DMSITSCR	Return point to which a program called by a CMS SVC returns when it is finished processing.
	DMSITSOR	Return point to which a program called by an OS SVC returns when it is finished processing.
	DMSITSK	Called by an SVC by the DMSKEY macro.
	DMSITSXS	Called by an SVC from the DMSEXS macro.
	DMSITSR	This is the DMSITS recovery and reinitialization routine, called by DMSABN. DMSABN is the ABEND recovery routine.
DMSLAD	DMSLAD, ADTLKP	Finds the active disk table block whose mode matches the one supplied by the caller.
	DMSLADN, ADTNXT,	Finds the first or the next ADT block in the active disk table.
	DMSLADW	Finds the read or write disk according to input parameters.
	DMSLADAD	Modifies the file status table chain to include an auxiliary directory, or clears the auxiliary directory from the chain.

Module Name	Entry Points	Function
DMSLAF	DMSLAF, ACTLKP	Finds the active file table block whose filename, file-type, and filemode match the one supplied by the caller.
	DMSLAFNX, ACTNXT,	Finds the next or first AFT block in the active file table.
	DMSLAFFE ACTFREE	Finds an empty block in the active file table or adds a new block from free storage to the active file table, if necessary, and places a file status entry (if given) into the AFT block.
	DMSLAFFT ACTFRET	Removes an AFT block from the active file table and returns it to free storage if necessary.
DMSLBM	DMSLBM	Generates a macro library, adds macros to an existing library, and lists the dictionary of an existing macro library.
DMSLBT	DMSLBT, TXTLIB,	Creates a text library, adds text files to an existing text library, creates a disk file that lists the control section and entry point names in a text library or types, at the terminal, the control section and entry point names in a text library.
DMSLDR	DMSLDRA	Begins execution of a group of programs loaded into real storage. Definition of all undefined programs is established at location zero. Entered from the START command or internally from DMSLDRB LDT routine if START is specified.
	DMSLDRB	Processes TEXT files that may contain the following cards: SLC, ICS, ESD, TXT, REP, RLD, END, LDT, LIBRARY, and ENTRY. Entered from DMSLDP when the load function is requested.
	DMSLDRC	Does the processing required by various loader routines when an invalid card is detected in a text file.
	DMSLDRD	Does the processing required when a fatal I/O error is detected in a text file.
DMSLDS	DMSLDS	Lists information about specified data sets residing on an OS disk. Processes the LISTDS command.
DMSLFS	DMSLFS, TYPSRCH	Finds a specified 40-byte FST entry within the FST blocks for read-only or read/write disks.
DMSLGT	DMSLGTA	Entered from DMSLDRB if not a dynamic load. Frees all the TXTLIB blocks on the TXTLIB chain.
	DMSLGTB	Reads TXTLIB directories into a chain of free storage directory blocks. Entered from DMSLDRB.
DMSLIB	DMSLIB	Searches TEXT libraries for undefined symbols and closes the libraries.
DMSLIO	DMSLIO	Creates the load map on disk and types it at the terminal. Performs disk and typewriter output for DMSLDR.
DMSLKD	DMSLKD	Provides an interface between CMS and the VS1 linkage editor.
DMSLLU	DMSLLU	Lists the assignments of logical units.

Module Name	Entry Points	Function
DMSLOA	DMSLOA	Processes the LOAD and INCLUDE commands to invoke the relocating loader.
DMSLSB	DMSLSBA	Hexadecimal to binary conversion routine.
	DMSLSBB	Adds a symbol to the string of locations waiting for an undefined symbol to be defined.
	DMSLBC	Removes the undefined bit from the REFTBL entry and replaces the ADCON with the relocated value.
	DMSLBD	Processes LDR options.
DMSLST	DMSLSTA	Processes the LISTFILE command. Prints information about the specified files.
DMSLSY	DMSLSY	Generates a unique character string of the form Z000001 for private code symbols.
DMSMDP	DMSMSP	Types the load map associated with the specified file on the terminal.
DMSMOD	DMSMOD	Processes the GENMOD command to create a file that is a core image copy; processes the LOADMOD command to load a file that is in core image form.
DMSMVE	DMSMVE	Transfers data between two specified OS ddnames, the ddnames may specify any devices or disk files supported by the CMS system.
DMSNCP	DMSNCP	Reads a 3705 control program module (Emulator Program or Network Control Program) in OS load module format and writes a page-format core image copy on a VM/370 system volume.
DMSNUC	DMSNUC	Contains CSECTS for nucleus work areas and permanent storage.
	NUCON	Nucleus constant area.
	SYSREF	Nucleus address table.
	DEVTAB	Device table.
	ADTSECT	Active disk table.
	AFTSECT	Active file table.
	EXTSECT	External interruption storage.
	IOSECT	I/O interruption storage.
DMSNUC	PGMSECT	Program Interruption storage.
	SVCSECT	SVC interruption storage.
	DIOSECT	Disk I/O storage.
	FVS	File system storage.
	OPSECT	Parameter lists.
	CVTSECT	Simulated OS CVT.
	DBGSECT	Debug storage.
	TSOBLKS	TSO control blocks.
DMSOLD		Performs initialization and processing for each loading operation by processing text files that contain the following cards: SLC, ICS, ESD, TXT, REP, RLD, END, LDT, LIBRARY, and ENTRY.
	DMSOLD	Entered from DMSSLN when load requested.
	DMSLDRC	Entered when an invalid card is detected in a text file.
	DMSLDRD	Entered when a fatal error occurs during loading.

Module Name	Entry Points	Function
DMSOPL	DMSOPL	Reads the appropriate system directory records and headers and determines if the specified libraries contain any active members. Returns the disk address of the specified system library and indicates whether or not there are active members to be accessed on the disk.
DMSOPT	DMSOPT	Sets DOS options in the System Communications Region as specified by the OPTION command.
DMSOR1	DMSOR1	Relocates all DFT (Define The File) Table address constants to executable storage addresses. (Called by $$BOPENR via SVC 2.)
DMSOR2	DMSOR2	Relocates all DTF (Define The File) Table address constants to executable storage addresses. (Called by DMSOR1.)
DMSOR3	DMSOR3	Relocates all DTF (Define The File) Table address constants to executable storage addresses. (Called by DMSOR2.)
DMSOVR	DMSOVR	Analyzes the SVCTRACE command parameter list and loads the DMSOVS tracing routine.
DMSOVS	DMSOVS	Provides trace information requested by the SVCTRACE command.
DMSPIO	DMSPIO	Prints one line.
	DMSPIOCC	Puts CCWs and data into the caller's buffer.
	DMSPIOSI	Prints the caller's buffer, issues an SIO to the virtual printer, and analyzes the resulting status.
DMSPNT	DMSPNT	Places the address of a file status table entry in the active file table (if necessary), and sets the read pointer or write pointer for that file to a given item number within the file.
DMSPRT	DMSPRT	Prints CMS files.
DMSPRV	DMSPRV	Copies procedures from the DOS/VS system procedure library to a specified output device.
DMSPUN	DMSPUN	Punches CMS files to the virtual card punch.
DMSQRY	DMSQRY	Processes the QUERY command. Displays at the user's terminal, the status of various CMS functions and tables.
DMSRDC	READCARD	Reads cards and assigns the indicated filename.
DMSREA	DMSREA	Reads error recording cylinder pages into storage for EREP (IFCEREP1) processing. It passes one logical record for each read request.
DMSRNE	DMSRNE	Provides an interface for the CMS Editor RENUM subcommand, which renumbers files with filetypes of VSBASIC and FREEFORT.

Module Name	Entry Points	Function
DMSRNM	DMSRNM	Processes the RENAME command. Changes the fileid of the specified file.
DMSROS	DMSROS ROSACC	Accesses OS disks.
	DMSROS+4 ROSSTT	Verifies the existence of OS disks.
DMSROS	DMSROS+8 ROSRPS	Reads OS disks.
	DMSROS+12 ROSFIND	Finds a member in an OS PDS.
	DMSROS+16 ROSNTPTB	Performs NOTE, POINT, and BSP functions.
DMSRRV	DMSRRV	Provides the capability to copy (to an output device) modules residing on DOS system or private relocatable libraries.
DMSSAB	DMSSAB	Processes OS ABEND macros.
DMSSBD	DMSSBD	Accesses data set records directly by item number. It converts record identifications given by OS BDAM macros into item numbers and uses these item numbers to access records.
DMSSBS		Processes OS BSAM READ and WRITE macros.
	DMSSBSRT	Entry for error return from call to DMSSBD.
DMSSCN	DMSSCN	Transforms the input line from a series of arguments to a series of 8-byte parameters.
DMSSCR	DMSSCR	Loads display buffers and issues a macro resulting in a CP DIAGNOSE to write to the display terminal.
DMSSCT	DMSSCTNP	Processes OS POINT, NOTE, CHECK, and FIND (type C) macros.
	DMSSCTCK	Processes OS CHECK macro.
	DMSSCTCE	Handles QSAM I/O errors for DMSSQS and PDS and keys errors for DMSSOP.
DMSSEB	DMSSEB	Calls device I/O routines to do I/O and sets up ECB and IOB return codes.
DMSSEG	DMSSEG	Contains a table of VCONS for CMS saved segment entries.
DMSSET	DMSSET	Processes the SET command.
DMSSLN	DMSSLN	Handles OS contents management requests issued under CMS (LINK, LOAD, XCTL, DELETE, ATTACH, EXIT).
DMSSMN	DMSSMN	Processes OS FREEMAIN and GETMAIN macros and CMS calls DMSSMNSB and DMSSMNST.
DMSSOP	DMSSOP	Processes OS OPEN and CLOSE macros.
DMSSQS	DMSSQS	Analyzes record formats and sets up the buffers for GET, PUT, and PUTX requests.

Module Name	Entry Points	Function
DMSSRT	DMSSRT	Arranges records within a file in descending sequential order.
DMSSRV	DMSSRV	Provides capability to copy books from a system or private source statement library to a specified output device.
DMSSSK	DMSSSK	Sets storage protect key for a specified saved system.
DMSSTG		Processes CMS calls to DMSSTGST and DMSSTGSB (STRINIT) and storage service routines.
	DMSSTGSB	STRINIT.
	DMSSTGST	
	DMSSTGCL	OS exit reset routine.
	DMSSTGSV	Service routine to change nucleus variables.
	DMSSTGAT	Initializes storage and sets up an anchor table.
DMSSTT	DMSSTT	Locates the file status table entry for a given file and, if found, provides the caller with the address of the entry.
DMSSVN	DMSSVN	Processes the OS WAIT and POST macros.
DMSSVT	DMSSVT	Processes OS macros: XDAP, TIME, SPIE, RESTORE, BLDL, FIND, STOW, DEVTYPE, TRKBAL, WTO, WTOR, EXTRACT, IDENTIFY, CHAP, TTIMER, STIMER, DEQ, SNAP, ENQ, FREEDBUF, STAE, DETACH, CHKPT, RDJFCB, SYNAD, BACKSPACE, and STAX.
DMSSYN	SYNONYM	Processes the SYNONYM command. Sets up user-defined command names and abbreviations for CMS commands.
DMSTIO	DMSTIO	Reads or writes a tape record or controls tape positioning.
DMSTMA	DMSTMA	Reads an IEHMOVE unloaded PDS from tape and places it in a CMS MACLIB.
DMSTPD	DMSTPD	Reads a tape consisting of card image members of a PDS and creates CMS disk files for each member of the data set. The PDS option allows reading unblocked tapes produced by the OS IEBPTPCH utility or blocked tapes produced by the OS IEHMOVE utility. The UPDATE option provides the "./ ADD" function to blocked or unblocked tapes produced by the IEBUPDTE utility.
DMSTPE	DMSTPE	Processes the TAPE command to perform certain tape functions, such as: dump a CMS file, load a CMS file, set tape mode, scan, skip, rewind, run, FSF, FSR, BSF, BSR, ERG, and WTM.
DMSTQQ	DMSTQQ	Allocates a 200-byte first chain link (FCL) to a calling program.
	DMSTQQX	Makes a 200-byte disk area no longer needed by one program available for allocation to another program.
DMSTRK	DMSTRKA	Allocates an 800-byte disk area to a calling program.
	DMKSTRKX	Makes an 800-byte disk area that is no longer needed by one program available for allocation to another.

Module Name	Entry Points	Function
DMSTYP	TYPE	Processes the TYPE command. Types all or a specified part of a given file on the user's console.
DMSUPD	DMSUPD	Processes the UPDATE command. Updates source files according to specifications in update files. Multiple updates can be made, according to specifications in control files that designate the update files.
DMSVAN	DMSVAN	Contains table of Access Method Services nonshared (nonreentrant) modules.
DMSVAS	DMSVAS	Contains a table of Access Method Services shared (reentrant) modules.
DMSVIB	DMSVIB	Loads the CMS/VSAM saved system and pass control to the CMS/VSAM interface routine, DMSVIP.
DMSVIP	DMSVIP	Finds the CMS/DOS discontiguous shared segment (DCSS); issues all necessary DOS ASSGN statements for CS user; maps all OS VSAM macro requests to DOS specifications; equivalents, where necessary; traps all transfers of control between VSAM and the OS user and sets the appropriate operating environment flags.
DMSVPD	DMSVPD	Reads DOS, VSAM, and Access Method Services modules from a DOS PTF tape and writes the modules to the CMS user's A-disk.
DMSVSR	DMSVSR	Resets any flags or fields set by VSAM processing; purges the VSAM discontiguous shared segment.
DMSV33	DMSV33	Contains a table of VSAM shared (reentrant) modules and is contained within the CMSVSAM shared system. Used by CMSVSAM and VSAMGEN to generate the CMSVSAM shared system, and by CDLOAD to locate the phases within CMSVSAM. Used for system generation from the DOS/VS Release 33 restored starter system. Contains no executable code.
DMSXCP	DMSXCP	Simulates the DOS EXCP function (DOS SVC 0) in the CMS/DOS environment. EXCP (Execute Channel Program) requests initiation of an I/O operation to a specific logical unit.
DMSZAP	DMSZAP	Processes the ZAP command. Provides a facility to maintain CMS LOADLIB members as written by the CMS command LKED.
DMSZAT	DMSZAT	Defines 8K-bytes of transient area.
DMSZIT	DMSZIT	Defines the end of the CMS nucleus.
DMSZNR	DMSZNR	Defines the end of NUCON (DMSNUC).
DMSZUS	DMSZUS	Defines the start of the user area.

```
MODULE       EXTERNAL REFERENCES (LABELS AND MODULES)

DMSABN    ABATABND ABNBIT   ABNERLST ABNPAS13 ABNPSW   ABNREGS  ABNRR    ABWSECT  ADMSFREB ADTFDA   ADTFFSTF ADTFLG1  ADTFLG2
          ADTFMIN  ADTFQQF  ADTFROS  ADTHBCT  ADTM     ADTMFDA  ADTMFDN  ADTPQM3  ADTSECT  AFVS     AINTRTBL AIOSECT  AOPSECT
          AOUTRTBL ASUBFST  ASUBSECT ASUBSTAT ATTN     AUSABRV  AUSRAREA AUSRILST AUSRITBL BALR     BATFLAGS BATFLAG2 BATLOAD
          BATRUN   BATSYSAB CMNDLINE CODE203  CONRDCNT CONRDCOD CONREAD  CURRSAVE DBGAEN   DBGEXEC  DBGFLAGS DBGNSHR  DBGSHR
          DCSSFLAG DCSSVTLD DMSABW   DMSCAT   DMSCITDB DMSCRD   DMSCWT   DMSDBG   DMSERR   DMSEXCAB DMSFRES  DMSINTAB DMSITSR
          DMSLADAD DMSLADN  DMSSTGSB DOSFIRST DOSFLAGS DOSMODE  DOSNUM   DOSSVC   DOSTRANS EGPRS    FCBFIRST FCBNUM   FREELOWE
          FVSECT   IONTABL  IOSECT   IPLPSW   KXFLAG   KXWANT   LDMSRCS  LOC      MACDIRC  MISFLAGS NOPAGREL NRMRET   NUCON
          NUM      NUMFINRD OLDPSW   OPSECT   OPTFLAGS OSADTFST OSFST    OSFSTLTH OSFSTNXT OSMODLDW PGMNPSW  PGMOPSW  RELPAGES
          R0       R1       R12      R13      R14      R15      R2       R3       R4       R5       R6       R7       R8
          R9       SSAVE    SUBFLAG  SUBSECT  TEXT     UFDBUSY  USERKEY  VSAMFLG1 VSAMRUN  VSAMSOS  WAIT

DMSACC    ADMSFREB ADTDTA   ADTFALUF ADTFDA   ADTFDOS  ADTFFSTF ADTFFSTV ADTFLG1  ADTFLG2  ADTFLG3  ADTFMIN  ADTFORCE ADTFRO
          ADTFROS  ADTFRW   ADTFSTC  ADTHBCT  ADTLHBA  ADTM     ADTMFDN  ADTMSK   ADTMX    ADTNUM   ADTPQM2  ADTPQM3  ADTRES
          ADTSECT  ADTUSED  ADT1ST   AFINIS   AFVS     AKILLEX  BALR     CODE203  CURRSAVE DTAD     EGPR0    ERRCODE  FSTFMODE
          FSTFNAME FSTFTYPE FVSECT   IADT     KXFLAG   KXWANT   LOC      MISFLAGS NUCON    NUM      RESET    R0       R1
          R10      R11      R12      R13      R14      R15      R2       R3       R4       R5       R6       R7       R8
          R9       SSAVE    TEXT     TEXTA    TYPE     UFDBUSY  VCADTLKP VCADTNXT VCFSTLKP VIRTUAL  WRBIT

DMSACF    ADMSFREB ADTADD   ADTCFST  ADTCHBA  ADTFALNM ADTFALTY ADTFALUF ADTFDA   ADTFFSTF ADTFLG1  ADTFLG2  ADTFLG3  ADTFMDRO
          ADTFORCE ADTFRO   ADTFROS  ADTFRW   ADTFSORT ADTFSTC  ADTFTYP  ADTHBCT  ADTLHBA  ADTM     ADTMFDA  ADTMFDN  ADTPQM2
          ADTRES   ADTSECT  AFVS     ARDTK    ATYPSRCH BALR     CODE203  DSKADR   DSKLOC   DSKLST   ERBIT    ERRCOD1  FSTIC
          FSTRP    FSTSECT  FSTT     FSTWP    FVSECT   F65535   JSR0     LOC      NUCON    REGSAV0  REGSAV1  RWCNT    R0
          R1       R10      R11      R12      R13      R14      R15      R2       R3       R4       R5       R6       R7
          R8       R9       TYPE     UFDBUSY

DMSACM    ADIOSECT ADMSFREB ADMSROS  ADTADD   ADTCYL   ADTDTA   ADTFLG1  ADTFLG2  ADTFLG3  ADTFMFD  ADTFORCE ADTFQQF  ADTFRO
          ADTFRW   ADTHBCT  ADTID    ADTMFDN  ADTMSK   ADTMX    ADTMXBML ADTNUM   ADTPQM1  ADTPQM2  ADTPQM3  ADTQQM   ADTRES
          ADTROX   ADTSECT  ADTUSED  AFVS     ARDTK    BALR     CDMSRCS  CODE203  DIOSECT  DSKADR   DSKLOC   DSKLST   DTAD
          DTADT    ERRCOD0  ERROR    FFD      FFE      FFF      FILE     FVSDSKA  FVSECT   FVSFSTIC FVSFSTIL F800     JSR0
          LDMSROS  LOC      LOCCNT   MODFLGS  NUCON    OSADTVTA QQDSK1   REGSAV0  RWMFD    R0       R1       R10      R11
          R12      R13      R14      R15      R2       R3       R4       R5       R6       R7       R8       R9       SECTNUM
          SEEKADR  SENSB    SIGNAL   SWTCH    SYSLOAD  TBENT    TEXT     TYPE     UFDBUSY  UPBIT    VCADTLKP

DMSALU    ABGCOM   ADMSFREB ADMSROS  ADTFDA   ADTFFSTF ADTFLG1  ADTFLG2  ADTFLG3  ADTFMIN  ADTFQQF  ADTFRO   ADTFROS  ADTFRW
          ADTFSTC  ADTFTYP  ADTID    ADTM     ADTMFDN  ADTMSK   ADTMX    ADTPQM1  ADTPQM3  ADTQQM   ADTRES   ADTROX   ADTSECT
          AFVS     BALR     CDMSROS  CODE203  DOSFLAGS DOSMODE  FCBDSMD  FCBFIRST FCBNEXT  FCBOSFST FCBSECT  FLGSAVE  FVSECT
          LDMSROS  LOC      NUCON    OSADTFST OSFST    OSFSTLTH OSFSTNXT REGSAV0  R0       R1       R10      R11      R12
          R13      R14      R15      R2       R3       R4       R5       R6       R7       R8       R9       SDISK    STATEFST
          VCADTLKP VCADTNXT

DMSAMS    AAMSSYS  ABGCOM   ADEVTAB  ADMSERL  ADMSFREB ADTM     ADTSECT  AERASE   ALTASAVE APPSAVE  ASCANN   ASTATE   ASTATEW
          ASYSNAMS ATABEND  BALR     BGCOM    CMSAMS   CODE203  COMNAME  DOSDD    DOSDEV   DOSDSMD  DOSDUM   DOSEXTNO DOSEXTTB
          DOSFIRST DOSFLAGS DOSMODE  DOSNEXT  DOSRC    DOSSECT  DOSSVC   DOSVOLNO DOSVCLTB DOSYSXXX DTAD     DTAS     ERRMSG
          FSTFV    FSTIL    FSTM     FSTN     FSTSECT  F4096    LOC      LTK      LUBPT    MISFLAGS NUCON    NUM      PIBPT
          PUBPT    RELPAGES RESET    R0       R1       R10      R11      R12      R13      R14      R15      R2       R3
```

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	R4	R5	R6	R7	R8	R9	SYSNAMES	SYSNEND	TEXT	TEXTA	VCADTLKW	VIRTUAL	VMSIZE
	VSAMFLG1	VSAMSERV	VSAMSOS										
DMSARE	ABATPROC	ADTDTA	ADTFLG1	ADTFLG2	ADTFLG3	ADTFNOAB	ADTFRC	ADTFROS	ADTFRW	ADTFSTC	ADTM	ADTSECT	AFINIS
	AUPDISK	BATCPEX	BATFLAGS	BATLOAD	BATRUN	BATUSEX	DTAD	NUCON	NUM	R0	R1	R10	R11
	R12	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	TEXT	VCADTLKP
	VCADTNXT												
DMSARN	ADTFLG1	ADTFRW	ADTM	ADTMX	ADTSECT	AOPSECT	ASTRINIT	BATFLAGS	BATRUN	COMPSWT	ERRCODE	ERROR	FCBBUFF
	FCBBYTE	FCBCATML	FCBCLOSE	FCBDD	FCBDEV	FCBFORM	FCBINIT	FCBIOSW	FCBITEM	FCBPROC	FCBPROCC	FCBPROCO	FCBREAD
	FCBSECT	FINIS	FSTL	FSTM	FSTSECT	INPUT	IOBCSW	IOBIN	IOBIOFLG	MISFLAGS	NOERASE	NUCON	NUM
	OSSFLAGS	RELPAGES	RESET	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3
	R4	R5	R6	R7	R8	R9	TEXT	VCADTLKW	VIRTUAL				
DMSARX	AADTLKW	ADTFLG1	ADTFRW	ADTM	ADTMX	ADTSECT	CC	CMNDLINE	COMPSWT	CONCNT	CONWR	DEVICE	DMSARD
	ERROR	FCBBUFF	FCBBYTE	FCBCATML	FCBCLOSE	FCBDD	FCBDEV	FCBDSK	FCBDSNAM	FCBFORM	FCBINIT	FCBIOSW	FCBITEM
	FCBPROCC	FCBRDR	FCBREAD	FCBSECT	FCBTAP	FILE	FLAG1	FLAG2	FREELOWE	FSTFV	FSTIL	FSTL	FSTM
	FSTSECT	IOBCSW	IOBIN	IOBIOFLG	MAINHIGH	MISFLAGS	NOERASE	NUCON	NUM	OPSECT	OSIOTYPE	OSSFLAGS	RELPAGES
	RESET	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R8	R9	SYSUT1	TEXT							
DMSASM	AADTLKW	ADTFLG1	ADTFRW	ADTM	ADTMX	ADTSECT	CC	CMNDLINE	COMPSWT	CONCNT	CONWR	DEVICE	DMSASD
	DOSFLAGS	DOSSVC	DUMMY	ERROR	FCBBUFF	FCBBYTE	FCBCATML	FCBCLOSE	FCBDD	FCBDEV	FCBDSK	FCBDSNAM	FCBFORM
	FCBINIT	FCBIOSW	FCBITEM	FCBPROCC	FCBRDR	FCBREAD	FCBSECT	FCBTAP	FILE	FLAG1	FLAG2	FREELOWE	FSTFV
	FSTIL	FSTL	FSTM	FSTSECT	IOBCSW	IOBIN	IOBIOFLG	MAINHIGH	MAX	MISFLAGS	NOERASE	NUCON	NUM
	OPSECT	OSIOTYPE	OSSFLAGS	PRFUSYS	PROTFLAG	RELPAGES	RESET	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVEREGS	SYSUT1	TEXT
DMSASN	ABATABND	ABGCOM	ADEVTAB	ADTDTA	ADTFDOS	ADTFLG1	ADTFLG2	ADTFRO	ADTFROS	ADTFRW	ADTSECT	ASYSREF	BATDCMS
	BATFLAGS	BATFLAG2	BATRUN	BGCOM	CLASDASD	CLASTAPE	CLASURI	CLASURO	DEVTAB	DOSFLAGS	DOSMODE	DOSVSAM	DTAD
	DTADT	FLAG2	FLAG3	FTRUCS	FTR35MB	NUCON	NUM	PACK	PUBPT	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SYSTEM
	TAPE1	TAPE4	TEXT	TYP1403	TYP2314	TYP2401	TYP2415	TYP2420	TYP2501	TYP2540P	TYP2540R	TYP3203	TYP3211
	TYP3340	TYP3420	TYP3525	VCADTLKP									
DMSAUD	ADMSFREB	ADTADD	ADTDTA	ADTFDA	ADTFLG3	ADTFNOAB	ADTFUPD1	ADTHBCT	ADTLAST	ADTMFDA	ADTMFDN	ADTMSK	ADTNUM
	ADTPQM1	ADTPQM2	ADTSECT	AFVS	AKILLEX	ATRKLKP	ATRKLKPX	AWRTK	BALR	CODE203	DSKADR	DSKLOC	DSKLST
	DTADT	FFD	FFE	FFF	FINISLST	FVSDSKA	FVSECT	F3	F800	KXFLAG	KXWANT	LOC	NUCON
	REGSAVO	RWCNT	RWFSTRG	RWMFD	R0	R1	R10	R11	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	TYPE	UFDBUSY	UPBIT			
DMSBAB	ABGCOM	ASYSCOM	BGCOM	DOSRC	IJBABTAB	NUCON	OLDPSW	OSTEMP	PCPTR	PIBADR	PIBPT	PIBSAVE	PIK
	R0	R1	R10	R12	R13	R14	R15	R2	R3	R4	R5	R6	R8
	R9	SSAVE	SVEARA	SVEPSW	SVEPSW2	SVEROF	SVER00	SVER01	SVER09	SYSCOM	VSAMFLG1	VSAMSERV	
DMSBOP	ABGCOM	ACBCAT	ACBDDNM	ACBERFLG	ACBIN	ACBINFLG	ACBMACR1	ACBOFLGS	ACBOLIGN	ACBOUT	ACBSTSKP	ADMSERL	ADMSFREB

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	ADTFDOS	ADTFLG1	ADTFLG2	ADTFLG3	ADTFMFD	ADTFRO	ADTFRCS	ADTFRW	ADTSECT	AERASE	ASTATE	ASYSCOM	ASYSNAMS
	ASYSREF	AVSAMSYS	BALR	BGCOM	BLANKS	BSR	BUFFER	CC	CMSVSAM	CODE203	COMNAME	CONSOLE	DEC
	DEVCODE	DOSBLKSZ	DOSBUFF	DOSDD	DOSDEV	DOSDSMD	DOSDUM	DOSEXT	DOSEXTCT	DOSFIRST	DOSFLAGS	DOSFORM	DOSINIT
	DOSNEXT	DOSNUM	DOSOP	DOSOSFST	DOSRC	DOSSECT	DOSSYS	DOSTRANS	DOSUCAT	DOSUCNAM	DOSVSAM	DOSYSXXX	DOUBLE
	EQCHK	FILE	FILETYPE	FREELN	FSTIC	FSTM	FSTSECT	F7	HOLD	IC	IJBFLG04	IKQACB	INPUT
	LOC	LUBPT	NICLPT	NUCON	NUM	ON	OSFST	OSFSTFM	OSFSTRFM	OSFSTXNO	OSFSTXTN	PACK	PIBPT
	PLIST	PUBADR	PUBCUU	PUBDEVT	PUBPT	PUBTAPM1	PUBTAPM2	PUBTAP7	READ	RESET	RMSROPEN	R0	R1
	R10	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SAVE1	SAVE2	SENSE	SKIP	SYSCOM	SYSNAMES	SYSNEND	TEMPSAVE	TEXT	TYPE	TYP2314	TYP3330	TYP3340
	TYP3350	VCADTLKP	VIPSOP	VMSIZE	VSAMFLG1	VSAMRUN	VSAMSERV	WRITE	WTM				
DMSBRD	AACTFREE	AACTLKP	ACTIVE	ADMSFREB	AFTADT	AFTCLA	AFTCLB	AFTCLD	AFTCLN	AFTDBA	AFTDBD	AFTDBN	AFTFBA
	AFTFCL	AFTFCLA	AFTFLG	AFTFST	AFTFV	AFTIC	AFTID	AFTIL	AFTIN	AFTRD	AFTRP	AFTSECT	AFTWRT
	AFVS	ARDTK	AUSRAREA	BALR	BALR9	CODE203	DISK$SEG	DMSLFS	FSCBD	FSCBFLG	FSCBFV	FSTFV	FSTIC
	FSTITAV	FSTNOIT	FSTRECAV	FSTRP	FSTSECT	FVSECT	ITEM	NUCON	PLIST	READ	READCNT	REGSAV3	RWFSTRG
	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6
	R7	R8	R9	STATEFST	STATER0	TYPE	VMSIZE						
DMSBTB	ABATABND	ABATLIMT	ABATPROC	AFVS	ALDRTBLS	AUSRAREA	BATDCMS	BATFLAGS	BATFLAG2	BATLOAD	BATNOEX	BATRUN	BATUSEX
	FVSECT	FVSFSTIC	FVSFSTIL	LOCCNT	NUCON	NUM	RESET	R0	R1	R12	R14	R15	R2
	R3	R4	R5	R8	TBENT	TEXT	TYPE						
DMSBTP	ABNBIT	ADMSCRD	AFVS	ASCANN	ASYSNAMS	BATCPEX	BATDCMS	BATFLAGS	BATFLAG2	BATMOVE	BATNOEX	BATRERR	BATSTOP
	BATTERM	BATUSEX	BATXCPU	BATXLIM	BATXPRT	BLK	CMSSEG	EDIT	ERROR	FVSECT	IPLADDR	KEYS	LINE
	NUCON	NUM	NUMFINRD	OFF	PACK	RESET	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SYSNAME	SYSNAMES	SYSNEND	TEXT
	UFDBUSY												
DMSBWR	AACTFREE	AACTFRET	AACTLKP	ADMSERL	ADMSFREB	ADTDTA	ADTFLG1	ADTFLG3	ADTFRW	ADTFSTC	ADTFXCHN	ADTM	ADTMX
	ADTNACW	ADTRES	ADTSECT	AFTADT	AFTCLA	AFTCLB	AFTCLD	AFTCLDX	AFTCLN	AFTCLX	AFTD	AFTDBA	AFTDBC
	AFTDBD	AFTDBF	AFTDBN	AFTFBA	AFTFCL	AFTFCLA	AFTFCLX	AFTFLG	AFTFLG2	AFTFST	AFTFULD	AFTFV	AFTIC
	AFTID	AFTIL	AFTIN	AFTM	AFTN	AFTNEW	AFTOCLDX	AFTOLDCL	AFTRD	AFTRP	AFTSECT	AFTWP	AFTWRT
	AFVS	AKILLEX	AQQTRK	AQQTRKX	ARDTK	ATFINIS	ATRKLKP	ATRKLKPX	AUPDISK	AWRTK	BALR	CODE203	DMSERR
	DMSLAD	DMSLFSW	FSTFV	FSTIL	FSTSECT	FSTWP	FVSECT	KXFLAG	KXWANT	LOC	NUCON	NUM	PLIST
	REGSAV3	RESET	RWFSTRG	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3
	R4	R5	R6	R7	R8	R9	TEXT	TEXTA	TYPE	UFDBUSY	VIRTUAL	VMSIZE	WRBIT
DMSCAT	ADMSFREB	BALR	CMNDLIST	CODE203	FSTFINRD	MISFLAGS	MSGFLAGS	NEGITS	NOTYPING	NUCON	NUMFINRD	R0	R1
	R12	R14	R15	R2	R3	R4	TYPE						
DMSCIO	ABATABND	ABATLIMT	ADMSERL	BATFLAGS	BATLSECT	BATNOEX	BATPUNC	BATPUNL	BATRUN	BATXLIM	BATXPUN	BUSY	CAW
	CSW	DE	ERRET	ERRMSG	NUCON	NUM	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	TEXTA	WAIT			
DMSCIT	ACTIVE	ADMSFREB	AFVS	AIOSECT	ASVCSECT	ATTN	ATTNHIT	BALR	BATFLAG2	BATSTOP	CAW	CE	CMSTAXE
	CODE203	CONCCWS	CONSTACK	CSW	CURRIOOP	DBGEXEC	DBGEXINT	DBGFLAGS	DE	DMSERR	FSTFINRD	FVSECT	IOOPSW

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	KXFLAG	KXWANT	LOC	LSTFINRD	MISFLAGS	MSGFLAGS	NOTYPING	NUCON	NUMFINRD	NUMPNDWR	OSSFLAGS	OSWAIT	OVSHO
	OVSON	OVSSO	OVSTAT	PACK	PENDREAD	PENDWRIT	R0	R1	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SVCSECT	TAIEIAD	TAIEMSGL	TAIERSAV	TAXEADDR	TAXEEXIT
	TAXEEXTS	TAXEFREQ	TAXEIOL	TAXEIOWS	TAXELNK	TAXERTNA	TAXESTAT	TAXETAIE	TAXETSOF	TEXT	TSOATCNL	TSOFLAGS	UE
	UFDBUSY	WAIT	WAITSAVE										
DMSCLS	ACBAMO	ADMSERL	ADMSFREB	AERASE	AFINIS	ASYSREF	AVSAMSYS	AVSRWORK	BALR	BGCOM	BSR	BUFFER	CODE203
	CPSTAT	DE	DEVCODE	DOSDD	DOSDSNAM	DOSDSTYP	DOSFIRST	DOSNEXT	DOSSECT	DOSTRANS	DOSYSXXX	DOUBLE	FILE
	FILETYPE	FREELN	IKQACB	LASTREC	LOC	LUBPT	NICLPT	NUCON	NUM	OFF	PIBPT	PLIST	PUBADR
	PUBCUU	PUBDEVT	PUBPT	PUBTAPM1	READ	RESET	RUN	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SENSE	TAPE	TEXT
	TYPE	VIPINIT	VIPSOP	VIPTCLOS	VSAMFLG1	VSAMSERV	WORKFILE	WRITE	WTM				
DMSCMP	ADMSFREB	ADTM	ADTSECT	AFINIS	ARDBUF	AREA	BALR	CODE203	ERROR	FILE	LOC	NUCON	NUM
	READ	R0	R1	R10	R11	R12	R14	R15	R2	R3	R4	R5	R6
	R7	R8	R9	SAVE	TEXT	TYPE	VIRTUAL						
DMSCPF	ABATPROC	BALRSAVE	BATCPEX	BATFLAGS	BATLOAD	BATRUN	BATUSEX	BS	CMNDLINE	CMNDLIST	NUCON	R0	R1
	R10	R12	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	
DMSCPY	AACTLKP	AADTLKW	ADTCFST	ADTCHBA	ADTFLG1	ADTFRW	ADTM	ADTSECT	AFSTLKP	AFSTLKW	AFTIC	AFTSECT	BLANKS
	BUFAD	CL	CODE	DOSFLAGS	DOSSVC	FSTD	FSTFAW	FSTFB	FSTFV	FSTIC	FSTIL	FSTITAV	FSTM
	FSTN	FSTSECT	FSTYR	HEX	INPUT	MISFLAGS	NUCON	NUM	OPSECT	PACK	RELPAGES	RESET	R0
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	TEXT	TYPE	UNPACK								
DMSCRD	ABATPROC	ADMSFREB	AFVS	AINTRTBL	AOPSECT	ATTN	BALR	BATFLAGS	BATLOAD	BATRUN	CODE203	CONINBLK	CONINBUF
	CSW	DMSCAT	DMSCITB	DMSERR	FSTFINRD	FVSECT	F255	KXFLAG	KXWSVC	LOC	LSTFINRD	MISFLAGS	MSGFLAGS
	NOTYPING	NUCON	NUMFINRD	NUMPNDWR	OPSECT	PENDREAD	QSWITCH	R0	R1	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R8	R9	TEXT	TSOATCNL	TSOFLAGS	UCASE	WAIT
	WAITLST												
DMSCWR	ADMSFREB	AFVS	AOPSECT	AOUTRTBL	BALR	CODE203	CONSOLE	CONSTACK	CSW	C1	DMSCITA	DMSCITB	DMSERR
	FVSECT	F256	KXFLAG	KXWSVC	MSGFLAGS	NOTYPING	NUCON	NUMPNDWR	OPSECT	PENDREAD	PENDWRIT	REDERRID	R0
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	TEXT	WAIT	WAITLST								
DMSCWT	AFVS	AOPSECT	FVSECT	KXFLAG	KXWSVC	NUCON	NUMPNDWR	OPSECT	PENDREAD	R0	R1	R10	R11
	R12	R14	R15	R9	WAIT	WAITLST							
DMSDBD	ADEVTAB	ARGS	BLANKS	CAW	CCWPRINT	CONHCT	CPULOG	DBDDMSG	DBDEXIT	DBGFLAGS	DBGOUT	DBGRECUR	DBGSECT
	DBGSWTCH	DEC	DECDEC	DEVTAB	F4096	INPUT	LASTLINE	LINE	LINE1	LINE1A	LINE1B	LINE1C	MVCNT1
	NUCON	PRINTER1	R0	R1	R10	R11	R14	R15	R2	R3	R4	R5	R6
	R7	R8	R9	SAVE1	SILI	TBLEND	TEXT						
DMSDBG	ABNPSW	ABNREGS	ABWSECT	ADMSCRD	ADMSERL	AIOSECT	AKILLEX	AOPSECT	ARGMAX	ARGS	ARGSAV	ARGSCT	BALRSAVE

```
MODULE       EXTERNAL REFERENCES (LABELS AND MODULES)

             BEGAT    BITS     BRKPNTBL CAW      CONHCT   CONHXT   CONWR    CONWRL   COUNT    CSW      CURRSAVE DBGABN   DBGEXEC
             DBGEXINT DBGFLAGS DBGOUT   DBGPGMCK DBGRECUR DBGSAV1  DBGSAV2  DBGSECT  DBGSET   DBGSWTCH DEC      DECDEC   DMPTITLE
             DMSABNRT DMSABW   DMSCWR   DMSCWT   DMSDBD   DMSERR   DMSIOWR  DMSITP   DUMPLIST EXAMLC   EXAMLG   EXTOPSW  FIRSTDMP
             FPRLOG   F0       F15      F6       GPRLOG   HEX      HEXHEX   IC       INPUT    INPUTSIZ INPUT1   IOOPSW   IPLPSW
             JFLAGS   LASTDMP  LINE     LOWSAVE  MVCNT    MVCNT1   MVCNT2   NUCON    OFF      OPSECT   ORG      OUTPT1   PGMOPSW
             PRFPOFF  PROTFLAG RETSAV   RSTNPSW  R0       R1       R10      R13      R14      R15      R2       R3       R4
             R5       R6       R7       R8       R9       SAVE1    SAVE2    SCAW     SILI     SSAVE    STOPAT   SYMTABLE SYMTBG
             TBLEND   TEXT     TPFUSR   TSYM     TYPFLAG  USERKEY  VMSIZE   WAITLIST WAITRD   WAITSAVE WTRDCNT  XPSW

DMSDIO       ADIOSECT ADMSFREB ADTADD   ADTDTA   ADTFLG1  ADTFRO   ADTFRW   ADTSECT  AFVS     AKILLEX  ANUCEND  BALR     CAW
             CCWX     CCW1     CCW1A    CCW2     CODE203  CSW      DEVTYP   DIAGNUM  DIAGRET  DIOBIT   DIOFLAG  DIOFREE  DIOSECT
             DOUBLE   DTAD     DTADT    ERRCODE  FREER0   FVSECT   INHIBIT  IOCOMM   IOOLD    IOOPSW   KXFLAG   KXWANT   LASTCYL
             LASTHED  LASTREC  LOC      NUCON    NUM      PLIST    QQDSK1   QQDSK2   QQTRK    READ     RETREG   RWCCW    R0
             R1       R10      R11      R12      R13      R14      R15      R2       R4       R5       R6       R7       R8
             R9       SAVEADT  SECTNUM  SEEKADR  SENCCW   SENSB    TEXT     TOOBIG   TYPE     TYP2314  TYP3330  TYP3350  UFDBUSY
             VCADTLKP WRITE    WRTKF    XRSAVE

DMSDLB       ADMSFREB ADTFDOS  ADTFLG2  ADTFROS  ADTSECT  ASYSREF  BALR     BGCOM    CMSOP    CODE203  CONREAD  CURRSAVE DOSBUFSP
             DOSCBID  DOSCMS   DOSDD    DOSDDCAT DOSDEV   DOSDOS   DOSDSK   DOSDSMD  DOSDSNAM DOSDSTYP DOSDUM   DOSEND   DOSENSIZ
             DOSEXTNO DOSEXTTB DOSFIRST DOSFLAGS DOSINIT  DOSJCAT  DOSMODE  DOSNEXT  DOSNUM   DOSOS    DOSOSDSN DOSOSFST DOSPERM
             DOSSECT  DOSSVC   DOSSYS   DOSTYPE  DOSUCAT  DOSUCNAM DOSVOLNC DOSVOLTB DOSXXX   DOSYSXXX DOUBLE   EDIT     EGPR0
             FILE     LOC      LUBPT    NICLPT   NUCON    NUM      PUBPT    READ     RESET    R0       R1       R10      R11
             R12      R13      R14      R15      R2       R3       R4       R5       R6       R7       R8       R9       SAVEXT
             SSAVE    SYSCODE  SYSTEM   TEXT     VCADTLKP VIRTUAL  VSAMFLG1 VSAMSERV VSJOECAT

DMSDLK       AADTLKP  AADTLKW  ABORT    ADTFLG1  ADTFRW   ADTM     ADTSECT  AERASE   AFINIS   ARDBUF   ASTATE   AWRBUF   BGCOM
             BLANKS   BUFFER   COMNAME  CSW      C0       DATE     DEC      DOSDD    DOSDEV   DOSDSK   DOSFIRST DOSFLAGS DOSMODE
             DOSOP    DOSOSFST DOSSECT  DOSSVC   ERROR    ESD1ST   FREELCWE FSCBBUFF FSCBD    FSCBFM   FSCBFN   FSCBFV   FSCBITNO
             FSTFB    FSTFRW   FSTFRWX  FSTFV    FSTIC    FSTIL    FSTM     FSTSECT  F1       F2       F3       F4       F5
             F6       HEX      JOBDATE  LABLEN   LUB      LUBPR    LUBRES   LUBRLB   LUB014   NOAUTO   NOMAP    NUCON    NUM
             OSFST    OSFSTDSK OSFSTXTN OUTPUT   PACK     PLIST    PO       PUBADR   PUBCUU   PUBDEVT  PUBPT    RA       READLST
             RESET    R0       R1       R2       R3       R4       R5       R6       R7       R8       R9       SF       SYSLINE
             SYSUT1   TEXT     TEXTA    TYPE     WRITE

DMSDMP       ADMSFREB ASYSREF  BALR     BGCOM    CODE203  EOCADR   LOC      NUCON    NUM      PLIST    R0       R1       R12
             R2       R3       R4       R5       R6       R7       TEXT     TYPE

DMSDOS       AAMSSYS  ABGCOM   ABNBIT   ACMSRET  ADIKQLAB ADMSERL  ADMSFREB AFVS     ALTASAVE ANCHENDA ANCHENTP ANCHINST ANCHLDPT
             ANCHLENG ANCHPHLN ANCHPHNM ANCHSECT ANCHSTSW AOSRET   APPSAVE  ARFLG    ARURTBL  ASYSCOM  ASYSNAMS ASYSREF  AVSAMSYS
             AVSREOJ  BALR     BGCOM    CALLER   CLKVALMD CMSVSAM  CODE203  COMNAME  CURRSAVE DACTIVE  DATIPCMS DIRC     DIRLL
             DIRN     DIRNAME  DIRTT    DMSFCH   DMSXCP   DOSFLAGS DOSRC    DOSTRANS DOSVSAM  EGPR0    EGPR1    EGPR14   EGPR15
             EGPR9    FCHLENG  FCHTAB   FREELOWE FVSECT   HEX      IJBABTAB IJBCCWT  IJBFTTAB INTINFO  JCSW2    JCSW4    JOBDATE
             LOC      LTK      MAINHIGH MAINLIST MAINSTRT NOTEXT   NUCON    NUCRSV3  NUM      OLDPSW   OSTEMP   PCPTR    PIBADR
             PIBFLG   PIBPT    PIBSAVE  PIB2PTR  PIK      PNOTFND  PPBEG    PPEND    R0       R1       R10      R11      R12
             R13      R14      R15      R2       R3       R4       R5       R6       R7       R8       R9       SSAVE    SVC12SAV
             SVEARA   SVEPSW   SVEPSW2  SVER0F   SVER00   SVER09   SYSCOM   SYSNAMES SYSNEND  TEXT     TEXTA    TPFSVO   TYPFLAG
```

```
MODULE         EXTERNAL REFERENCES (LABELS AND MODULES)

          TYP3330   TYP3340   TYP3350   UFDBUSY   VIPINIT   VMSIZE    VSAMFLG1  VSAMRUN   VSAMSERV  WAIT

DMSDSK    ABATABND  ADISK     ADTFTYP   ADTID     ADTSECT   AERASE    AFINIS    AFVS      AKILLEX   ARDBUF    ASTATE    ATYPSRCH  AUPDISK
          AWRBUF    BATDCMS   BATFLAGS  BATFLAG2  BATRUN    BLANKS    BUFFER    CCUNT     DEC       ERROR     FILE      FNAME     FSTDBC
          FSTFV     FSTIC     FSTIL     FSTM      FSTN      FSTSECT   FSTT      FVSECT    FVSFSTM   F65535    F800      HOLD      IADT
          KXFLAG    KXWANT    NUCON     NUM       READ      R0        R1        R13       R14       R15       R2        R3        R4
          R5        R6        R7        R8        R9        STATER1   TEXT      TYPE      UFDBUSY   UPBIT     VCFSTLKP  WRBIT

DMSDSL    ADTFLG1   ADTFRW    ADTM      ADTSECT   AERASE    ASTATE    BUFFER    DA        DIRNAME   DIRR      DIRTT     DOSFLAGS  DOSSVC
          ERROR     FCBIOSW2  FCBITEM   FCBMVPDS  FCBSECT   FILE      FSTL      FSTSECT   FXD       INPUT     NUCON     NUM       OUTPUT
          PO        PS        READ      RESET     R0        R1        R10       R12       R14       R15       R2        R3        R4
          R5        R8        SAVE1     SF        TEXT      VCADTLKP  WRITE

DMSDSV    BGCOM     BLANKS    BLANK2    COMNAME   DEC       DOSDD     DOSFIRST  DOSFLAGS  DOSMODE   DOSSECT   ERROR     FREELOWE  F1
          HEX       INPUT     LUB       LUBCLB    LUBP      LUBPR     LUBRES    LUBRLB    LUBSLB    LUB014    NUCON     NUM       PLIST
          PUBADR    PUBCUU    PUBPT     READ      RESET     R0        R1        R10       R11       R12       R13       R14       R15
          R2        R3        R4        R5        R6        R7        R8        R9        SAVER0    SEEK      TEXT      TIC       TYPE
          VMCOMP    VMDISP    VMDISP1

DMSEDC    DUALNOS   EDCB      R0        R1        R10       R13       R14       R15       R2        R3        R4        R5        R6
          R7        R8        R9        SAVEAR

DMSEDI    ADEVTAB   ADMSERL   AERASE    AEXTEND   AFINIS    AFSTFNRD  AINCORE   ALCHAR1   ALCHAR2   ALTLIST   ARDBUF    AREA      ASTATE
          ATTN      ATTNLEN   AUTOCNT   AUTOCURR  AUTOREG   AWRBUF    BLOC      BYTE      CARDINCR  CARDNO    CASEREAD  CASESW    CHGTRUNC
          CHNGCNT   CHNGFLAG  CHNGMSG   CHNGNUM   CMODE     CONSOLE   CORITEM   COUNT     CRBIT     DEC       DECIMAL   DEVTAB    DITCNT
          DMSSCR    DOSFLAGS  DOSSVC    EDCB      EDCT      EDIT      EDLIN     EDRET     ENDBLOC   ENDTABS   ERROR     FILE      FILEMS
          FLAG      FLAG2     FMODE     FNAME     FPTR      FREELEN   FSIZE     FTYPE     FV        GETFLAG   HALF      HEX       INCRNO
          INPUT     INVLD     IOID      IOLIST    IOMODE    ITEM      JAR       LINE      LINENO    LMCURR    LMINCR    LMSTART   MACRO
          MISFLAGS  MSGFLAGS  NEWMODE   NEWNAME   NEWTYPE   NOTYPING  NUCON     NUM       OFF       ON        PACK      PADBUF    PADCHAR
          PLIST     PTR1      PTR2      PTR3      RANGE     REGSAV    REGSAVX   RELPAGES  REPCNT    RESET     RPLIST    R0        R1
          R10       R13       R14       R15       R2        R3        R4        R5        R6        R7        R8        R9        SAVCNT
          SAVCWD    SAVE      SCRFLGS   SCRFLG2   SEQNAME   SERSAV    SERTSEQ   SERTSW    SIGNAL    SPARES    STACKAT   STACKATL  STRTNO
          TABLIN    TABS      TEMPTAB   TEXT      TIN       TOUT      TRNCNUM   TRUNCOL   TVERCOL1  TVERCOL2  TWITCH    TYPE      TYPFLG
          UTILFLAG  VERCOL1   VERCOL2   VERLEN    XAREA     XXXCWD    XYCNT     XYFLAG    YAREA     ZONE1     ZONE2

DMSEDX    ACMSSEG   ADEVTAB   ADMSFREB  ADTM      ADTSECT   AEDLIN    AEXTEND   AFINIS    AFLAGLOC  AFSTFNRD  ALINELOC  ALTMODE   ANUMLOC
          ARDBUF    ASTATE    ASTATEW   ASYSNAMS  BALR      BLANK1    BLANK2    BLANK3    BLOC      BUFFER    CARDINCR  CASESW    CHNGMSG
          CLASTERM  CMDBLOK   CMSSEG    CODE203   CONSOLE   CORITEM   DCSSAVAL  DCSSFLAG  DCSSLDED  DEC       DEVTAB    DOSFLAGS  DOSSVC
          EDCB      EDCBEND   EDCBLTH   EDLIN     EDRET     EDWORK    ENDBLOC   ENDTABS   ERROR     FILE      FLAG      FLAGLOC   FLAG2
          FMODE     FNAME     FREELEN   FSTD      FSTFINRD  FSTFMODE  FSTRECCT  FSTRECFM  FTYPE     FV        INVLDHDR  IOAD      IOID
          IOLIST    IOMODE    ITEM      JAR       LINE      LINELOC   LMSTART   LOC       LOCCNT    MAINAD    NUCON     NUM       NUMLOC
          ON        PADBUF    PADCHAR   PLIST     PTR1      PTR2      PTR3      RECS      REPCNT    R0        R1        R10       R12
          R13       R14       R15       R2        R3        R4        R5        R6        R7        R8        R9        SCRBUFAD  SEQNAME
          SPARES    SUBACT    SUBFLAG   SUBREJ    SYSNAMES  SYSNEND   TABS      TEXT      TIN       TRUNCOL   TWITCH    TYPE      TYPSCR
          TYP3277   TYP3278   VCFSTLKP  VERCOL1   VERCOL2   VERLEN    VIRTUAL   ZONE1     ZONE2
```

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSERR	ABATABND	AUSERRST	BATFLAGS	BATFLAG2	BATRUN	BATSYSAB	CALLEE	CAW	CONCCWS	CURRSAVE	DMSCWR	DMSCWT	DMSERT
	ERBL	ERDSECT	ERF1BF	ERF1HD	ERF1SBN	ERF1SB1	ERF1TX	ERF2CM	ERF2DI	ERF2DT	ERF2PR	ERF2SI	ERLET
	ERMESS	ERNUM	ERPAS13	ERPBFA	ERPCS	ERPF1	ERPF2	ERPHDR	ERPLET	ERPNUM	ERPSBA	ERPTXA	ERSAVE
	ERSBD	ERSBF	ERSBL	ERSECT	ERSFA	ERSFL	ERSFLST	ERSSZ	ERTEXT	ERTPL	ERTPLA	ERTPLL	ERTSIZE
	ERT1	ERT2	NUCON	OLDPSW	R0	R1	R10	R12	R13	R14	R15	R2	R3
	R4	R5	R6	R7	R8	R9	SM	SSAVE					
DMSERS	AACTFRET	AACTLKP	AACTNXT	ADMSERL	ADMSFREB	ADTADD	ADTCFST	ADTCHBA	ADTFLG1	ADTFRO	ADTFRW	ADTFSTC	ADTHBCT
	ADTLFST	ADTLHBA	ADTM	ADTRES	ADTSECT	AFTADT	AFTDBC	AFTFCL	AFTFLG	AFTPFST	AFTSECT	AFVS	AKILLEX
	AQQTRKX	ARDTK	ASTATEW	ATFINIS	ATRKLKPX	AUPDISK	BALR	CODE203	DMSERR	DMSLAD	DMSLADW	DMSLFSW	DSKADR
	DSKLOC	DSKLST	ERBIT	ERRCOD1	ERRMSG	ERSFLAG	FSTBKWD	FSTDBC	FSTFCL	FSTFWDP	FSTM	FSTN	FSTSECT
	FSTT	FVSECT	FVSERAS0	FVSERAS1	FVSERAS2	KXFLAG	KXWANT	LOC	NUCON	NUM	ON	REGSAV1	R0
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	SIGNAL	STATEFST	STATER1	TEXTA	TYPE	UFDBUSY					
DMSEXC	ACMSSEG	ADMSFREB	ADTM	ADTSECT	AEXEC	AFINIS	AFVS	AOPSECT	ASYSNAMS	BALR	CMSSEG	CODE203	DCSSAVAL
	DCSSFLAG	DCSSLDED	DMSLFS	EXADD	EXECFLAG	EXECRUN	EXLEVEL	EXNUM	FFD	FILEBUFF	FILEBYTE	FILEMODE	FSTD
	FSTLRECL	LOC	MISFLAGS	NEGITS	NOSYS	NUCON	NUM	OPSECT	PLIST	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SYSNAMES
	SYSNEND	TEXT	TYPE										
DMSEXT	ADMSFREB	ADTFDOS	ADTFLG2	ADTFMFD	ADTFROS	ADTM	ADTSECT	AFINIS	AGETCLK	AOPSECT	APOINT	ARDBUF	ASCANO
	ASTATE	BALR	BLANKS	BUFFER	BUFSIZE	CODE203	CONDFLG	CURRDATE	CURRTIME	DOSDSK	DOSFLAGS	DOSMODE	DOSSVC
	DSKLIN	ENDFREE	ERR$202	ERRMSG	EXADD	EXLEVEL	FLAG	FLAG1	FMODE	FNAME	FREENEXT	FSIZE	FSTFINRD
	F1	LABLEN	LASTCMND	LASTEXEC	LINKLEN	LOC	MSGFLAGS	NEED	NOTYPING	NUCON	OFF	ON	OPSECT
	OSRESET	OSSFLAGS	PREVCMND	PREVEXEC	READCNT	R0	R1	R10	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SKIP	SUBFLAG	SVC$202	TIMBUF	TYPLIN	TYPLIST	UNPACK	VCADTLKP
	VCADTLKW	VIPINIT	VSAMFLG1										
DMSFCH	ADMSERL	ADMSFREB	ANCHSIZ	ASTATE	ASYSREF	AUSRAREA	BALR	BGCOM	BUSOUT	CC	CMDREJ	CODE203	COMNAME
	CSW	DACTIVE	DATACHK	DIRAAA	DIRC	DIREEE	DIRLL	DIRN	DIRNAME	DIRPPP	DIRRR	DIRTT	DIRTTR
	DOSFIRST	DOSFLAGS	DOSKPART	DOSLIBL	DOSREAD	DOSSVC	DOSTRANS	DOSVSAM	EQCHK	ERRMSG	ERROR	FCBDD	FCBDEV
	FCBDSK	FCBDSNAM	FCBINIT	FCBOP	FCBOSFST	FCBSECT	FREELCWE	FRERESPG	HIPHAS	HIPROG	IHADEB	INPUT	INTREQ
	LOC	LUBPT	MAINHIGH	MAINLIST	MAINSTRT	NOTEXT	NUCON	OSFST	OSFSTDSK	OSFSTXTN	PCTVSAM	PNOTFND	PO
	PPEND	PS	PUBPT	READ	READCNT	RELPHSE	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SEARCH	SEEK	SF	TEXT
	TIC	VIRTUAL	VSAMFLG1	VSAMRUN	VSAMSERV	VSMINSTL							
DMSFET	ABGCOM	ADMSERL	ADMSFREB	ALDRTBLS	ASYSCOM	AUSRAREA	BALR	BGCOM	CODE203	COMNAME	DACTIVE	DIRN	DIRNAME
	DMSERR	DOSCOMP	DOSFLAGS	DOSMODE	DOSRC	DOSSVC	FCHAPHNM	FCHLENG	FCHOPT	FCHTAB	HIPHAS	IJBFTTAB	LASTLOC
	LOC	LOCCNT	NOTEXT	NUCON	NUM	PNOTFND	R0	R1	R12	R14	R15	R2	R3
	R4	R5	R6	R7	START	STRTADDR	SYSCOM	TBENT	TEXT	VSMINSTL			
DMSFLD	ABATABND	ASTATE	BATDCMS	BATFLAGS	BATFLAG2	BATRUN	CONREAD	CURRSAVE	DUMMY	EGPRO	FCBBLKSZ	FCBCASE	FCBCATML
	FCBCON	FCBDD	FCBDEV	FCBDOSL	FCBDSK	FCBDSMD	FCBDSNAM	FCBDSORG	FCBDSTYP	FCBDUM	FCBEND	FCBENSIZ	FCBFIRST
	FCBINIT	FCBIOSW	FCBLRECL	FCBMEMBR	FCBMODE	FCBNEXT	FCBNUM	FCBOSDSN	FCBPCH	FCBPROC	FCBPTR	FCBRDR	FCBRECFM

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	FCBSECT	FCBTAP	FCBTAPID	FCBXTENT	FILE	FLAG1	FLAG2	FLAG3	JFCBIND2	JFCBUFNO	JFCKEYLE	JFCLIMCT	JFCOPTCD
	LOC	NUCON	NUM	PACK	RESET	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	SSAVE	TABEND	TEXT	TYPE	
DMSFNC	ATTN	CONREAD	DMSABNSV	DMSBWR	DMSCAT	DMSCIOSI	DMSCITDB	DMSCPF	DMSCRD	DMSCWR	DMSCWT	DMSDBG	DMSERR
	DMSEXC	DMSFET	DMSFREB	DMSFREES	DMSFREEX	DMSFRES	DMSFRETS	DMSFRETX	DMSITET	DMSITSK	DMSITSXS	DMSLADAD	DMSLDRA
	DMSLOA	DMSMOD	DMSPIO	DMSPIOCC	DMSPIOSI	DMSSTGAT	DMSSTGCL	DMSSTGSB	DMSSTGSV	DMSVSR	FINIS	LOC	NUM
	R0	START	TRAP	TYPLIN	WAIT	WAITRD							
DMSFNS	AACTFRET	AACTLKP	ADIOSECT	ADMSERL	ADMSFREB	ADTADD	ADTDTA	ADTFLG3	ADTFTYP	ADTFUPD1	ADTFXCHN	ADTNACW	ADTRES
	ADTSECT	ADTXNREC	AERASE	AFTADT	AFTCLA	AFTCLB	AFTCLD	AFTCLDX	AFTCLN	AFTCLX	AFTDBA	AFTDBD	AFTFBA
	AFTFCL	AFTFCLA	AFTFCLX	AFTFLG	AFTFLG2	AFTFST	AFTFULD	AFTM	AFTN	AFTNEW	AFTPFST	AFTRD	AFTSECT
	AFTUSED	AFTWP	AFTWRT	AFVS	AKILLEX	AQQTRKX	ARDTK	ATRKLKPX	ATYPSRCH	AUPDISK	AWRTK	BALR	BALRSAVE
	CLKVALMD	CODE203	DATIPCMS	DEVTYP	DIOCSW	DIOSECT	DISK$SEG	DMSERR	DMSLFSW	DSKLOC	DSKLST	FINISLST	FNBIT
	FSTD	FSTFB	FSTIC	FSTM	FSTN	FSTRP	FSTSECT	FSTT	FSTWP	FSTYR	FVSECT	HEX	KXFLAG
	KXWANT	LOC	NUCON	NUM	QQDSK1	REGSAV3	RWFSTRG	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R5	R6	R7	R8	R9	SECTNUM	SEEKADR	SENSB	STATEFST	SUBFLAG
	SUBINIT	TEXT	TYPE	UFDBUSY	VIRTUAL								
DMSFOR	ADEVTAB	ADMSFREB	ADTCYL	ADTDTA	ADTFALUF	ADTFDA	ADTFDCS	ADTFFSTF	ADTFLG1	ADTFLG2	ADTFQQF	ADTFRO	ADTFROS
	ADTFRW	ADTHBCT	ADTID	ADTLAST	ADTLEFT	ADTLHBA	ADTM	ADTMSK	ADTNUM	ADTPQM1	ADTPQM2	ADTPQM3	ADTQQM
	ADTRES	ADTSECT	ADTUSED	ADT1ST	AFINIS	ARDTK	AUPDISK	AWRTK	BALR	CC	CODE203	DTAD	FLAG
	LOC	NUCON	NUM	QQDSK1	RESET	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	SECTNUM	SEEKADR	SENSB	SENSE	SILI
	START	TEXT	TYPE	VCADTLKP	WAITRD								
DMSFRE	ABNPSW	ABNREGS	ABWSECT	ACALL	ADMSERL	AFREETAB	ASSTAT	ASVCSECT	AUSRAREA	BALR	BATFLAGS	BATLOAD	BLOCKLEN
	CALLER	CL	CODE203	CURRSAVE	DMSABNGO	DMSABW	DMSERR	DMSFRT	DMSNUCU	FINIS	FLAGS	FLCLN	FLHC
	FLNU	FLPA	FRDSECT	FREEFLG1	FREEFLG2	FREEHN	FREEHU	FREELN	FREELOWE	FREELOW1	FREELU	FREESAVE	FRF1B
	FRF1C	FRF1E	FRF1H	FRF1L	FRF1M	FRF1N	FRF1V	FRF2CKE	FRF2CKT	FRF2CKX	FRF2CL	FRF2NOI	FRF2SVP
	LOC	LOCCNT	MAINHIGH	MAX	MAXCODE	NUCCODE	NUCKEY	NUCON	NUM	POINTER	PRFPOFF	PROTFLAG	R0
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	SIZE	SKEY	SSAVE	SVCAB	SVCSECT	SYSCODE	TCODE	TRNCODE	TYPE	USARCODE	USERCODE
	USERKEY	VMSIZE											
DMSGIO	ADEVTAB	CMDBLOK	CSW	EDCB	LOC	NUCON	R0	R1	R10	R13	R14	R15	R2
	R3	R4	R5	R9									
DMSGLB	AFINIS	ARDBUF	ASTATE	BUFFER	DOSLBSV	DOSLIBL	FILE	LOC	MACLESV	MACLIBL	NUCON	R0	R1
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R7	R8	TEXT	TOTLIBS
	TXLIBSV	TXTDIRC	TXTLIBS										
DMSGND	ALDRTBLS	ASTATE	DIRNAME	FILE	FSTD	FSTDATEW	FSTSECT	NUCON	NUM	R0	R1	R11	R12
	R14	R15	R2	R3	R4	R5	R6	R9	STATEFST	TBENT	TEXT		
DMSGRN	BLANKS	ERROR	EXECRUN	FFS	FSCBFM	FSCBFN	FSCBFT	INPUT	OUTPUT	PARMLIST	PROCERR	RUN	R0

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)											
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	SAVE	START	TEXT	TEXTA							
DMSHDI	ADMSFREB	AIOSECT	ANUCEND	AUSRILST	AUSRITBL	BALR	CODE203	DOSFLAGS	DOSSVC	ERRCODE	F256	IONTABL	IOSECT
	LOC	NUCON	R0	R1	R10	R12	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R8	R9	VMSIZE								
DMSHDS	ADMSFREB	ANUCEND	ASVCSECT	BALR	CODE203	DOSFLAGS	DOSSVC	ERRCODE	F256	JFIRST	JLAST	JNUMB	LOC
	NUCON	R0	R1	R10	R12	R13	R14	R15	R2	R3	R4	R5	R6
	R7	R8	R9	SVCSECT	VMSIZE								
DMSIFC	AADTLKW	ADTM	ADTSECT	BUFFER	COMPSWT	CURRSAVE	DMSREA	DOSFLAGS	DOSSAVE	DOSSVC	EDIT	EGPR15	ERROR
	FILE	FSCBBUFF	FSCBD	FSCBFM	FSCBFN	FSCBFV	FSTFV	FSTIL	FSTM	FSTSECT	IOBECB	LOADLIST	LOC
	NUCON	NUM	OLDPSW	OSSFLAGS	RESET	R0	R1	R12	R13	R14	R15	R2	R3
	R4	R5	R6	R7	R8	R9	SAVER0	SAVER1	SAVER14	SAVER15	SAVE2	SSAVE	TEXT
	TXTDIRC	TXTLIBS	TYPE										
DMSINA	AUSABRV	BALRSAVE	EDIT	NOABBREV	NOSTDSYN	NUCON	NUM	OPTFLAGS	R0	R1	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	TYPE					
DMSINI	ADEVTAB	BLANKS	CAW	CC	CE	CHANO	CLASDASD	CLASTERM	CONSOLE	CSW	DE	DEVTAB	DMSDBGP
	DMSINS	DMSINSE	DMSITS1	EXTNPSW	INSTALID	IONPSW	IOOPSW	IPLCCW1	IPLPSW	MCKM	MCKNPSW	NOP	NUCON
	RDCONS	RDDATA	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SDISK	SEARCH	SEEK	SETSEC	SILI	SYSADDR	SYSTEMID	TIC
	TYP2305	TYP2311	TYP2314	TYP3210	TYP3330	TYP3340	TYP3350	WAIT	WRDATA	WRITE	WRITE1	YDISK	ZEROES
DMSINM	ASUBSECT	BALRSAVE	CURRCPUT	CURRDATE	CURRVIRT	NUCON	R0	R1	R10	R14	R15	R2	R3
	R4	R5	R8	SUBSECT	TIMBUF								
DMSINS	ABGCOM	ACMSCVT	ACMSSEG	ADISK	ADMSFREB	ADTFDA	ADTFFSTF	ADTFFSTV	ADTFLG1	ADTFLG3	ADTFORCE	ADTFSORT	ADTFSTC
	ADTSECT	AEXTSECT	ALDRTBLS	AOPSECT	AOSMODL	AREA	ASSTAT	ASTATE	ASTATEXT	ASYSNAMS	ASYSREF	AUSRAREA	BALR
	BATFLAGS	BATFLAG2	BATIPLSS	BATLOAD	BATRUN	BGCOM	CAW	CC	CHANO	CLKVALMD	CMNDLINE	CMNDLIST	CMSCVT
	CMSSEG	CODE203	CONRDCNT	CONRDCOD	CONREAD	CONSOLE	CURRDATE	CVTMDL	CVTMZ00	CVTNUCB	CVTOPTA	CVTSECT	DATIPCM
	DCSSAVAL	DCSSFLAG	DCSSOVLP	DCSSVTLD	DDISK	DMSDBG	DMSFRES	DMSLAD	DMSLOA	DMSSCNN	DTAD	EXTSECT	FREELOW
	FRERESPG	FVS	F0	GRAFDEV	IONPSW	IPLADDR	IPLPSW	LOADSTRT	LOC	LOCCNT	MAINHIGH	MCKM	MISFLAG
	MODFLGS	MSGFLAGS	NOVMREAD	NUCON	NUM	OPSECT	OPTFLAGS	OSMODLDW	PGMNPSW	PRFTSYS	PROTFLAG	REGSAV	RGPRS
	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6
	R7	R9	SDISK	SILI	SPECLF	SYSLOAD	SYSNAME	SYSNAMES	SYSNEND	SYSREF	SYSTEMID	TEXT	TIMCHAR
	TIMER	TIMINIT	TYPE	VMSIZE	WAIT	YDISK	YYDDD						
DMSINT	AACTLKP	ADMSFREB	AEXTSECT	AFTM	AFTN	AFTSECT	AFTWP	AFVS	AIOSECT	AOPSECT	ASCBPTR	ASUBFST	ASUBRET
	ASUBSECT	ASUBSTAT	ASVCSECT	ASYSNAMS	AUSRAREA	BALR	CMNDLINE	CMSSEG	CMSTIM	CODE203	CONRDCNT	CONRDCOD	CONREAD
	CONWRBUF	CONWRCOD	CONWRITE	DCSSFLAG	DCSSJLNS	DCSSLDED	DMSCPF	DMSDBG	DMSLFS	DMSSCNN	DMSSTGSB	DOSFLAGS	DOSMODE
	DOSSVC	ERRET	ERRNUM	EXTPSW	EXTSECT	FILENAME	FILETYPE	FINISLST	FREELCWE	FSTFINRD	FVSECT	IONTABL	IOSECT
	JNUMB	LASTCMND	LOC	LOCCNT	MISFLAGS	MSGFLAGS	NEGITS	NOABBREV	NOIMPCP	NOIMPEX	NOPAGREL	NORDYTIM	NOTYPIN
	NOVMREAD	NUCON	OPSECT	OPTFLAGS	OSRESET	OSSFLAGS	PLIST	PREVCMND	QSWITCH	REDERRID	RELPAGES	RMSGBUF	R0

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)											
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R9	SPECLF	SPIESAV	STAESAV	STARS	STATEFST	SUBACT	SUBFLAG	SUBREJ	SUBSECT	SVCSECT	SWTCHSAV	SYSNAMES
	SYSNEND	TIMCHAR	TIMER	TIMINIT	TYPE	VIPINIT	VSAMFLG1						
DMSIOW	AEXTSECT	CSW	DBGEXINT	DBGFLAGS	DEVICE	DMSDBG	EXTFLAG	EXTSECT	IONPSW	IONTABL	IOOPSW	NUCON	REALTIMR
	R0	R1	R10	R11	R14	R15	R2	R4	R5	R6	R7	R8	R9
	TIMCHAR	TIMER	TIMINIT	WAITSAVE									
DMSITE	ABATABND	ABATLIMT	ADMSFREB	AEXTSECT	ARGS	ASVCSECT	BALR	BATCPUC	BATCPUL	BATFLAGS	BATFLAG2	BATLOAD	BATLSECT
	BATRUN	BATUSEX	BATXCPU	BATXLIM	CMSTAXE	CODE203	CONHCT	CSW	DBGEXEC	DBGEXINT	DBGFLAGS	DBGOUT	DBGSECT
	DECDEC	DMSCWR	DMSDBG	DOSFLAGS	DOSSVC	EXSAVE	EXSAVE1	EXTFLAG	EXTOPSW	EXTPSW	EXTRET	EXTSECT	FVS
	FVSECT	F0	F2	F4	F6	INPUT	IONPSW	IOOPSW	IR1	LASTUSER	LINE	LOC	MVCNT1
	NUCON	NUMPNDWR	OSSFLAGS	OSWAIT	OVSTAT	PENDREAD	REALTIMR	RESET	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R7	R8	SAVEXT	SCAW	SILI	STIMEXIT	SVCSECT	TAXEADDR
	TAXEFREQ	TAXELNK	TAXESTAT	TBLEND	TIMCCW	TIMCHAR	TIMER	TIMINIT	TRAP	TSOATCNL	TSOFLAGS	TYPE	TYPLIST
	UFDBUSY	WAIT	XPSW										
DMSITI	ABNPSW	ABNREGS	ABWSECT	ADIOSECT	AFVS	AIOSECT	ATTNHIT	BALR14	CMSTAXE	CSW	DEVICE	DIOSECT	DMSABNGO
	DMSABW	FVSECT	HOLD	IONTABL	IOOLD	IOOPSW	IOPSW	IOSAVE	IOSECT	KXFLAG	KXWANT	MISFLAGS	NEXTO
	NUCON	OLDEST	QQDSK1	R0	R1	R10	R11	R12	R13	R14	R15	R3	R4
	R5	R6	R7	R8	R9	SECTNUM	SEEKADR	SENSB	TAXEADDR	TAXEFREQ	TAXEIOL	TAXELNK	TAXESTAT
	TSOATCNL	TSOFLAGS	UFDBUSY	VSTRANGE	WAIT								
DMSITP	ABNERLST	ABNPSW	ABNREGS	ABWSECT	ADMSFREB	AFVS	ALTASAVE	APGMSECT	APPSAVE	ASYSCOM	ASYSREF	AUPIE	BALR
	BGCOM	CALLEE	CODE203	CURRSAVE	DMSABNGO	DMSABW	DMSERR	DOSFLAGS	DOSMCDE	DOSSVC	FVSECT	IJBABTAB	INTINFO
	LOC	LTK	NUCON	NUM	OPSW	PCPTR	PGMNPSW	PGMOPSW	PGMSECT	PIBADR	PIBPT	PIBSAVE	PICADDR
	PIE	PIK	PSAVE	RESET	R0	R1	R10	R11	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SCBPTR	SSAVE	SVEARA	SVEPSW	SVEPSW2	SVER00
	SVER09	SYSCOM	TPFUSR	TYPE	TYPFLAG	UFDBUSY	VSAMFLG1	VSAMSERV					
DMSITS	ABNPSW	ABNREGS	ABWSECT	ACMSSEG	ADMSERL	ADMSFREB	ADMSOVS	ADOSDCSS	AERR	AFVS	AOSMODL	ASVCSECT	ASYSNAMS
	AWAIT	BALR	CALLEE	CALLER	CHKWRD1	CHKWRD2	CMSSEG	CODE	CODE203	CURRALOC	CURRSAVE	DCSSAVAL	DCSSFLAG
	DCSSLDED	DCSSVTLD	DEPTH	DMSABNGO	DMSABW	DMSCWT	DMSERR	DMSFNC	DMSFNC3	DMSMOD	DOSFLAGS	DOSSVC	DUMCOM
	EFPRS	EGPRS	EGPR0	EGPR11	EGPR15	EGPR2	ERRET	FLAGS	FVSECT	F0	F6	GPRLOG	ITSBIT
	JFIRST	JLAST	JNUMB	KEYMAX	KEYP	KEYS	KXFLAG	KXWANT	KXWSVC	LASTALOC	LASTTMOD	LENOVS	LOC
	MCKM	MISFLAGS	MODLIST	NEGITS	NRMRET	NRMSAV	NRMUSAV	NUCON	NUM	OFF	OLDPSW	ON	OVSAFT
	OVSECT	OVSON	OVSTAT	PRFPOFF	PRFTSYS	PRFUSYS	PROTFLAG	RGPRS	RGPR11	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SFLAG
	SFNUC	SFREN	SFSYS	SFTRN	SSAVE	SSAVENXT	SSAVEPRV	SSAVESZ	START	STRTADDR	SVCAB	SVCOPSW	SVCOUNT
	SVCSAVE	SVCSECT	SVCSTOP	SYSNAMES	SYSNEND	TEMP02	TEXT	TPFERT	TPFNS	TPFR01	TPFSVO	TPFUSR	TSOATCNL
	TSOFLAGS	TYPE	TYPFLAG	UFDBUSY	USAVE	USAVEPTR	USAVESZ	USERKEY					
DMSLAD	ADMSFREB	ADTFDA	ADTFFSTV	ADTFLG1	ADTFLG2	ADTFRO	ADTFRCS	ADTFRW	ADTFVS	ADTHBCT	ADTLEFT	ADTM	ADTPSTM
	ADTPTR	ADTRES	ADTSECT	AFVS	ASVCSECT	BALR	CODE203	FVSECT	IADT	LOC	NUCON	REGSAV0	R0
	R1	R10	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	SVCSECT	SVLAD	SVLADW	TYPE								

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSLAF	ADMSFREB	ADTFLG1	ADTFRW	ADTM	ADTMX	ADTSECT	AFTADT	AFTFB	AFTFLG	AFTFSF	AFTFST	AFTLD	AFTM
	AFTN	AFTPFST	AFTPTR	AFTSECT	AFTT	AFTUSED	BALR	CODE203	FSTL	FSTSECT	LOC	NUCON	R0
	R1	R11	R12	R13	R14	R15	R2	R3	R4	R5	TYPE		
DMSLBM	AADTLKP	AADTLKW	ADTFLG1	ADTFRO	ADTFRW	ADTM	ADTSECT	BUFFER	DOUBLE	ERRCODE	ERROR	FILE	FLAGS
	FREELOWE	FSTFV	FSTIC	FSTIL	FSTM	FSTSECT	INSIZE	MISFLAGS	NUCON	NUM	PLIST	PREVIOUS	RELPAGES
	RESET	R0	R1	R10	R11	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	TEXT	TEXTA	VIRTUAL								
DMSLBT	AADTLKP	AADTLKW	ADTFLG1	ADTFRO	ADTFRW	ADTSECT	ARDBUF	AWRBUF	BLANKS	BUFFER	DOUBLE	ENDFREE	FILE
	FINIS	FLAGS	FMODE	FSIZE	MISFLAGS	NOLIBE	NUCON	NUM	RADD	RELPAGES	RESET	RITEM	R0
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	SAVE	TEXT	TEXTA	TYPLIN							
DMSLDR	ACMSRET	ADMSFREB	AFINIS	ALDRTBLS	AOSMODL	APRILB	APSV	ARDBUF	ASCANN	ASTATE	AUSRAREA	BALR	BATFLAGS
	BATLOAD	BLANKS	BRAD	CALLEE	CLOSELIB	CMD	CMNDLIST	CODE203	COMMONEX	CRDPTR	CURRSAVE	C12	C7
	C9	DMSLGTA	DMSLGTB	DMSLIB	DMSLIO	DMSLSBA	DMSLSBB	DMSLSBC	DMSLSBD	DMSLSY	DMSSTGSB	DOSCOMP	DOSFLAGS
	DOSMODE	DOSRC	DOSSVC	DYLD	DYNAEND	EGPR1	ENDCDADR	ENTADR	ENTNAME	ESDIST	ESIDTB	FDISK	FINIS
	FLAGS	FLAG1	FLAG2	FLAG3	FREELOWE	FRSTSDID	FSTXTADR	FTYPE	GPRSAV	LDRADDR	LDRFLAGS	LDRRTCD	LDRST
	LOC	LOCCNT	LOCCT	LUNDEF	MAINHIGH	MEMBOUND	MODFLGS	NEED	NOAUTO	NODUP	NOINV	NOLIBE	NOREP
	NOSLCADR	NUCON	NUM	NUMBYTE	NXTSYM	OSRESET	OSSFLAGS	OUTBUF	OUTPUT	PARMLIST	PLISTSAV	PREXIST	PRFTSYS
	PRFUSYS	PRHOLD	PROTFLAG	PRVCNT	PSW	READBUF	REFCMD	REFLG1	REFLG2	REFLIB	REFUND	REG13SAV	RESET
	RETREG	RLDCONST	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SAV67	SPEC	SSAVE	START	STRTADDR	SYSLOAD	SYSUT1	TBENT
	TBLCT	TBLREF	TEMPST	TEXT	TMPLOC	TPFUSR	TXTDIRC	TYPFLAG	UNRES	USERKEY	VMSIZE		
DMSLDS	ADMSROS	ADTCYL	ADTFLG1	ADTFLG2	ADTFRO	ADTFROS	ADTFRW	ADTID	ADTM	ADTSECT	CC	CONCNT	CSW
	DOSFLAGS	DOSSVC	FCBIOSW2	FCBMEMBR	FCBMVPDS	FCBOSDSN	FCBSECT	FMODE	HALF	NUCON	NUM	ON	OSADTDSK
	OSADTVTA	OSADTVTB	PO	POU	RESET	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	TEXT	VCADTLKP	VCADTNXT		
DMSLFS	ADMSFREB	ADMSROS	ADTCHBA	ADTFDA	ADTFFSTV	ADTFLG1	ADTFLG2	ADTFLG3	ADTFRO	ADTFROS	ADTFRW	ADTFSORT	ADTFTYP
	ADTHBCT	ADTLFST	ADTLHBA	ADTM	ADTMX	ADTPSTM	ADTRES	ADTSECT	AFVS	ASVCSECT	BALR	CODE203	DISK$SEG
	DMSLAD	DMSLADN	DMSSTTR	FVSECT	NUCON	REGSAV0	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SVCSECT	SVLFS	TYPE	
DMSLGT	ADMSFREB	APSV	ARDBUF	BALR	CODE203	DMSLDRD	FILE	FMODE	FNAME	FTYPE	LDRST	LOC	NUCON
	OUTBUF	RADD	READBUF	RFIX	RITEM	RLENG	RNUM	R0	R1	R10	R12	R13	R14
	R15	R3	R4	R5	R6	R7	R8	R9	SPEC	TEXT	TXTDIRC	TXTLIBS	TYPE
DMSLIB	ADMSFREB	AFINIS	APOINT	APSV	ASTATE	BALR	CLOSELIB	CODE203	DEC	DMSLDRD	DYMBRNM	FILE	FINIS
	FLAGS	FLAG2	FMODE	FNAME	FTYPE	LDRST	LOC	NOAUTO	NOLIBE	NUCON	NUMBYTE	OSSFLAGS	OUTBUF
	RADD	READBUF	RITEM	RLENG	RNUM	R0	R1	R11	R12	R13	R14	R15	R5
	R7	SEARCH	SETLIB	SPEC	TBLCT	TBLREF	TXTDIRC	TXTLIBS	TYPE				

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSLIO	AERASE	AFINIS	ALIASENT	APSV	AWRBUF	DMSERR	DSKAD	DSKLIN	DYLD	ERROR	FILE	FLAG1	FLAG2
	FNAME	LDRADDR	LDRST	LINE1	NOERASE	NOMAP	NUCON	NUM	OSSFLAGS	OUTBUF	OUTPUT	PACK	PARMLIST
	PLISTSAV	R0	R1	R10	R11	R13	R14	R15	R2	R3	R4	TEXT	TYPE
	TYPEAD	TYPLIN	UNPACK	VIRTUAL									
DMSLKD	AADTLKW	ADTM	ADTSECT	CODE	FILE	FSTFV	FSTIL	FSTM	FSTSECT	MISFLAGS	NUCON	PROCERR	RELPAGES
	R0	R1	R10	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7
	R9	SIZE	SYSUT1	TEXT									
DMSLLU	ADTFLG1	ADTFLG3	ADTFRW	ADTFRWOS	ADTSECT	AERASE	AFINIS	ASYSREF	AWRBUF	BGCOM	BLANKS	DEVTAB	DEVTYP
	DOSFLAGS	DOSMODE	DSKLST	ERROR	FINIS	LUBPT	NICLPT	NUCON	PUBADR	PUBCUU	PUBDEVT	PUBDSKM	PUBPT
	R0	R1	R10	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7
	R8	TAPE	TEXT	VCADTLKP									
DMSLOA	ALDRTBLS	AUSRAREA	DMSLDRB	FSTXTADR	LDRADDR	LDRFLAGS	LOCCNT	MAINHIGH	NOAUTO	NOERASE	NOINV	NOLIBE	NOMAP
	NOREP	NUCON	PRHOLD	R0	R1	R12	R14	R15	R2	R6	STRTADDR	SUBACT	SUBFLAG
	SYSREF	TBENT	TEXT	TYPE	UNRES								
DMSLSB	ADMSFREB	ADTRANS	APSV	AUSRAREA	BALR	BATFLAGS	BATLOAD	BRAD	CLEAROP	CODE203	DMSLDRC	DMSLDRD	ENDCDADR
	ENTNAME	FLAGS	FLAG1	FLAG2	FREELOWE	FRSTSDID	FSTXTADR	LASTTMOD	LDRST	LOC	LOCCT	MAINHIGH	MODFLGS
	NOAUTO	NODUP	NOINV	NOLIBE	NOMAP	NOREP	NUCON	OUTBUF	RESET	RETT	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	START	STRTADDR	SYSLOAD	TMPLOC	TYPE								
DMSLST	ADTFDA	ADTFLG1	ADTFLG2	ADTFRO	ADTFROS	ADTFRW	ADTID	ADTM	ADTSECT	AERASE	BRAD	COMNAME	DATE
	DEC	FLAG	FLAGS	FMODE	FNAME	FTYPE	NUCON	NUM	RETREG	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SEARCH
	TEXTA	TYPE	VCADTLKP	VCADTNXT									
DMSLSY	DSYM	GET1	JSYM	NUCON	NXTSYM	R0	R1	R14	R15				
DMSMDP	ALDRTBLS	MDPCALL	MODFLGS	NUCON	PLIST	R0	R1	R14	R15	R2	R3	R4	TBENT
	TEXT												
DMSMOD	ACTIVE	ADMSERL	ADMSFREB	ADTRANS	AERASE	AFINIS	AFVS	ALDRTBLS	ARDBUF	ARDTK	ASTATE	ASTATEW	AUSRAREA
	AWRBUF	BALR	CODE203	DMSERR	DMSSTGSB	DOSFLAGS	DOSMODE	DOSSVC	DSKLIN	DSKLOC	DSKLST	ERROR	FILE
	FREELOWE	FRSTLOC	FVSECT	FVSFSTAD	FVSFSTCL	FVSFSTFV	FVSFSTIC	FVSFSTIL	F65535	LASTLMOD	LASTTMOD	LDRFLAGS	LOC
	LOCCNT	MDPCALL	MODFLGS	MODGNALL	MODGNDOS	NOERASE	NOMAPFLG	NUCON	NUM	PRFTSYS	PRFUSYS	PROTFLAG	REGSAV3
	RWCNT	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R8	R9	SEARCH	STRTADDR	SUBFLAG	SYSTEM	TBENT	TEXT	TEXTA		
DMSMVE	AADTLKP	ADTFLG1	ADTFRO	ADTFRW	ADTSECT	BATFLAGS	BATMOVE	CCNFLAG	DA	DDNAM	DOSFLAGS	DOSSVC	EXSAVE
	FCBBLKSZ	FCBDD	FCBDEV	FCBDSK	FCBDSMD	FCBDSNAM	FCBINIT	FCBIOSW2	FCBITEM	FCBLRECL	FCBMMV	FCBMVFIL	FCBMVPDS
	FCBOP	FCBOPCB	FCBOSFST	FCBRECFM	FCBSECT	FCBTAP	FCBTAPID	FLAG	FSTFV	FSTIL	FSTSECT	IHADEB	INPUT
	NUCON	NUM	OSFST	OSFSTBLK	OSFSTLRL	OSFSTRFM	OUTPUT	PLIST	PS	RESET	R0	R1	R10
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	TEXT

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSNCP	BYTE	CCPADDR	CCPARM	CCPCAONE	CCPENTRY	CCPHBFNO	CCPHBFSZ	CCPMAXID	CCPNAME	CCPPAD0	CCPPAD1	CCPPSIZE	CCPRESID
	CCPRSTAT	CCPRSTEP	CCPRSTYP	CCPSIZE	CCPSTOR	CCPTEP	CCPTEP4	CCPTNCP	CCPTPEP	CCPTYPE	CCPTYPE1	CCPTYPE2	CCPVPAD0
	CCPVPAD1	CODE	DA	ERROR	FILE	FILEMODE	FILENAME	FREELOWE	FSTD	FSTFMODE	INPUT	NICCIBM	NICCTLR
	NICDISA	NICEPMD	NICGRAF	NICLBSC	NICLGRP	NICLINE	NICMLTP	NICRCPU	NICRSPL	NICSDLC	NICSWCH	NICSWEP	NICTELE
	NICTERM	NUCON	NUM	PO	QS	RDBUFLN	RDBUFNO	READBUF	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R8	R9	SAVE	SF	TEXT
	VIRTUAL												
DMSNUC	ADISK	ADTB	ADTC	ADTD	ADTE	ADTF	ADTG	ADTS	ADTY	ADTZ	ARGS	BDISK	CDISK
	CONHCT	DBGOUT	DDISK	DECDEC	DMSDBG	DMSINALT	DMSINA1S	EDISK	FDISK	GDISK	INPUT	LINE	MVCNT1
	QQDSK1	SDISK	SECTNUM	SEEKADR	SENSB	SILI	TBLEND	TIMCHAR	YDISK	ZDISK			
DMSOLD	ADMSFREB	ADMSLIO	ADTRANS	AERASE	AFINIS	ALDRTBLS	APRILB	APSV	ARDBUF	ASCANN	ASTATE	AUSRAREA	AWRBUF
	BALR	BATFLAGS	BATLOAD	BLANKS	BRAD	CLOSELIB	CMD	CMNDLIST	CODE203	COMMONEX	CRDPTR	DMSLGTA	DMSLGTB
	DMSLIB	DMSLSBA	DMSLSBB	DMSLSBC	DMSLSBD	DMSLSY	DYLD	DYNAEND	ENDCLADR	ENTADR	ENTNAME	ESD1ST	ESIDTB
	FDISK	FINIS	FLAGS	FLAG1	FLAG2	FLAG3	FREELOWE	FSTXTADR	FTYPE	GPRSAV	LDRADDR	LDRFLAGS	LDRRTCD
	LDRST	LOC	LOCCNT	LOCCT	LUNDEF	MEMBOUND	MODFLGS	NEED	NOAUTO	NODUP	NOINV	NOLIBE	NOREP
	NOSLCADR	NUCON	NUM	NUMBYTE	NXTSYM	OSRESET	OSSFLAGS	OUTBUF	OUTPUT	PARMLIST	PLISTSAV	PREXIST	PRVCNT
	READBUF	REFCMD	REFLG1	REFLG2	REFLIB	REFUND	REG13SAV	RESET	RETREG	RLDCONST	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SAV67	SPEC	STRTADDR	SYSLOAD	SYSUT1	TBENT	TBLCT	TBLREF	TEMPST	TMPLOC	TXTDIRC	UNRES	WORKFILE
DMSOPL	ACTIVE	ADMSFREB	ASYSREF	BALR	BGCOM	BUFFER	CODE203	DOSDD	DOSFIRST	DOSNEXT	DOSSECT	DOSSYS	LOC
	LUBPT	NUCON	NUM	R0	R1	R12	R15	R2	R3	R4	R5	R6	R7
	R8	R9	SEEK	TEXT	TIC	TYPE							
DMSOPT	ABGCOM	BGCOM	DOSFLAGS	DOSMODE	JCSW3	JCSW4	NUCON	RESET	R0	R1	R10	R11	R12
	R14	R15	R2	SOB1	TEXT								
DMSOR1	ADMSFREB	BALR	CODE203	INPUT	LOC	NUCON	NUM	ON	OUTPUT	R0	R1	R12	R15
	R2	R5	R6	TEXT	TRUN	TYPE	VAR	ZEROES					
DMSOR2	R1	R12											
DMSOR3	CCW2	CONSOLE	F7	R1	R12	R14							
DMSOVR	ADMSOVS	ASVCSECT	BUFFER	DEC	DMSOVS	ERROR	LENOVS	LOC	NUCON	NUM	OFF	ON	OVAPF
	OVBPF	OVF1F	OVF1FS	OVF1GA	OVF1GB	OVF1GS	OVF1ON	OVF1PA	OVF2CM	OVF2NR	OVF2OS	OVF2WA	OVSECT
	OVSHO	OVSON	OVSSO	OVSTAT	R0	R1	R12	R14	R15	R3	R4	R5	R6
	R7	R8	SVCSECT	TEXT	TYPE								
DMSOVS	ASVCSECT	BUFFA	CALLEE	CALLER	CURRSAVE	DEPTH	EFPRS	EGPRS	EGPR0	EGPR15	FLAGS	NUCON	OLDPSW
	ON	OUTPUT	OVAPF	OVBPF	OVF1F	OVF1FS	OVF1GA	OVF1GB	OVF1GS	OVF1ON	OVF1PA	OVF2CM	OVF2NR
	OVF2OS	OVF2ST	OVSAFT	OVSHO	OVSON	OVSSO	OVSTAT	RFPRS	RGPRS	RGPR8	R0	R1	R12
	R13	R14	R15	R3	R4	R5	R6	R7	R8	SSAVE	START	SVCOUNT	SVCSECT

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)											
	TEXT	TEXTA	TPFSVO	TYPE	TYPFLAG	VMSIZE	XCOUNT	XGPRO	XGPR1	XGPR15		
DMSPIO	ABATABND	ABATLIMT	ADMSERL	BATFLAGS	BATLSECT	BATNOEX	BATPRTC	BATPRTL	BATRUN	BATXLIM	BATXPRT	BUSY	CAW
	CC	CSW	DOSFLAGS	ERRET	ERRMSG	NUCON	NUM	PWAIT	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SENCCW	SILI	TEXTA
	WAIT												
DMSPNT	AACTFREE	AACTLKP	AFTIC	AFTRP	AFTSECT	AFTWP	AFVS	DMSLFS	FVSECT	F65535	NUCON	REGSAV3	R0
	R1	R11	R12	R13	R14	R15	R2	R4	R5	R6			
DMSPRT	ADMSERL	ADMSPIOC	AFINIS	ARDBUF	AREA	ASTATE	BITS	CC	CLASURO	CLOSIO	ERRET	FILE	FILEBUFF
	FILEMODE	FILENAME	FILETYPE	HEX	INSTALID	LOC	NUCON	NUM	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	TEXTA	TYP1403
	TYP3203	TYP3211											
DMSPRV	AERASE	AFINIS	ASYSREF	AWRBUF	BGCOM	BUFFER	CC	CDISK	DOSFLAGS	DOSMODE	DSKLST	ERROR	FNAME
	FTYPE	INPUT	LUBPT	NUCON	PUBADR	PUBCUU	PUBPT	RDCOUNT	RDDATA	RESET	R0	R1	R10
	R12	R14	R15	R2	R3	SEARCH	SEEK	SENSE	TEXT	TIC			
DMSPUN	ADMSERL	ADTID	ADTSECT	AFINIS	ARDBUF	ASTATE	BITS	CLASURO	CLOSIO	ERRET	FILE	FILEBUFF	FILEMODE
	FILENAME	FILETYPE	FVSFSTAD	LOC	NOTIME	NUCON	NUM	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	STATEFST	TEXTA	TYPPUN
DMSQRY	ABGCOM	ADTCYL	ADTDTA	ADTFDOS	ADTFLG1	ADTFLG2	ADTFLG3	ADTFRO	ADTFROS	ADTFRW	ADTFRWOS	ADTFSTC	ADTID
	ADTM	ADTMX	ADTNUM	ADTSECT	AEXTSECT	AFVS	AINTRTBL	ALDRTBLS	AOUTRTBL	ASYSNAMS	ASYSREF	AUSABRV	BGCOM
	BLANKS	CDISK	CMSSEG	DEC	DECDEC	DMSDBG	DOSBUFSP	DOSDD	DOSDEV	DOSDOS	DOSDSNAM	DOSDSTYP	DOSDUM
	DOSEXTNO	DOSEXTTB	DOSFIRST	DOSFLAGS	DOSINIT	DOSKPART	DOSLIBL	DOSMODE	DOSNUM	DOSOS	DOSOSDSN	DOSPERM	DOSSECT
	DOSSVC	DOSSYS	DOSTYPE	DOSUCNAM	DOSVOLNO	DOSVOLTB	DOSXXX	DTAD	DTADT	DUMMY	EDIT	EXTM	EXTSECT
	FCBDD	FCBDEV	FCBDSNAM	FCBDSTYP	FCBFIRST	FCBNUM	FCBSECT	FCBTAPID	FVSECT	INPUT	LOC	MACLIBL	MISFLAGS
	MSGFLAGS	NEGITS	NOABBREV	NOIMPCP	NOIMPEX	NOPAGREL	NORDYTIM	NOSTDSYN	NUCON	NUM	OPTFLAGS	OUTPUT	PRFPOFF
	PROTFLAG	REDERRID	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SEARCH	SYSCOM	SYSLINE	SYSNAMES	SYSNEND	TEXT	TIMCCW	TIMCHAR
	TXTLIBS	VCADTLKP	VCADTNXT	VIRTUAL									
DMSRDC	ABATABND	AERASE	AFINIS	ASCANN	ASTATEW	AWRBUF	BATDCMS	BATFLAGS	BATFLAG2	BATRUN	BUFFER	CLASURI	CLOSIO
	DEVTYPE	ERROR	FILE	FILEBUFF	FILEMODE	FILENAME	FMODE	IOAREA	NUCON	NUM	READ	RPLIST	R0
	R1	R10	R11	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SAVE	TEXT	TYPRDR										
DMSREA	NUM	R0	R1	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	SAVER0	SAVER1	SAVER14	SAVER15	SAVER2	TEXT							
DMSRNE	AERASE	AFINIS	AINCORE	ARDBUF	AWRBUF	ERROR	FMODE	FNAME	FSIZE	LOC	NUCON	PACK	PLIST
	R0	R1	R10	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	STRTNO	TEXT	TYPE	VCADTLKW									

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSRNM	AACTLKP	ADTCHBA	ADTFLG1	ADTFRO	ADTFRW	ADTFTYP	ADTM	ADTSECT	AFTADT	AFTSECT	AFVS	AKILLEX	ASTATEW
	ATFINIS	ATYPSRCH	AUPDISK	ERBIT	ERRCOD1	ERSFLAG	FILE	FSTM	FSTN	FSTSECT	FSTT	FVSECT	FVSERAS0
	FVSERAS1	FVSERAS2	KXFLAG	KXWANT	NEWMODE	NEWNAME	NEWTYPE	NUCON	NUM	ON	REGSAV1	R0	R1
	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	STATEFST	TEXT	UFDBUSY	VCADTLKP	VCFSTLKW							
DMSROS	ADTCYL	ADTDTA	ADTFDOS	ADTFLG1	ADTFLG2	ADTFLG3	ADTFORCE	ADTFROS	ADTFRWOS	ADTM	ADTSECT	BALR	CC
	CSW	DOSFIRST	DOSFLAGS	DOSSVC	DTAD	FCBBLKSZ	FCBDSMD	FCBDSNAM	FCBDSTYP	FCBFIRST	FCBIOSW2	FCBLRECL	FCBMEMBR
	FCBMVPDS	FCBNEXT	FCBOP	FCBOSDSN	FCBOSFST	FCBPROC	FCBRECFM	FCBSECT	FILEBUFF	FILEBYTE	FILENAME	FILEREAD	LOC
	NUCON	OPSECT	OSADTDSK	OSADTFST	OSADTVTA	OSADTVTB	OSFST	OSFSTALT	OSFSTBLK	OSFSTCHR	OSFSTDBK	OSFSTDSK	OSFSTDSN
	OSFSTEND	OSFSTEX4	OSFSTFLG	OSFSTFM	OSFSTFVF	OSFSTLRL	OSFSTLTH	OSFSTMEM	OSFSTMVL	OSFSTNTE	OSFSTNXT	OSFSTRFM	OSFSTRSW
	OSFSTTRK	OSFSTTYP	OSFSTUMV	OSFSTXNO	OSFSTXTN	PO	PS	READBLK	R0	R1	R10	R11	R12
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVEREGS	SEEK	SKIP
	TEXT	TYPE	TYP3350	UND	VAR	VCADTNXT	ZEROES						
DMSRRV	AERASE	AFINIS	AREA	ASTATE	ASYSREF	AWRBUF	BGCOM	BLANKS	CC	CDISK	DOSDD	DOSDEV	DOSDSK
	DOSFIRST	DOSFLAGS	DOSMODE	DOSOP	DOSOSFST	DOSSECT	DSKLST	ERROR	FNAME	FTYPE	INPUT	LUBPT	NUCON
	OSFST	OSFSTDSK	OSFSTXTN	OUTBUF	PUBPT	RDCOUNT	RDDATA	RESET	R0	R1	R10	R11	R12
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVE1	SEARCH	SEEK
	SENSE	TEXT	TIC										
DMSSAB	AABNSVC	ACMSSEG	ADMSFREB	AOSMODL	APGMSECT	BALR	CALLEE	CODE203	CURRSAVE	DCSSAVAL	DCSSFLAG	DCSSVTLD	DEBDCBAD
	EGPRS	EGPR0	EGPR1	EGPR11	EGPR12	EGPR14	EGPR15	EGPR9	ERRCODE	FCBDD	FCBDEV	FCBDUM	FCBFIRST
	FCBSECT	LASTUSER	LINKLAST	LOC	NUCON	OLDPSW	PGMOPSW	PGMSECT	RESET	RETRYBIT	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SCBPTR	SCBSAV12	SCBWORK	SETUP	SETUP2	SSAVE	SSAVEPRV	STAEBIT	STAIBIT	TPFUSR	TYPE	TYPFLAG	USAVEPTR
DMSSBD	DA	DATAEND	DECAREA	DECKYADR	DECLNGTH	DECRECPT	DECSDECB	DECTYPE	DMSSES	DMSSBSRT	DUMMY	FCBBYTE	FCBITEM
	FCBKEYS	FCBOP	FCBRECFM	FCBSECT	FCBXTENT	FINIS	IHADECB	IOBIN	IOBIOFLG	KEYCHNG	KEYCOUT	KEYLNGTH	KEYNAME
	KEYOP	KEYSECT	KEYTBLAD	KEYTBLNO	OPSECT	PS	R0	R1	R10	R11	R12	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	SEBSAV	SKEY	TBLLNGTH	VAR	
DMSSBS	AOPSECT	CHNGBYTE	DA	DECAREA	DECDCBAD	DECIOBPT	DECLNGTH	DECSDECB	DECTYPE	DMSSBD	DMSSEB	FCBBUFF	FCBBYTE
	FCBCATML	FCBCOUT	FCBDEV	FCBDSMD	FCBDSNAM	FCBINIT	FCBITEM	FCBMODE	FCBCP	FCBOS	FCBPDS	FCBREAD	FCBSECT
	FCBTAP	FCBTBSP	FCBXTENT	IHADEB	IHADECB	IOBBCSW	IOBBECBP	IOBBFLG	IOBCSW	IOBIN	IOBIOFLG	IOBOUT	NUCON
	OPSECT	OSIOTYPE	PO	PREVIOUS	PS	READ	R0	R1	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R8	TAPEDEV	TAPELIST	TAPEMASK	TAPEOPER	UND	VAR	WRITE
DMSSCN	BALRSAVE	CMNDLIST	NUCON	R0	R1	R12	R14	R15	R2	R3	R4	R5	R6
	R7	R8											
DMSSCR	BUFFLOC	CHNGFLAG	DECLTH	DMSGIO	EDCB	EDMSK	ERROR	FLAG	FLAGLOC	FLAG2	FMODE	FNAME	FTYPE
	FV	GIOPLIST	HOLDFLAG	ITEM	LINELOC	NUM	NUMLOC	PTR1	PTR2	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R9	SAVCNT	SAVEAR
	SCLNO	SCRBUFAD	SCRFLGS	SCRFLG2	TABLIN	TEXT	TRUNCCL	TWITCH	TYPE	TYPSCR	UTILFLAG	VERCOL1	VERLEN
	Y2												

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSSCT	ADMSROS	AOPSECT	CMSOP	DA	DECDCBAD	DECIOBPT	DECSDECB	FCBCATML	FCBCLOSE	FCBCOUT	FCBDEV	FCBDSNAM	FCBINIT
	FCBIOSW	FCBITEM	FCBOP	FCBOS	FCBOSFST	FCBPDS	FCBR13	FCBSECT	FCBTAP	FILENAME	IHADEB	IHADECB	IOBBFLG
	IOBCSW	IOBIOFLG	IOBOUT	MACDIRC	MACLIBL	NUCON	NUM	OPSECT	PS	RESET	R0	R1	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVER14
DMSSEB	ADMSROS	AOPSECT	BLK	CMNDLINE	CONRDCNT	CONRDCOD	CONREAD	CONSOLE	CONWR	CONWRBUF	CONWRCNT	CONWRCOD	CONWRITE
	DUMMY	FCBBUFF	FCBBYTE	FCBCASE	FCBCOUT	FCBDEV	FCBDSMD	FCBDSTYP	FCBFORM	FCBINIT	FCBIO	FCBIOSW	FCBIOSW2
	FCBITEM	FCBMEMBR	FCBMODE	FCBMVFIL	FCBMVPDS	FCBOP	FCBOPCB	FCBOS	FCBPROC	FCBPRPU	FCBREAD	FCBRECFM	FCBRECL
	FCBR13	FCBSECT	FCBTAPID	FXD	IHADECB	IOBBCSW	IOBBECBC	IOBBECBP	IOBIN	IOBIOFLG	NUCON	OPSECT	PO
	PRINTLST	PS	PUNCHLST	RDBUFF	RDCCW	RDCOUNT	READLST	R0	R1	R11	R13	R14	R15
	R2	R3	R8	SAVER14	SEBSAV	TAPE	TAPEBUFF	TAPECOUT	TAPEDEV	TAPELIST	TAPEMASK	TAPEOPER	TAPESIZE
	TSOATCNL	TSOFLAGS	TYPE	UND	VAR								
DMSSEG	DMSEDC	DMSEDI	DMSEXT	DMSGIO	DMSLGT	DMSLIB	DMSLSB	DMSLSY	DMSCLD	DMSSAB	DMSSBD	DMSSBS	DMSSCR
	DMSSCT	DMSSEB	DMSSLN	DMSSMN	DMSSOP	DMSSQS	DMSSVN	DMSSVT					
DMSSET	ABATABND	ABGCOM	ACMSSEG	ADEVTAB	ADMSERL	ADMSFREB	ADMSFRT	ADOSDCSS	ADTDTA	ADTFDOS	ADTFLG2	ADTM	ADTSECT
	AEXTSECT	AFREETAB	AINTRTBL	ALDRTBLS	ALTASAVE	AOSMODL	AOUTRTBL	APPSAVE	AREA	ASTATE	ASYSCOM	ASYSNAMS	ASYSREF
	AUSRAREA	BALR	BATDCMS	BATFLAGS	BATFLAG2	BATNOEX	BATRUN	BGCOM	CC	CMSDOS	CMSSEG	CMSVSAM	CODE
	CODE203	CPULOG	CURRDATE	DCSSAVAL	DCSSFLAG	DCSSJLNS	DCSSLDED	DCSSVTLD	DEC	DMSDBG	DOSFLAGS	DOSKPART	DOSMODE
	DOSSVC	DOSTRANS	DOSVSAM	ERROR	EXTSECT	FRDSECT	FREELCWE	FREELOW1	FRERESPG	JCSW3	JCSW4	JOBDATE	LOADSTRT
	LOC	LOCCNT	LTK	LUBPT	MAINHIGH	MISFLAGS	MODFLGS	MSGFLAGS	NEGITS	NOABBREV	NOIMPCP	NOIMPEX	NOPAGREL
	NORDYMSG	NORDYTIM	NOVMREAD	NUCKEY	NUCON	NUM	OFF	ON	OPTFLAGS	OSMODLDW	PIBPT	PPEND	PRFPOFF
	PROTFLAG	PUBPT	REDERRID	RESET	RGPRS	R0	R1	R10	R11	R12	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SEARCH	SEEK	SOB1	STRTADDR	SYSCODE	SYSLINE
	SYSLOAD	SYSNAMES	SYSNEND	SYSREF	SYSTEM	TBENT	TEXT	TIC	TIMCCW	TIMCHAR	TIMER	TIMINIT	TSOBLKS
	TYPE	UPSI	UPTMID	UPTSWS	USERCODE	USERKEY	VCADTLKP	VIRTUAL	VMSIZE				
DMSSLN	ADMSFREB	ADTRANS	AFINIS	AFVS	ALDRTBLS	ALIASENT	APGMSECT	ARDBUF	ASTATE	ASVCSECT	AUSRAREA	BALR	CODE203
	COMPSWT	CURRSAVE	DMSOLD	DMSSMNSB	DSKLIN	DUMCOM	DYLD	DYLIBO	DYMBRNM	DYNAEND	EGPRS	EGPR0	EGPR1
	EGPR13	EGPR14	EGPR15	ERROR	FILE	FREELOWE	FRSTLCC	FVSECT	F65535	LASTLMOD	LASTTMOD	LDRFLAGS	LINKLAST
	LINKSTRT	LOC	LOCCNT	MODLIST	NUCON	OLDPSW	OSRESET	OSSFLAGS	OSTEMP	PGMSECT	PRFTSYS	PRFUSYS	PROTFLAG
	SCBPTR	SSAVE	STRTADDR	SUBACT	SUBFLAG	SVCSECT	SYSTEM	TBENT	TEXT	USAVEPTR			
DMSSMN	ABGCOM	AUSRAREA	BALRSAVE	BGCOM	COMPSWT	CURRSAVE	DMSDBG	EGPR1	EGPR15	EOCADR	FREELOWE	FRERESPG	LOCCNT
	MAINHIGH	MAINLIST	MAINSTRT	NUCON	OSSFLAGS	OSSMNU	PPEND	R0	R1	R10	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SSAVE	TEXT	TIMCHAR	TOTLIBS
	VIRTUAL												
DMSSOP	AACTLKP	ACBID	ACMSCVT	ADMSFREB	ADTFLG1	ADTFRO	ADTM	ADTNACW	ADTSECT	AERASE	AFINIS	AFTADT	AFTFLG
	AFTFST	AFTIC	AFTIN	AFTPFST	AFTSECT	AFVS	AOPSECT	AOSRET	ASTATE	AUPDISK	BALR	BLK	CDISK
	CMSCVT	CMSNAME	CMSOP	CODE203	CURRSAVE	CVTAVIB	DA	DCBSAV	DEBDCBAD	DEBDEBID	DEBOPATB	DEVTYP	DMSSBS
	DMSSCTCE	DMSSCTCK	DMSSCTNP	DMSSQSGT	DMSSQSPT	DMSSQSUP	DOSDIRC	DOSLIBL	EGPR0	EGPR1	EGPR15	EGPR2	FCBBLKSZ
	FCBBUFF	FCBBYTE	FCBCASE	FCBCATML	FCBCLEAV	FCBCLOSE	FCBCON	FCBCOUT	FCBDCBCT	FCBDD	FCBDEV	FCBDOSL	FCBDSK
	FCBDSMD	FCBDSNAM	FCBDSTYP	FCBDUM	FCBFIRST	FCBFORM	FCBINIT	FCBIOSW	FCBIOSW2	FCBITEM	FCBKEYS	FCBLRECL	FCBMEMBR

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	FCBMODE	FCBMVPDS	FCBOP	FCBOS	FCBOSFST	FCBPDS	FCBPROC	FCBPROCC	FCBPROCO	FCBRDR	FCBRECFM	FCBRECL	FCBSECT
	FCBTAP	FCBTCLOS	FCBXTENT	FILEBYTE	FILEMODE	FILENAME	FILEREAD	FILETYPE	FSTD	FSTFLAGS	FSTFMODE	FSTRWDSK	FSTXRDSK
	FVSECT	FXD	F6	IHADEB	IOBDCBPT	IOBEND	IOBIN	IOBIOFLG	IOBNXTAD	IOBSTART	JFCBIND2	JFCBMASK	JFCDSORG
	JFCKEYLE	JFCLIMCT	JFCOPTCD	LASTUSER	LOC	MACDIRC	MACLIBL	NUCON	NUM	OPSECT	OSFST	OSFSTBLK	OSFSTCHR
	OSFSTLRL	OSFSTRFM	OSIOTYPE	PLIST	PO	PREVIOUS	PS	QS	RESET	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVER1
	SAVER15	SSAVE	STATER0	TAPEDEV	TAPELIST	TAPEMASK	TAPEOPER	TPFACB	TYPE	TYPFLAG	UND	USAVEPTR	VAR
DMSSQS	AOPSECT	BLK	DEBTCBAD	DMSSCTCE	DMSSCTCK	DMSSEB	FCBBUFF	FCBBYTE	FCBCLOSE	FCBCOUT	FCBDEV	FCBDSMD	FCBINIT
	FCBIORD	FCBIOSW	FCBIOWR	FCBITEM	FCBOP	FCBPVMB	FCBREAD	FCBSECT	FXD	IHADEB	IOBECB	IOBECBPT	IOBIN
	IOBIOFLG	IOBOUT	IOBSTART	IOBUPD	LOC	NUCON	OPSECT	OSIOTYPE	PREVIOUS	PS	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	UND	VAR
DMSSRT	ASCANO	ASTRINIT	DEC	DOSFLAGS	DOSSVC	FINIS	FLAG	INSIZE	MISFLAGS	NUCON	NUM	RELPAGES	RESET
	R0	R1	R12	R14	R15	R2	R3	R4	R5	R6	SKIP	TEXT	VCADTLKW
DMSSRV	AERASE	AFINIS	ASTATE	ASYSREF	AWRBUF	BGCOM	CC	CDISK	DOSDD	DOSDEV	DOSDSK	DOSFIRST	DOSFLAGS
	DOSMODE	DOSOP	DOSOSFST	DOSSECT	DSKLST	ERROR	FNAME	FTYPE	INPUT	LUBPT	NUCON	OSFST	OSFSTDSK
	OSFSTXTN	OUTBUF	PUBPT	RDCOUNT	RDDATA	RESET	R0	R1	R10	R12	R14	R15	R2
	R3	R4	R5	R9	SAVE1	SEARCH	SEEK	SENSE	TEXT	TIC			
DMSSSK	DEC	HEX	NUCON	NUM	R0	R1	R12	R14	R15	R2	R3	R4	R5
	R6	R8	R9	SYSTEM	TEXT	VMSIZE							
DMSSTG	ABGCOM	ADMSFREB	AEXTSECT	ALDRTBLS	ANCHENDA	ANCHSECT	ANCHSIZ	APGMSECT	ASTATEXT	ASYSCOM	ATSOCPPL	AUSRAREA	BALR
	BALRSAVE	BGCOM	CODE203	COMPSWT	CORESIZE	CURRSAVE	DMSDBG	DMSLGTA	DOSFLAGS	DOSKPART	DOSVSAM	DYLD	DYLIBO
	DYMBRNM	EGPR12	EGPR14	EGPR15	EOCADR	EXTSECT	FREELOWE	FRERESPG	F1	IJBBOX	LINKLAST	LINKSTRT	LOC
	LOCCNT	MACDIRC	MACLIBL	MAINHIGH	MAINLIST	MAINSTRT	MISFLAGS	NUCON	OLDPSW	OPTNBYTE	OSSFLAGS	PCTVSAM	PDSSECT
	PGMSECT	PICADDR	PPEND	RELPAGES	R0	R1	R10	R12	R13	R14	R15	R2	R3
	R4	R5	R6	R7	R8	R9	SCBPTR	SCBWORK	SSAVE	STIMEXIT	SYSCOM	TAXEADDR	TIMCHAR
	USAVEPTR	VIPINIT	VSAMFLG1	VSAMRUN	VSAMSERV								
DMSSTT	AACTLKP	ADMSERL	ADTFLG1	ADTFLG2	ADTFRO	ADTFROS	ADTFRW	ADTM	ADTMX	ADTSECT	AFTADT	AFTFLG	AFTFST
	AFTRD	AFTSECT	AFTWRT	AFVS	BALR12	DMSERR	DMSLAD	DMSLADW	DMSLFS	DMSLFSW	FILE	FSTFAP	FSTFAR
	FSTFAW	FSTFB	FSTFRO	FSTFROX	FSTFRW	FSTFRWX	FSTM	FSTSECT	FVSECT	FVSFSTAD	FVSFSTDT	FVSFSTM	FVSFSTN
	NUCON	OSFST	OSFSTFLG	OSFSTFM	REGSAV3	R0	R1	R10	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R9	STATEFST	STATER0	TEXT					
DMSSVN	ADMSFREB	AEXTSECT	AOPSECT	ATTN	BALR	CODE203	CONRDBUF	CONRDCNT	CONRDCOD	CONREAD	CONSTACK	CONWRBUF	CONWRCNT
	CONWRCOD	CONWRITE	CURRSAVE	DMSDBG	EGPR0	EGPR1	EGPR15	EXTFLAG	EXTSECT	FCBSECT	FSTFINRD	LOC	LSTFINRD
	NUCON	NUMFINRD	NUMPNDWR	OPSECT	OSSFLAGS	OSWAIT	PENDREAD	PENDWRIT	PS	REALTIMR	R0	R1	R10
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R8	SSAVE	STIMEXIT	TIMCHAR
	TIMER	TIMINIT	TSOATCNL	TSOFLAGS	WAITEND								
DMSSVT	ADMPEXEC	ADMSFREB	ADMSROS	AERASE	AEXTSECT	AOPSECT	APGMSECT	APIE	ARDBUF	ASTATE	ATFINIS	AUPDISK	AWRBUF
	BALR	CALLER	CHNGBYTE	CMNDLINE	CMSNAME	CMSOP	CMSTAXE	CODE203	CONRDCNT	CONREAD	CONWRBUF	CONWRCNT	CONWRITE

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	CORESIZE	CURRDATE	CURRSAVE	DATAEND	DATE	DECSDECB	DEVTAB	DEVTYPE	DIAGTIME	DIRNAME	DIRPTR	DMSDBG	DMSLGT
	DMSLSB	DMSSAB	DMSSBDFR	DMSSBS	DMSSCT	DMSSLN	DMSSLN3	DMSSLN42	DMSSLN6	DMSSLN7	DMSSLN8	DMSSLN9	DMSSMN
	DMSSMN10	DMSSMN4	DMSSMN5	DMSSOP	DMSSOP19	DMSSOP20	DMSSOP22	DMSSOP23	DMSSQS	DMSSVN	DMSSVN1	DMSSVN2	DMSSVN93
	DMSSVN94	DOSDD	DOSDIRC	DOSFIRST	DOSLIBL	DOSNEXT	DOSSECT	DUMPLIST	EFPRS	EGPR0	EGPR1	EGPR13	EGPR14
	EGPR15	EGPR2	EXTSECT	FCBBUFF	FCBBYTE	FCBCATML	FCBCOUT	FCBDD	FCBDEV	FCBDOSL	FCBDSK	FCBDSNAM	FCBDSTYP
	FCBDUM	FCBFIRST	FCBFORM	FCBINIT	FCBIOSW2	FCBITEM	FCBKEYS	FCBMMV	FCBMVPDS	FCBOP	FCBOS	FCBOSFST	FCBPDS
	FCBSECT	FCBTAB	FCBTAP	FCBTBSP	FCBXTENT	FILEBUFF	FILEBYTE	FILECOUT	FILEITEM	FILEMODE	FILENAME	FILETYPE	FLAG
	IHADEB	IHADECB	IHAJFCB	IOBIN	IOBIOFLG	JFCBMASK	JFCLRECL	KEYCHNG	KEYCOUT	KEYFORM	KEYLNGTH	KEYNAME	KEYOP
	KEYSECT	KEYTABLE	KEYTBLAD	KEYTBLNO	KEYTYPE	LINKSTRT	LOC	LOWSAVE	MACDIRC	MACLIBL	NEWBLKS	NUCON	NUM
	OLDPSW	OPSECT	OSIOTYPE	OSRESET	OSSFLAGS	OSTEMP	PDSBLKSI	PDSDIR	PDSSECT	PGMSECT	PLIST	PREVIOUS	PS
	READBLK	RESET	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SCBPTR	SEARCH	SSAVE	STIMEXIT	TAXEADDR	TAXEDEF	TAXEEXIT	TAXELNK
	TBLLNGTH	TEMPBYTE	TEXTA	TEXT3	TIMBUF	TIMCHAR	TIMER	TYPE	USAVEPTR	VAR	VCADTLKP	VMSIZE	WAIT
	WAITLIST												
DMSSYN	AFINIS	AFST	ARDBUF	ASTATE	AUSABRV	BLANKS	ERRCODE	ERROR	FILE	LOC	NOSTDSYN	NUCON	NUM
	OPTFLAGS	R0	R1	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7
	R8	SYSCOM	TEXT	TYPE									
DMSTIO	ADEVTAB	ATABEND	CC	CSW	DEVADDR	DEVMISC	DEVNAME	DEVSECT	DEVSIZE	NUCON	PLIST	R0	R1
	R11	R12	R13	R14	R15	SILI	TAPE						
DMSTMA	BLK	CSW	DMSLIB	ERROR	FINIS	FXD	PACK	R0	R1	R10	R11	R12	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVER10	TAPE	TEXT	TYPLIST
	VIRTUAL												
DMSTPD	BLK	CSW	DEC	DOSFLAGS	DOSSVC	ERROR	FILE	FILEBUFF	FILEMODE	FILENAME	FILETYPE	FLAG	FLAG2
	FXD	NUCON	NUM	R0	R1	R10	R11	R12	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	STOP	TEXT	VAR	VIRTUAL				
DMSTPE	AACTLKP	ADEVTAB	ADTFTYP	ADTM	ADTSECT	AERASE	AFINIS	AFTFST	AFTSECT	AFVS	AKILLEX	ASTATE	ATABEND
	ATYPSRCH	AUPDISK	AWRBUF	BSR	CL	CLASTAPE	DEC	DEVADDR	DEVMISC	DEVNAME	DEVSECT	DEVSIZE	ERROR
	FILE	FINIS	FLAGS	FSTD	FSTDBC	FSTFCL	FSTFV	FSTIC	FSTIL	FSTM	FSTN	FSTRP	FSTSECT
	FSTT	FSTWP	FTRDCONV	FTRDLDNS	FTRTRANS	FTR7TRK	FVSECT	HEX	INPUT	KXFLAG	KXWANT	LOC	NUCON
	NUM	OUTPUT	READ	RESET	R0	R1	R10	R11	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SAVER1	SAVER14	TAPE	TEXT	TYP2401	TYP2420
	TYP3420	UFDBUSY	VCFSTLKP	VCFSTLKW	WRBIT	WRITE	WTM						
DMSTQQ	ADTDTA	ADTFLG1	ADTFLG2	ADTFMFD	ADTFRW	ADTQQM	ADTSECT	AQQTRK	ATRKLKP	ATRKLKPX	COUNT	DTADT	FVSECT
	F4	F65535	NUCON	QQTRK	R0	R1	R11	R12	R13	R14	R15	R2	R4
	R6	TRKLSAVE											
DMSTRK	ADTFLG1	ADTFLG2	ADTFMFD	ADTFRW	ADTMSK	ADTRES	ADTSECT	ADT1ST	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9		
DMSTYP	AFINIS	ARDBUF	AREA	ASTATE	FILE	FMODE	FNAME	FTYPE	HEX	IOAREA	LOC	MSGFLAGS	NOTYPING

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	NUCON	NUM	R0	R1	R10	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	START	TEXT	TYPLIN								
DMSUPD	ADTFLG1	ADTFRO	ADTFRW	ADTM	ADTMX	ADTSECT	AERASE	AEXTEND	AFINIS	ARDBUF	ASTATE	AWRBUF	BLANKS
	BUFFA	CORITEM	CTL	CUE	DATE	DOSFLAGS	DOSSVC	ERRMSG	ERROR	FNAME	FPTR	FREAD	FREELEN
	FSTFV	FSTIL	FSTM	FSTSECT	ITEM	LOC	MISFLAGS	NEWNAME	NOERASE	NOREP	NUCON	NUM	OFF
	ON	PLIST	PTR1	PTR2	REGSAV	RELPAGES	RESET	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SPARES	TEMPSAVE	TEXT
	TEXTA	TYPE	VCADTLKP	VCADTLKW									
DMSVIB	ACMSCVT	ADMSERL	ASYSNAMS	AVIPWORK	AVSAMSYS	BALRSAVE	CMSVSAM	DEC	NUCON	NUM	RESET	R0	R1
	R12	R14	R15	R2	R3	R5	SYSNAMES	SYSNEND	TEXT	TYPE	VIRTUAL	VMSIZE	VSAMFLG1
	VSAMRUN												
DMSVIP	ACBAMBL	ACBAMO	ACBBFPL	ACBBUFND	ACBDDNM	ACBDOSID	ACBDTFID	ACBERFLG	ACBEXLST	ACBIBUF	ACBID	ACBIDD	ACBLEN
	ACBLIST	ACBMACRF	ACBOCEXT	ACBOCTER	ACBOEMPT	ACBOFLGS	ACBOKBUF	ACBOPEN	ACBPRTCT	ACBST	ACBSTRNO	ACBSTYP	ACBUAPTR
	ACMSRET	AOSRET	AVIPWORK	AVSAMSYS	BLANKS	CALLEE	CURRSAVE	DOSDD	DOSDEV	DOSDSMD	DOSDUM	DOSEXTNO	DOSEXTTB
	DOSFIRST	DOSFLAGS	DOSNEXT	DOSRC	DOSSECT	DOSSVC	DOSVOLNO	DOSVOLTB	DOSYSXXX	ERRET	EXENACTB	EXENADDR	EXLEODF
	EXLEODL	EXLEODP	EXLJRN	EXLJRNL	EXLLEN	EXLLERF	EXLLERL	EXLLERP	EXLSYNF	EXLSYNL	EXLSYNP	IKQACB	IKQEXLST
	IKQRPL	LOC	NRMRET	NUCON	NUM	OLDPSW	RESET	RETSAV	RPLACB	RPLAREA	RPLARG	RPLASY	RPLBUFL
	RPLCHAIN	RPLECBPR	RPLEOFDS	RPLFDBKC	RPLFLAG	RPLKEYL	RPLNUP	RPLOPT1	RPLOPT2	RPLRLEN	RPLRTNCD	RPLST	RPLSTRID
	RPLUPD	RPLVLERR	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SAVER0	SAVER1	SAVER14	SAVER2	SSAVE	SSAVEPRV	TEXT	TPFSVO
	TYPE	TYPFLAG	VIPINIT	VIPSOP	VIPTCLOS	VIRTUAL	VSAMFLG1	WAITING					
DMSVPD	DEC	DUMMY	EDIT	ERROR	FNAME	LOC	NUM	R0	R1	R11	R12	R14	R15
	R2	R3	R4	R5	R6	R7	R9	TEXT	VIRTUAL	WRITE			
DMSVSR	AAMSSYS	ABGCOM	ACBLIST	ACMSCVT	ADIKQLAB	ADMSFREB	ADMSVIB	ARURTBL	ASYSNAMS	AVIPWORK	AVSAMSYS	AVSRWORK	BALR
	BGCOM	CMSAMS	CMSCVT	CMSVSAM	CODE203	CVTAVIB	DOSFLAGS	DOSMODE	DOSSVC	LOC	NUCON	PIB2PTR	PIK
	PPEND	REGSAV	R0	R1	R12	R13	R14	R15	R2	R3	R4	R5	R6
	R7	R8	SYSNAMES	SYSNEND	VIPINIT	VSAMFLG1	VSAMRUN	VSAMSERV	VSAMSOS				
DMSXCP	ADIKQLAB	ADMSERL	ADMSFREB	ADTDTA	ADTFDOS	ADTFLG2	ADTFLG3	ADTFROS	ADTFRW	ADTID	ADTM	ADTSECT	AFINIS
	ARDBUF	ASTATE	ASYSREF	AWRBUF	BALR	BGCOM	CALLER	CC	CCBCCW	CCBCNT	CCBCOM1	CCBCOM2	CCBCSW
	CCBCSW1	CCBCSW2	CCBDC	CCBEOC	CCBEOF	CCBERMAP	CCBILEN	CCBNOREC	CCBSUCLS	CCBSUNUM	CCBSYMU	CCBUE	CCBVER
	CD	CODE203	CONWR	CSW	DATACHK	DMSCCB	DOSBUFF	DOSBUFSP	DOSBYTE	DOSCBID	DOSCOUT	DOSDD	DOSDEV
	DOSDSK	DOSDSMD	DOSDSNAM	DOSDSTYP	DOSDUM	DOSEXTCX	DOSEXTNO	DOSEXTTB	DOSFIRST	DOSFLAGS	DOSFORM	DOSINIT	DOSITEM
	DOSNEXT	DOSNUM	DOSOP	DOSOSDSN	DOSOSFST	DOSREAD	DOSSAVE	DOSSECT	DOSSENSE	DOSTAPID	DOSTYPE	DOSUCNAM	DOSVOLNO
	DOSVOLTB	DOSWORK	DOSYSXXX	EGPR5	ERRMSG	ERROR	FSTIC	FSTIL	FSTSECT	F5	F7	INPUT	LOC
	LUBPT	NDIKQLAB	NICLPT	NOP	NUCON	OFF	ON	OUTPUT	PUBADR	PUBCUU	PUBDEVT	PUBDSKM	PUBPT
	PUBTAPM1	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R8	R9	SEEK	SILI	SKIP	SSAVE	SYSTEM	TAPE	TEXT	TIC	TYPE
	VAR	VCADTLKP	VCFSTLKP										
DMSZAP	ADTRANS	BLANKS	BUFSIZE	CLOSELIB	COMNAME	CONSOLE	DEC	DOSFLAGS	DOSSVC	ERROR	FILE	FLAGS	FSCBBUFF
	FSCBD	FSCBFN	FSCBFT	FSCBFV	FSTFB	FSTFRW	FSTFV	FSTIC	FSTIL	FSTM	FSTSECT	HEX	INPUT
	LASTLINE	LASTREC	LOC	MODDISP	NUCON	NUM	RESET	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVESIZE	TABEND	TEXT
	TYPE	VIRTUAL											

```
LABEL     COUNT      REFERENCES

AABNSVC   000001     DMSSAB
AACTFREE  000004     DMSBRD    DMSBWR    DMSPNT
AACTFRET  000005     DMSBWR    DMSERS    DMSFNS
AACTLKP   000013     DMSBRD    DMSBWR    DMSCPY    DMSERS    DMSFNS    DMSINI    DMSPNT    DMSRNM    DMSSOP    DMSSTT    DMSTPE
AACTNXT   000001     DMSERS
AADTLKP   000004     DMSDLK    DMSLBM    DMSLBT    DMSMVE
AADTLKW   000012     DMSARX    DMSASM    DMSCPY    DMSDLK    DMSIFC    DMSLBM    DMSLBT    DMSLKD
AAMSSYS   000004     DMSAMS    DMSDOS    DMSVSR
ABATABND  000012     DMSABN    DMSASN    DMSBTB    DMSCIO    DMSDSK    DMSERR    DMSFLD    DMSITE    DMSPIO    DMSRDC    DMSSET
ABATLIMT  000004     DMSBTB    DMSCIO    DMSITE    DMSPIO
ABATPROC  000004     DMSARE    DMSBTB    DMSCPF    DMSCRD
ABGCOM    000033     DMSALU    DMSAMS    DMSASN    DMSBAB    DMSBOP    DMSDOS    DMSFET    DMSINS    DMSOPT    DMSQRY    DMSSET    DMSSMN
                     DMSSTG    DMSVSR
ABNBIT    000004     DMSABN    DMSETP    DMSDOS
ABNERLST  000010     DMSABN    DMSITP
ABNPAS13  000001     DMSABN
ABNPSW    000030     DMSABN    DMSDBG    DMSFRE    DMSITI    DMSITP    DMSITS
ABNREGS   000013     DMSABN    DMSDBG    DMSFRE    DMSITI    DMSITP    DMSITS
ABNRR     000002     DMSABN
ABORT     000001     DMSDLK
ABWSECT   000008     DMSABN    DMSDBG    DMSFRE    DMSITI    DMSITP    DMSITS
ACALL     000004     DMSFRF
ACBAMBL   000001     DMSVIP
ACBAMO    000005     DMSCLS    DMSVIP
ACBBFPL   000001     DMSVIP
ACBBUFND  000001     DMSVIP
ACBCAT    000001     DMSBOP
ACBDDNM   000002     DMSBOP    DMSVIP
ACBDOSID  000001     DMSVIP
ACBDTFID  000001     DMSVIP
ACBERFLG  000007     DMSBOP    DMSVIP
ACBEXLST  000004     DMSVIP
ACBIBUF   000001     DMSVIP
ACBID     000006     DMSSOP    DMSVIP
ACBIDD    000007     DMSVIP
ACBIN     000001     DMSBOP
ACBINFLG  000001     DMSBOP
ACBLEN    000001     DMSVIP
ACBLIST   000011     DMSVIP    DMSVSR
ACBMACRF  000001     DMSVIP
ACBMACR1  000002     DMSBOP
ACBOCEXT  000001     DMSVIP
ACBOCTER  000001     DMSVIP
ACBOEMPT  000001     DMSVIP
ACBOFLGS  000003     DMSBOP    DMSVIP
ACBOKBUF  000001     DMSVIP
```

```
LABEL     COUNT     REFERENCES

ACBOLIGN 000001    DMSBOP
ACBOPEN  000002    DMSVIP
ACBOUT   000001    DMSBOP
ACBPRTCT 000001    DMSVIP
ACBST    000001    DMSVIP
ACBSTRNO 000001    DMSVIP
ACBSTSKP 000001    DMSBOP
ACBSTYP  000001    DMSVIP
ACBUAPTR 000001    DMSVIP
ACMSCVT  000004    DMSINS   DMSSOP   DMSVIB   DMSVSR
ACMSRET  000004    DMSDOS   DMSLDR   DMSVIP
ACMSSEG  000011    DMSEDX   DMSEXC   DMSINS   DMSITS   DMSSAB   DMSSET
ACTIVE   000005    DMSBRD   DMSCIT   DMSMOD   DMSOPL
ADEVTAB  000017    DMSAMS   DMSASN   DMSDBD   DMSEDI   DMSEDX   DMSFOR   DMSGIO   DMSINI   DMSSET   DMSTIO   DMSTPE
ADIKQLAB 000006    DMSDOS   DMSVSR   DMSXCP
ADIOSECT 000005    DMSACM   DMSDIO   DMSFNS   DMSITI
ADISK    000006    DMSDSK   DMSINS   DMSNUC
ADMPEXEC 000001    DMSSVT
ADMSCRD  000002    DMSBTP   DMSDBG
ADMSERL  000053    DMSAMS   DMSBOP   DMSBWR   DMSCIO   DMSCLS   DMSDBG   DMSDOS   DMSEDI   DMSERS   DMSFCH   DMSFET   DMSFNS
                   DMSFRE   DMSITS   DMSMOD   DMSPIO   DMSPRT   DMSPUN   DMSSET   DMSSTT   DMSVIB   DMSXCP
ADMSFREB 000195    DMSABN   DMSACC   DMSACF   DMSACM   DMSALU   DMSAMS   DMSAUD   DMSBOP   DMSBRD   DMSBWR   DMSCAT   DMSCIT
                   DMSCLS   DMSCMP   DMSCRD   DMSCWR   DMSDIO   DMSDLE   DMSDMP   DMSDOS   DMSEDX   DMSERS   DMSEXC   DMSEXT
                   DMSFCH   DMSFET   DMSFNS   DMSFOR   DMSHDI   DMSHDS   DMSINS   DMSINT   DMSITE   DMSITP   DMSITS   DMSLAD
                   DMSLAF   DMSLDR   DMSLFS   DMSLGT   DMSLIB   DMSLSB   DMSMOD   DMSOLD   DMSOPL   DMSOR1   DMSSAB   DMSSET
                   DMSSLN   DMSSOP   DMSSTG   DMSSVN   DMSSVT   DMSVSR   DMSXCP
ADMSFRT  000002    DMSSET
ADMSLIO  000020    DMSOLD
ADMSOVS  000008    DMSITS   DMSOVR
ADMSPIOC 000001    DMSPRT
ADMSROS  000016    DMSACM   DMSALU   DMSLDS   DMSLFS   DMSSCT   DMSSEB   DMSSVT
ADMSVIB  000001    DMSVSR
ADOSDCSS 000002    DMSITS   DMSSET
ADTADD   000009    DMSACF   DMSACM   DMSAUD   DMSDIO   DMSERS   DMSFNS
ADTB     000001    DMSNUC
ADTC     000001    DMSNUC
ADTCFST  000006    DMSACF   DMSCPY   DMSERS
ADTCHBA  000017    DMSACF   DMSCPY   DMSERS   DMSLFS   DMSRNM
ADTCYL   000008    DMSACM   DMSFOR   DMSLDS   DMSQRY   DMSROS
ADTD     000001    DMSNUC
ADTDTA   000027    DMSACC   DMSACM   DMSARE   DMSASN   DMSAUD   DMSBWR   DMSDIO   DMSFNS   DMSFOR   DMSQRY   DMSROS   DMSSET
                   DMSTQQ   DMSXCP
ADTE     000001    DMSNUC
ADTF     000001    DMSNUC
ADTFALNM 000003    DMSACF
ADTFALTY 000004    DMSACF
```

```
LABEL     COUNT      REFERENCES


ADTFALUF 000004     DMSACC    DMSACF    DMSFOR
ADTFDA   000025     DMSABN    DMSACC    DMSACF    DMSALU    DMSAUD    DMSFOR    DMSINS    DMSLAD    DMSLFS    DMSLST
ADTFDOS  000017     DMSACC    DMSASN    DMSBOP    DMSDLB    DMSEXT    DMSFOR    DMSQRY    DMSRCS    DMSSET    DMSXCP
ADTFFSTF 000008     DMSABN    DMSACC    DMSACF    DMSALU    DMSFOR    DMSINS
ADTFFSTV 000007     DMSACC    DMSINS    DMSLAD    DMSLFS
ADTFLG1  000105     DMSABN    DMSACC    DMSACF    DMSACM    DMSALU    DMSARE    DMSARN    DMSARX    DMSASM    DMSASN    DMSBOP    DMSBWR
                    DMSCPY    DMSDIO    DMSDLK    DMSDSL    DMSERS    DMSFOR    DMSINS    DMSLAD    DMSLAF    DMSLBM    DMSLBT    DMSLDS
                    DMSLFS    DMSLLU    DMSLST    DMSMVE    DMSQRY    DMSRNM    DMSROS    DMSSCP    DMSSTT    DMSTQQ    DMSTRK    DMSUPD
ADTFLG2  000066     DMSABN    DMSACC    DMSACF    DMSACM    DMSALU    DMSARE    DMSASN    DMSBCP    DMSDLB    DMSEXT    DMSFOR    DMSLAD
                    DMSLDS    DMSLFS    DMSLST    DMSQRY    DMSROS    DMSSET    DMSSTT    DMSTQQ    DMSTRK    DMSXCP
ADTFLG3  000030     DMSACC    DMSACF    DMSACM    DMSALU    DMSARE    DMSAUD    DMSBOP    DMSBWR    DMSFNS    DMSINS    DMSLFS    DMSLLU
                    DMSQRY    DMSROS    DMSXCP
ADTFMDRO 000003     DMSACF
ADTFMFD  000006     DMSACM    DMSBOP    DMSEXT    DMSTQQ    DMSTRK
ADTFMIN  000004     DMSABN    DMSACC    DMSALU
ADTFNOAB 000002     DMSARE    DMSAUD
ADTFORCE 000005     DMSACC    DMSACF    DMSACM    DMSINS    DMSROS
ADTFQQF  000005     DMSABN    DMSACM    DMSALU    DMSFOR
ADTFRO   000034     DMSACC    DMSACF    DMSACM    DMSALU    DMSARE    DMSASN    DMSBOP    DMSDIO    DMSERS    DMSFOR    DMSLAD    DMSLBM
                    DMSLBT    DMSLDS    DMSLFS    DMSLST    DMSMVE    DMSQRY    DMSRNM    DMSSCP    DMSSTT    DMSUPD
ADTFROS  000033     DMSABN    DMSACC    DMSACF    DMSALU    DMSARE    DMSASN    DMSBOP    DMSDLB    DMSEXT    DMSFOR    DMSLAD    DMSLDS
                    DMSLFS    DMSLST    DMSQRY    DMSROS    DMSSTT    DMSXCP
ADTFRW   000071     DMSACC    DMSACF    DMSACM    DMSALU    DMSARE    DMSARN    DMSARX    DMSASM    DMSASN    DMSBOP    DMSBWR    DMSCPY
                    DMSDIO    DMSDLK    DMSDSL    DMSERS    DMSFOR    DMSLAD    DMSLAF    DMSLBM    DMSLBT    DMSLDS    DMSLFS    DMSLLU
                    DMSLST    DMSMVE    DMSQRY    DMSRNM    DMSSTT    DMSTQQ    DMSTRK    DMSUPD    DMSXCP
ADTFRWOS 000004     DMSLLU    DMSQRY    DMSROS
ADTFSORT 000003     DMSACF    DMSINS    DMSLFS
ADTFSTC  000015     DMSACC    DMSACF    DMSALU    DMSARE    DMSBWR    DMSERS    DMSINS    DMSQRY
ADTFTYP  000012     DMSACF    DMSALU    DMSDSK    DMSFNS    DMSLFS    DMSRNM    DMSTPE
ADTFUPD1 000006     DMSAUD    DMSFNS
ADTFVS   000001     DMSLAD
ADTFXCHN 000005     DMSBWR    DMSFNS
ADTG     000001     DMSNUC
ADTHBCT  000016     DMSABN    DMSACC    DMSACF    DMSACM    DMSAUD    DMSERS    DMSFOR    DMSLAD    DMSLFS
ADTID    000012     DMSACM    DMSALU    DMSDSK    DMSFOR    DMSLDS    DMSLST    DMSPUN    DMSQRY    DMSXCP
ADTLAST  000006     DMSAUD    DMSFOR
ADTLEFT  000003     DMSFOR    DMSLAD
ADTLFST  000002     DMSERS    DMSLFS
ADTLHBA  000007     DMSACC    DMSACF    DMSERS    DMSFOR    DMSLFS
ADTM     000093     DMSABN    DMSACC    DMSACF    DMSALU    DMSAMS    DMSARE    DMSARN    DMSARX    DMSASM    DMSBWR    DMSCMP    DMSCPY
                    DMSDLK    DMSDSL    DMSEDX    DMSERS    DMSEXC    DMSEXT    DMSFOR    DMSIFC    DMSLAD    DMSLAF    DMSLBM    DMSLDS
                    DMSLFS    DMSLKD    DMSLST    DMSQRY    DMSRNM    DMSROS    DMSSET    DMSSCP    DMSSTT    DMSTPE    DMSUPD    DMSXCP
ADTMFDA  000004     DMSABN    DMSACF    DMSAUD
ADTMFDN  000014     DMSABN    DMSACC    DMSACF    DMSACM    DMSALU    DMSAUD
ADTMSK   000011     DMSACC    DMSACM    DMSALU    DMSAUD    DMSFOR    DMSTRK
ADTMX    000030     DMSACC    DMSACM    DMSALU    DMSARN    DMSARX    DMSASM    DMSBWR    DMSLAF    DMSLFS    DMSQRY    DMSSTT    DMSUPD
```

LABEL	COUNT	REFERENCES											
ADTMXBML	000001	DMSACM											
ADTNACW	000008	DMSBWR	DMSFNS	DMSSOP									
ADTNUM	000012	DMSACC	DMSACM	DMSAUD	DMSFOR	DMSQRY							
ADTPQM1	000010	DMSACM	DMSALU	DMSAUD	DMSFOR								
ADTPQM2	000009	DMSACC	DMSACF	DMSACM	DMSAUD	DMSFOR							
ADTPQM3	000006	DMSABN	DMSACC	DMSACM	DMSALU	DMSFOR							
ADTPSTM	000006	DMSLAD	DMSLFS										
ADTPTR	000002	DMSLAD											
ADTQQM	000005	DMSACM	DMSALU	DMSFOR	DMSTQQ								
ADTRANS	000012	DMSLSB	DMSMOD	DMSOLD	DMSSLN	DMSZAP							
ADTRES	000018	DMSACC	DMSACF	DMSACM	DMSALU	DMSBWR	DMSERS	DMSFNS	DMSFOR	DMSLAD	DMSLFS	DMSTRK	
ADTROX	000003	DMSACM	DMSALU										
ADTS	000001	DMSNUC											
ADTSECT	000120	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBWR	DMSCMP	DMSCPY	DMSDIO	DMSDLE	DMSDLK	DMSDSK	DMSDSL	DMSEDX	DMSERS	DMSEXC
		DMSEXT	DMSFNS	DMSFOR	DMSIFC	DMSINS	DMSLAD	DMSLAF	DMSLEM	DMSLET	DMSLDS	DMSLFS	DMSLKD
		DMSLLU	DMSLST	DMSMVE	DMSPUN	DMSQRY	DMSRNM	DMSROS	DMSSET	DMSSOP	DMSSTT	DMSTPE	DMSTQQ
		DMSTRK	DMSUPD	DMSXCP									
ADTUSED	000010	DMSACC	DMSACM	DMSFOR									
ADTXNREC	000005	DMSFNS											
ADTY	000001	DMSNUC											
ADTZ	000001	DMSNUC											
ADT1ST	000007	DMSACC	DMSFOR	DMSTRK									
AEDLIN	000001	DMSEDX											
AERASE	000045	DMSAMS	DMSBOP	DMSCLS	DMSDLK	DMSDSK	DMSDSL	DMSEDI	DMSFNS	DMSLIO	DMSLLU	DMSLST	DMSMOD
		DMSOLD	DMSPRV	DMSRDC	DMSRNE	DMSRRV	DMSSOP	DMSSRV	DMSSVT	DMSTPE	DMSUPD		
AERR	000001	DMSITS											
AEXEC	000002	DMSEXC											
AEXTEND	000007	DMSEDI	DMSEDX	DMSUPD									
AEXTSECT	000014	DMSINS	DMSINT	DMSIOW	DMSITE	DMSQRY	DMSSET	DMSSTG	DMSSVN	DMSSVT			
AFINIS	000068	DMSACC	DMSARE	DMSCLS	DMSCMP	DMSDLK	DMSDSK	DMSEDI	DMSEDX	DMSEXC	DMSEXT	DMSFOR	DMSGLB
		DMSLDR	DMSLIB	DMSLIO	DMSLLU	DMSMOD	DMSOLD	DMSPRT	DMSPRV	DMSPUN	DMSRDC	DMSRNE	DMSRRV
		DMSSLN	DMSSOP	DMSSRV	DMSSYN	DMSTPE	DMSTYE	DMSUPD	DMSXCP				
AFLAGLOC	000001	DMSEDX											
AFREETAB	000006	DMSFRE	DMSSET										
AFST	000001	DMSSYN											
AFSTFNRD	000004	DMSEDI	DMSEDX										
AFSTLKP	000004	DMSCPY											
AFSTLKW	000001	DMSCPY											
AFTADT	000024	DMSBRD	DMSBWR	DMSERS	DMSFNS	DMSLAF	DMSRNM	DMSSOP	DMSSTT				
AFTCLA	000012	DMSBRD	DMSBWR	DMSFNS									
AFTCLB	000010	DMSBRD	DMSBWR	DMSFNS									
AFTCLD	000015	DMSBRD	DMSBWR	DMSFNS									
AFTCLDX	000005	DMSBWR	DMSFNS										
AFTCLN	000014	DMSBRD	DMSBWR	DMSFNS									
AFTCLX	000006	DMSBWR	DMSFNS										

LABEL	COUNT	REFERENCES											
AFTD	000002	DMSBWR											
AFTDBA	000019	DMSBRD	DMSBWR	DMSFNS									
AFTDBC	000008	DMSBWR	DMSERS										
AFTDBD	000010	DMSBRD	DMSBWR	DMSFNS									
AFTDBF	000003	DMSBWR											
AFTDBN	000010	DMSBRD	DMSBWR										
AFTFB	000001	DMSLAF											
AFTFBA	000005	DMSBRD	DMSBWR	DMSFNS									
AFTFCL	000012	DMSBRD	DMSBWR	DMSERS	DMSFNS								
AFTFCLA	000008	DMSBRD	DMSBWR	DMSFNS									
AFTFCLX	000008	DMSBWR	DMSFNS										
AFTFLG	000040	DMSBRD	DMSBWR	DMSERS	DMSFNS	DMSLAF	DMSSOP	DMSSTT					
AFTFLG2	000016	DMSBWR	DMSFNS										
AFTFSF	000002	DMSLAF											
AFTFST	000009	DMSBRD	DMSBWR	DMSFNS	DMSLAF	DMSSOP	DMSSTT	DMSTPE					
AFTFULD	000002	DMSBWR	DMSFNS										
AFTFV	000007	DMSBRD	DMSBWR										
AFTIC	000012	DMSBRD	DMSBWR	DMSCPY	DMSPNT	DMSSOP							
AFTID	000010	DMSBRD	DMSBWR										
AFTIL	000006	DMSBRD	DMSBWR										
AFTIN	000014	DMSBRD	DMSBWR	DMSSOP									
AFTLD	000002	DMSLAF											
AFTM	000008	DMSBWR	DMSFNS	DMSINT	DMSLAF								
AFTN	000005	DMSBWR	DMSFNS	DMSINT	DMSLAF								
AFTNEW	000005	DMSBWR	DMSFNS										
AFTOCLDX	000003	DMSBWR											
AFTOLDCL	000006	DMSBWR											
AFTPFST	000007	DMSERS	DMSFNS	DMSLAF	DMSSOP								
AFTPTR	000012	DMSLAF											
AFTRD	000006	DMSBRD	DMSBWR	DMSFNS	DMSSTT								
AFTRP	000008	DMSBRD	DMSBWR	DMSPNT									
AFTSECT	000026	DMSBRD	DMSBWR	DMSCPY	DMSERS	DMSFNS	DMSINT	DMSLAF	DMSPNT	DMSRNM	DMSSOP	DMSSTT	DMSTPE
AFTT	000001	DMSLAF											
AFTUSED	000004	DMSFNS	DMSLAF										
AFTWP	000010	DMSBWR	DMSFNS	DMSINT	DMSPNT								
AFTWRT	000008	DMSBRD	DMSBWR	DMSFNS	DMSSTT								
AFVS	000053	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCIT	DMSCRD
		DMSCWR	DMSCWT	DMSDIO	DMSDOS	DMSDSK	DMSERS	DMSEXC	DMSFNS	DMSINT	DMSITI	DMSITP	DMSITS
		DMSLAD	DMSLFS	DMSMOD	DMSPNT	DMSQRY	DMSRNM	DMSSLN	DMSSOP	DMSSTT	DMSTPE		
AGETCLK	000001	DMSEXT											
AINCORE	000005	DMSEDI	DMSRNE										
AINTRTBL	000008	DMSABN	DMSCRD	DMSQRY	DMSSET								
AIOSECT	000008	DMSABN	DMSCIT	DMSDBG	DMSHDI	DMSINT	DMSITI						
AKILLEX	000010	DMSACC	DMSAUD	DMSBWR	DMSDBG	DMSDIO	DMSDSK	DMSERS	DMSFNS	DMSRNM	DMSTPE		
ALCHAR1	000002	DMSEDI											
ALCHAR2	000002	DMSEDI											

```
LABEL     COUNT     REFERENCES

ALDRTBLS 000028     DMSBTB   DMSFET   DMSGND   DMSINS   DMSLDR   DMSLOA   DMSMDP   DMSMCD   DMSOLD   DMSQRY   DMSSET   DMSSLN
                    DMSSTG
ALIASENT 000004     DMSLIO   DMSSLN
ALINELOC 000001     DMSEDX
ALTASAVE 000008     DMSAMS   DMSDOS   DMSITP   DMSSET
ALTLIST  000008     DMSEDI
ALTMODE  000006     DMSEDX
ANCHENDA 000003     DMSDOS   DMSSTG
ANCHENTP 000001     DMSDOS
ANCHINST 000001     DMSDOS
ANCHLDPT 000002     DMSDOS
ANCHLENG 000002     DMSDOS
ANCHPHLN 000001     DMSDOS
ANCHPHNM 000005     DMSDOS
ANCHSECT 000003     DMSDOS   DMSSTG
ANCHSIZ  000005     DMSFCH   DMSSTG
ANCHSTSW 000001     DMSDOS
ANUCEND  000003     DMSDIO   DMSHDI   DMSHDS
ANUMLOC  000001     DMSEDX
AOPSECT  000026     DMSABN   DMSARN   DMSCRD   DMSCWR   DMSCWT   DMSDBG   DMSEXC   DMSEXT   DMSINS   DMSINT   DMSSBS   DMSSCT
                    DMSSEB   DMSSOP   DMSSQS   DMSSVN   DMSSVT
AOSMODL  000022     DMSINS   DMSITS   DMSLDR   DMSSAB   DMSSET
AOSRET   000003     DMSDOS   DMSSOP   DMSVIP
AOUTRTBL 000007     DMSABN   DMSCWR   DMSQRY   DMSSET
APGMSECT 000007     DMSITP   DMSSAB   DMSSLN   DMSSTG   DMSSVT
APIE     000001     DMSSVT
APOINT   000002     DMSEXT   DMSLIB
APPSAVE  000004     DMSAMS   DMSDOS   DMSITP   DMSSET
APRILB   000006     DMSLDR   DMSOLD
APSV     000035     DMSLDR   DMSLGT   DMSLIB   DMSLIO   DMSLSB   DMSOLD
AQQTRK   000003     DMSBWR   DMSTQQ
AQQTRKX  000006     DMSBWR   DMSERS   DMSFNS
ARDBUF   000059     DMSCMP   DMSDLK   DMSDSK   DMSEDI   DMSEDX   DMSEXT   DMSGLB   DMSLBT   DMSLDR   DMSLGT   DMSMOD   DMSOLD
                    DMSPRT   DMSPUN   DMSRNE   DMSSLN   DMSSVT   DMSSYN   DMSTYP   DMSUPD   DMSXCP
ARDTK    000011     DMSACF   DMSACM   DMSBRD   DMSBWR   DMSERS   DMSFNS   DMSFOR   DMSMCD
AREA     000029     DMSCMP   DMSEDI   DMSINS   DMSPRT   DMSRRV   DMSSET   DMSTYP
ARFLG    000002     DMSDOS
ARGMAX   000001     DMSDBG
ARGS     000046     DMSDBD   DMSDBG   DMSITE   DMSNUC
ARGSAV   000008     DMSDBG
ARGSCT   000016     DMSDBG
ARURTBL  000006     DMSDOS   DMSVSR
ASCANN   000005     DMSAMS   DMSBTP   DMSLDR   DMSOLD   DMSRDC
ASCANO   000002     DMSEXT   DMSSRT
ASCBPTR  000002     DMSINT
ASSTAT   000002     DMSFRE   DMSINS
```

```
LABEL     COUNT      REFERENCES

ASTATE    000041     DMSAMS   DMSBOP   DMSDLK   DMSDSK   DMSDSL   DMSEDI   DMSEDX   DMSEXT   DMSFCH   DMSFLD   DMSGLB   DMSGND
                     DMSINS   DMSLDR   DMSLIB   DMSMOD   DMSOLD   DMSPRT   DMSPUN   DMSRRV   DMSSET   DMSSLN   DMSSOP   DMSSRV
                     DMSSVT   DMSSYN   DMSTPE   DMSTYP   DMSUPD   DMSXCP
ASTATEW   000007     DMSAMS   DMSEDX   DMSERS   DMSMOD   DMSRDC   DMSRNM
ASTATEXT  000002     DMSINS   DMSSTG
ASTRINIT  000002     DMSARN   DMSSRT
ASUBFST   000003     DMSABN   DMSINT
ASUBRET   000002     DMSINT
ASUBSECT  000006     DMSABN   DMSINM   DMSINT
ASUBSTAT  000003     DMSABN   DMSINT
ASVCSECT  000028     DMSCIT   DMSFRE   DMSHDS   DMSINT   DMSITE   DMSITS   DMSLAD   DMSLFS   DMSOVR   DMSOVS   DMSSLN
ASYSCOM   000011     DMSBAB   DMSBOP   DMSDOS   DMSFET   DMSITP   DMSSET   DMSSTG
ASYSNAMS  000025     DMSAMS   DMSBOP   DMSBTP   DMSDOS   DMSEDX   DMSEXC   DMSINS   DMSINT   DMSITS   DMSQRY   DMSSET   DMSVIB
                     DMSVSR
ASYSREF   000027     DMSASN   DMSBOP   DMSCLS   DMSDLB   DMSDMP   DMSDOS   DMSFCH   DMSINS   DMSITP   DMSLLU   DMSOPL   DMSPRV
                     DMSQRY   DMSRRV   DMSSET   DMSSRV   DMSXCP
ATABEND   000005     DMSAMS   DMSTIO   DMSTPE
ATFINIS   000006     DMSBWR   DMSERS   DMSRNM   DMSSVT
ATRKLKP   000003     DMSAUD   DMSBWR   DMSTQQ
ATRKLKPX  000012     DMSAUD   DMSBWR   DMSERS   DMSFNS   DMSTQQ
ATSOCPPL  000001     DMSSTG
ATTN      000016     DMSABN   DMSCIT   DMSCRD   DMSEDI   DMSFNC   DMSSVN
ATTNHIT   000004     DMSCIT   DMSITI
ATTNLEN   000007     DMSEDI
ATYPSRCH  000005     DMSACF   DMSDSK   DMSFNS   DMSRNM   DMSTPE
AUPDISK   000016     DMSARE   DMSBWR   DMSDSK   DMSERS   DMSFNS   DMSFOR   DMSRNM   DMSSOP   DMSSVT   DMSTPE
AUPIE     000002     DMSITP
AUSABRV   000004     DMSABN   DMSINA   DMSQRY   DMSSYN
AUSERRST  000003     DMSERR
AUSRAREA  000039     DMSABN   DMSBRD   DMSBTB   DMSFCH   DMSFET   DMSFRE   DMSINS   DMSINT   DMSLDR   DMSLOA   DMSLSB   DMSMOD
                     DMSOLD   DMSSET   DMSSLN   DMSSMN   DMSSTG
AUSRILST  000008     DMSABN   DMSHDI
AUSRITBL  000007     DMSABN   DMSHDI
AUTOCNT   000005     DMSEDI
AUTOCURR  000003     DMSEDI
AUTOREG   000002     DMSEDI
AVIPWORK  000009     DMSVIB   DMSVIP   DMSVSR
AVSAMSYS  000007     DMSBOP   DMSCLS   DMSDOS   DMSVIB   DMSVIP   DMSVSR
AVSREOJ   000001     DMSDOS
AVSRWORK  000005     DMSCLS   DMSVSR
AWAIT     000001     DMSITS
AWRBUF    000036     DMSDLK   DMSDSK   DMSEDI   DMSLBT   DMSLIO   DMSLLU   DMSMOD   DMSOLD   DMSPRV   DMSRDC   DMSRNE   DMSRRV
                     DMSSRV   DMSSVT   DMSTPE   DMSUPD   DMSXCP
AWRTK     000005     DMSAUD   DMSBWR   DMSFNS   DMSFOR
BALR      000239     DMSABN   DMSACC   DMSACF   DMSACM   DMSALU   DMSAMS   DMSAUD   DMSBOP   DMSBRD   DMSBWR   DMSCAT   DMSCIT
                     DMSCLS   DMSCMP   DMSCRD   DMSCWR   DMSDIO   DMSDLE   DMSDMP   DMSDOS   DMSEDX   DMSERS   DMSEXC   DMSEXT
```

LABEL	COUNT	REFERENCES
		DMSFCH DMSFET DMSFNS DMSFOR DMSFRE DMSHDI DMSHDS DMSINS DMSINT DMSITE DMSITP DMSITS
		DMSLAD DMSLAF DMSLDR DMSLFS DMSLGT DMSLIB DMSLSB DMSMCD DMSOLD DMSOPL DMSOR1 DMSROS
		DMSSAB DMSSET DMSSLN DMSSOP DMSSTG DMSSVN DMSSVT DMSVSR DMSXCP
BALRSAVE	000027	DMSCPF DMSDBG DMSFNS DMSINA DMSINM DMSSCN DMSSMN DMSSTG DMSVIB
BALR12	000002	DMSSTT
BALR14	000002	DMSITI
BALR9	000001	DMSBRD
BATCPEX	000006	DMSARE DMSBTP DMSCPF
BATCPUC	000002	DMSITE
BATCPUL	000001	DMSITE
BATDCMS	000009	DMSASN DMSBTB DMSBTP DMSDSK DMSFLD DMSRDC DMSSET
BATFLAGS	000065	DMSABN DMSARE DMSARN DMSASN DMSBTB DMSBTP DMSCIO DMSCPF DMSCRD DMSDSK DMSERR DMSFLD
		DMSFRE DMSINS DMSITE DMSLDR DMSLSB DMSMVE DMSOLD DMSPIO DMSRDC DMSSET
BATFLAG2	000020	DMSABN DMSASN DMSBTB DMSBTP DMSCIT DMSDSK DMSERR DMSFLD DMSINS DMSITE DMSRDC DMSSET
BATIPLSS	000001	DMSINS
BATLOAD	000016	DMSABN DMSARE DMSBTB DMSCPF DMSCRD DMSFRE DMSINS DMSITE DMSLDR DMSLSB DMSOLD
BATLSECT	000003	DMSCIO DMSITE DMSPIO
BATMOVE	000007	DMSBTP DMSMVE
BATNOEX	000010	DMSBTB DMSBTP DMSCIO DMSPIO DMSSET
BATPRTC	000002	DMSPIO
BATPRTL	000001	DMSPIO
BATPUNC	000002	DMSCIO
BATPUNL	000001	DMSCIO
BATRERR	000003	DMSBTP
BATRUN	000026	DMSABN DMSARE DMSARN DMSASN DMSBTB DMSCIO DMSCPF DMSCRD DMSDSK DMSERR DMSFLD DMSINS
		DMSITE DMSPIO DMSRDC DMSSET
BATSTOP	000002	DMSBTP DMSCIT
BATSYSAB	000004	DMSABN DMSERR
BATTERM	000005	DMSBTP
BATUSEX	000006	DMSARE DMSBTB DMSBTP DMSCPF DMSITE
BATXCPU	000002	DMSBTP DMSITE
BATXLIM	000005	DMSBTP DMSCIO DMSITE DMSPIO
BATXPRT	000002	DMSBTP DMSPIO
BATXPUN	000001	DMSCIO
BDISK	000001	DMSNUC
BEGAT	000003	DMSDBG
BGCOM	000051	DMSAMS DMSASN DMSBAB DMSBOP DMSCLS DMSDLB DMSDLK DMSCMP DMSDOS DMSDSV DMSFCH DMSFET
		DMSINS DMSITP DMSLLU DMSOPL DMSOPT DMSPRV DMSQRY DMSRRV DMSSET DMSSMN DMSSRV DMSSTG
		DMSVSR DMSXCP
BITS	000009	DMSDBG DMSPRT DMSPUN
BLANKS	000059	DMSBOP DMSCPY DMSDBD DMSDLK DMSDSK DMSDSV DMSEXT DMSGRN DMSINI DMSLBT DMSLDR DMSLLU
		DMSOLD DMSQRY DMSRRV DMSSYN DMSUPD DMSVIB DMSZAP
BLANK1	000001	DMSEDX
BLANK2	000002	DMSDSV DMSEDX
BLANK3	000001	DMSEDX
BLK	000015	DMSBTP DMSSEB DMSSOP DMSSQS DMSTMA DMSTPD

```
LABEL     COUNT      REFERENCES

BLOC      000006     DMSEDI   DMSEDX
BLOCKLEN  000010     DMSFRE
BRAD      000021     DMSLDR   DMSLSB   DMSLST   DMSOLD
BRKPNTBL  000003     DMSDBG
BS        000001     DMSCPF
BSR       000012     DMSBOP   DMSCLS   DMSTPE
BUFAD     000009     DMSCPY
BUFFA     000013     DMSOVS   DMSUPD
BUFFER    000163     DMSBOP   DMSCLS   DMSDLK   DMSDSK   DMSDSL   DMSEDX   DMSEXT   DMSGLB   DMSIFC   DMSLBM   DMSLBT   DMSOPL
                     DMSOVR   DMSPRV   DMSRDC
BUFFLOC   000001     DMSSCR
BUFSIZE   000008     DMSEXT   DMSZAP
BUSOUT    000001     DMSFCH
BUSY      000002     DMSCIO   DMSPIO
BYTE      000004     DMSEDI   DMSNCP
CALLEE    000026     DMSERR   DMSITP   DMSITS   DMSLDR   DMSOVS   DMSSAB   DMSVIP
CALLER    000009     DMSDOS   DMSFRE   DMSITS   DMSOVS   DMSSVT   DMSXCP
CARDINCR  000003     DMSEDI   DMSEDX
CARDNO    000003     DMSEDI
CASEREAD  000001     DMSEDI
CASESW    000006     DMSEDI   DMSEDX
CAW       000016     DMSCIO   DMSCIT   DMSDBD   DMSDBG   DMSDIO   DMSERR   DMSINI   DMSINS   DMSPIO
CC        000309     DMSARX   DMSASM   DMSBOP   DMSFCH   DMSFOR   DMSINI   DMSINS   DMSLDS   DMSPIO   DMSPRT   DMSPRV   DMSROS
                     DMSRRV   DMSSET   DMSSRV   DMSTIO   DMSXCP
CCBCCW    000004     DMSXCP
CCBCNT    000017     DMSXCP
CCBCOM1   000004     DMSXCP
CCBCOM2   000012     DMSXCP
CCBCSW    000003     DMSXCP
CCBCSW1   000007     DMSXCP
CCBCSW2   000004     DMSXCP
CCBDC     000001     DMSXCP
CCBEOC    000006     DMSXCP
CCBEOF    000004     DMSXCP
CCBERMAP  000017     DMSXCP
CCBILEN   000004     DMSXCP
CCBNOREC  000001     DMSXCP
CCBSUCLS  000002     DMSXCP
CCBSUNUM  000002     DMSXCP
CCBSYMU   000002     DMSXCP
CCBUE     000006     DMSXCP
CCBVER    000006     DMSXCP
CCPADDR   000001     DMSNCP
CCPARM    000004     DMSNCP
CCPCAONE  000003     DMSNCP
CCPENTRY  000001     DMSNCP
```

LABEL	COUNT	REFERENCES					
CCPHBFNO	000003	DMSNCP					
CCPHBFSZ	000003	DMSNCP					
CCPMAXID	000001	DMSNCP					
CCPNAME	000001	DMSNCP					
CCPPAD0	000003	DMSNCP					
CCPPAD1	000003	DMSNCP					
CCPPSIZE	000003	DMSNCP					
CCPRESID	000006	DMSNCP					
CCPRSTAT	000006	DMSNCP					
CCPRSTEP	000003	DMSNCP					
CCPRSTYP	000009	DMSNCP					
CCPSIZE	000001	DMSNCP					
CCPSTOR	000001	DMSNCP					
CCPTEP	000001	DMSNCP					
CCPTEP4	000001	DMSNCP					
CCPTNCP	000001	DMSNCP					
CCPTPEP	000003	DMSNCP					
CCPTYPE	000007	DMSNCP					
CCPTYPE1	000002	DMSNCP					
CCPTYPE2	000001	DMSNCP					
CCPVPAD0	000001	DMSNCP					
CCPVPAD1	000001	DMSNCP					
CCWPRINT	000017	DMSDBD					
CCWX	000002	DMSDIO					
CCW1	000006	DMSDIO					
CCW1A	000004	DMSDIO					
CCW2	000003	DMSDIO	DMSOR3				
CD	000002	DMSXCP					
CDISK	000006	DMSNUC	DMSPRV	DMSQRY	DMSRRV	DMSSOP	DMSSRV
CDMSROS	000006	DMSACM	DMSALU				
CE	000004	DMSCIT	DMSINI				
CHAN0	000002	DMSINI	DMSINS				
CHGTRUNC	000002	DMSEDI					
CHKWRD1	000002	DMSITS					
CHKWRD2	000002	DMSITS					
CHNGBYTE	000010	DMSSBS	DMSSVT				
CHNGCNT	000003	DMSEDI					
CHNGFLAG	000021	DMSEDI	DMSSCR				
CHNGMSG	000003	DMSEDI	DMSEDX				
CHNGNUM	000005	DMSEDI					
CL	000003	DMSCPY	DMSFRE	DMSTPE			
CLASDASD	000002	DMSASN	DMSINI				
CLASTAPE	000002	DMSASN	DMSTPE				
CLASTERM	000002	DMSEDX	DMSINI				
CLASURI	000002	DMSASN	DMSRDC				
CLASURO	000004	DMSASN	DMSPRT	DMSPUN			

```
LABEL     COUNT      REFERENCES

CLEAROP   000004     DMSLSB
CLKVALMD  000005     DMSDOS    DMSFNS    DMSINS
CLOSELIB  000016     DMSLDR    DMSLIB    DMSOLD    DMSZAP
CLOSIO    000003     DMSPRT    DMSPUN    DMSRDC
CMD       000006     DMSLDR    DMSOLD
CMDBLOK   000002     DMSEDX    DMSGIO
CMDREJ    000001     DMSFCH
CMNDLINE  000013     DMSABN    DMSARX    DMSASM    DMSCPF    DMSINS    DMSINT    DMSSEB    DMSSVT
CMNDLIST  000025     DMSCAT    DMSCPF    DMSINS    DMSLDR    DMSOLD    DMSSCN
CMODE     000019     DMSEDI
CMSAMS    000005     DMSAMS    DMSVSR
CMSCVT    000003     DMSINS    DMSSOP    DMSVSR
CMSDOS    000002     DMSSET
CMSNAME   000002     DMSSOP    DMSSVT
CMSOP     000016     DMSDLB    DMSSCT    DMSSOP    DMSSVT
CMSSEG    000018     DMSBTP    DMSEDX    DMSEXC    DMSINS    DMSINT    DMSITS    DMSQRY    DMSSET
CMSTAXE   000007     DMSCIT    DMSITE    DMSITI    DMSSVT
CMSTIM    000007     DMSINT
CMSVSAM   000011     DMSBOP    DMSDOS    DMSSET    DMSVIB    DMSVSR
CODE      000014     DMSCPY    DMSITS    DMSLKD    DMSNCP    DMSSET
CODE203   000210     DMSABN    DMSACC    DMSACF    DMSACM    DMSALU    DMSAMS    DMSAUD    DMSBOP    DMSBRD    DMSBWR    DMSCAT    DMSCIT
                     DMSCLS    DMSCMP    DMSCRD    DMSCWR    DMSDIO    DMSDLB    DMSDMP    DMSDOS    DMSEDX    DMSERS    DMSEXC    DMSEXT
                     DMSFCH    DMSFET    DMSFNS    DMSFOR    DMSFRE    DMSHDI    DMSHDS    DMSINS    DMSINT    DMSITE    DMSITP    DMSITS
                     DMSLAD    DMSLAF    DMSLDR    DMSLFS    DMSLGT    DMSLIB    DMSLSB    DMSMCD    DMSOLD    DMSOPL    DMSOR1    DMSSAB
                     DMSSET    DMSSLN    DMSSOP    DMSSTG    DMSSVN    DMSSVT    DMSVSR    DMSXCP
COMMONEX  000006     DMSLDR    DMSOLD
COMNAME   000015     DMSAMS    DMSBOP    DMSDLK    DMSDOS    DMSDSV    DMSFCH    DMSFET    DMSLST    DMSZAP
COMPSWT   000016     DMSARN    DMSARX    DMSASM    DMSIFC    DMSSLN    DMSSMN    DMSSTG
CONCCWS   000008     DMSCIT    DMSERR
CONCNT    000003     DMSARX    DMSASM    DMSLDS
CONDFLG   000011     DMSEXT
CONFLAG   000002     DMSMVE
CONHCT    000004     DMSDBD    DMSDBG    DMSITE    DMSNUC
CONHXT    000002     DMSDBG
CONINBLK  000004     DMSCRD
CONINBUF  000005     DMSCRD
CONRDBUF  000001     DMSSVN
CONRDCNT  000007     DMSABN    DMSINS    DMSINT    DMSSEB    DMSSVN    DMSSVT
CONRDCOD  000007     DMSABN    DMSINS    DMSINT    DMSSEB    DMSSVN
CONREAD   000009     DMSABN    DMSDLB    DMSFLD    DMSFNC    DMSINS    DMSINT    DMSSEB    DMSSVN    DMSSVT
CONSOLE   000020     DMSBOP    DMSCWR    DMSEDI    DMSEDX    DMSINI    DMSINS    DMSOR3    DMSSEB    DMSZAP
CONSTACK  000008     DMSCIT    DMSCWR    DMSSVN
CONWR     000005     DMSARX    DMSASM    DMSDBG    DMSSEB    DMSXCP
CONWRBUF  000005     DMSINT    DMSSEB    DMSSVN    DMSSVT
CONWRCNT  000004     DMSSEB    DMSSVN    DMSSVT
CONWRCOD  000008     DMSINT    DMSSEB    DMSSVN
```

```
LABEL     COUNT     REFERENCES


CONWRITE 000005    DMSINT   DMSSEB   DMSSVN   DMSSVT
CONWRL   000001    DMSDBG
CORESIZE 000009    DMSSTG   DMSSVT
CORITEM  000007    DMSEDI   DMSEDX   DMSUPD
COUNT    000080    DMSDBG   DMSDSK   DMSEDI   DMSTQQ
CPSTAT   000001    DMSCLS
CPULOG   000005    DMSDBD   DMSSET
CRBIT    000002    DMSEDI
CRDPTR   000006    DMSLDR   DMSOLD
CSW      000055    DMSCIO   DMSCIT   DMSCRD   DMSCWR   DMSDBG   DMSDIC   DMSDLK   DMSFCH   DMSGIO   DMSINI   DMSIOW   DMSITE
                   DMSITI   DMSLDS   DMSPIO   DMSROS   DMSTIO   DMSTMA   DMSTPD   DMSXCP
CTL      000002    DMSUPD
CUE      000003    DMSUPD
CURRALOC 000013    DMSITS
CURRCPUT 000001    DMSINM
CURRDATE 000006    DMSEXT   DMSINM   DMSINS   DMSSET   DMSSVT
CURRIOOP 000003    DMSCIT
CURRSAVE 000061    DMSABN   DMSACC   DMSDBG   DMSDLB   DMSDOS   DMSERR   DMSFLD   DMSFRE   DMSIFC   DMSITP   DMSITS   DMSLDR
                   DMSOVS   DMSSAB   DMSSLN   DMSSMN   DMSSOP   DMSSTG   DMSSVN   DMSSVT   DMSVIP
CURRTIME 000001    DMSEXT
CURRVIRT 000002    DMSINM
CVTAVIB  000002    DMSSOP   DMSVSR
CVTMDL   000001    DMSINS
CVTMZ00  000001    DMSINS
CVTNUCB  000001    DMSINS
CVTOPTA  000001    DMSINS
CVTSECT  000001    DMSINS
C0       000002    DMSDLK
C1       000001    DMSCWR
C12      000001    DMSLDR
C7       000002    DMSLDR
C9       000001    DMSLDR
DA       000021    DMSDSL   DMSMVE   DMSNCP   DMSSBD   DMSSBS   DMSSCT   DMSSOP
DACTIVE  000010    DMSDOS   DMSFCH   DMSFET
DATACHK  000002    DMSFCH   DMSXCP
DATAEND  000015    DMSSBD   DMSSVT
DATE     000016    DMSDLK   DMSLST   DMSSVT   DMSUPD
DATIPCMS 000007    DMSDOS   DMSFNS   DMSINS
DBDDMSG  000003    DMSDBD
DBDEXIT  000003    DMSDBD
DBGABN   000005    DMSABN   DMSDBG
DBGEXEC  000005    DMSABN   DMSCIT   DMSDBG   DMSITE
DBGEXINT 000008    DMSCIT   DMSDBG   DMSIOW   DMSITE
DBGFLAGS 000040    DMSABN   DMSCIT   DMSDBD   DMSDBG   DMSIOW   DMSITE
DBGNSHR  000001    DMSABN
DBGOUT   000034    DMSDBD   DMSDBG   DMSITE   DMSNUC
```

```
LABEL    COUNT     REFERENCES


DBGPGMCK 000004    DMSDBG
DBGRECUR 000017    DMSDBD   DMSDBG
DBGSAV1  000002    DMSDBG
DBGSAV2  000001    DMSDBG
DBGSECT  000007    DMSDBD   DMSDBG   DMSITE
DBGSET   000003    DMSDBG
DBGSHR   000001    DMSABN
DBGSWTCH 000012    DMSDBD   DMSDBG
DCBSAV   000003    DMSSOP
DCSSAVAL 000014    DMSEDX   DMSEXC   DMSINS   DMSITS   DMSSAB   DMSSET
DCSSFLAG 000043    DMSABN   DMSEDX   DMSEXC   DMSINS   DMSINT   DMSITS   DMSSAB   DMSSET
DCSSJLNS 000004    DMSINT   DMSSET
DCSSLDED 000010    DMSEDX   DMSEXC   DMSINT   DMSITS   DMSSET
DCSSOVLP 000001    DMSINS
DCSSVTLD 000018    DMSABN   DMSINS   DMSITS   DMSSAB   DMSSET
DDISK    000003    DMSINS   DMSNUC
DDNAM    000001    DMSMVE
DE       000006    DMSCIO   DMSCIT   DMSCLS   DMSINI
DEBDCBAD 000002    DMSSAB   DMSSOP
DEBDEBID 000001    DMSSOP
DEBOPATB 000002    DMSSOP
DEBTCBAD 000004    DMSSQS
DEC      000074    DMSBOP   DMSDBD   DMSDBG   DMSDLK   DMSDSK   DMSDSV   DMSEDI   DMSEDX   DMSLIB   DMSLST   DMSOVR   DMSQRY
                   DMSSET   DMSSRT   DMSSSK   DMSTPD   DMSTPE   DMSVIB   DMSVPD   DMSZAP
DECAREA  000007    DMSSBD   DMSSBS
DECDCBAD 000002    DMSSBS   DMSSCT
DECDEC   000038    DMSDBD   DMSDBG   DMSITE   DMSNUC   DMSQRY
DECIMAL  000009    DMSEDI
DECIOBPT 000003    DMSSBS   DMSSCT
DECKYADR 000004    DMSSBD
DECLNGTH 000005    DMSSBD   DMSSBS
DECLTH   000002    DMSSCR
DECRECPT 000002    DMSSBD
DECSDECB 000024    DMSSBD   DMSSBS   DMSSCT   DMSSVT
DECTYPE  000025    DMSSBD   DMSSBS
DEPTH    000007    DMSITS   DMSOVS
DEVADDR  000048    DMSTIO   DMSTPE
DEVCODE  000002    DMSBOP   DMSCLS
DEVICE   000004    DMSARX   DMSASM   DMSIOW   DMSITI
DEVMISC  000005    DMSTIO   DMSTPE
DEVNAME  000003    DMSTIO   DMSTPE
DEVSECT  000005    DMSTIO   DMSTPE
DEVSIZE  000003    DMSTIO   DMSTPE
DEVTAB   000011    DMSASN   DMSDBD   DMSEDI   DMSEDX   DMSINI   DMSLLU   DMSSVT
DEVTYP   000027    DMSDIO   DMSFNS   DMSLLU   DMSSOP
DEVTYPE  000025    DMSRDC   DMSSVT
```

LABEL	COUNT	REFERENCES
DIAGNUM	000001	DMSDIO
DIAGRET	000003	DMSDIO
DIAGTIME	000001	DMSSVT
DIOBIT	000003	DMSDIO
DIOCSW	000001	DMSFNS
DIOFLAG	000009	DMSDIO
DIOFREE	000003	DMSDIO
DIOSECT	000007	DMSACM DMSDIO DMSFNS DMSITI
DIRAAA	000001	DMSFCH
DIRC	000017	DMSDOS DMSFCH
DIREEE	000001	DMSFCH
DIRLL	000004	DMSDOS DMSFCH
DIRN	000006	DMSDOS DMSFCH DMSFET
DIRNAME	000039	DMSDOS DMSDSL DMSFCH DMSFET DMSGND DMSSVT
DIRPPP	000003	DMSFCH
DIRPTR	000007	DMSSVT
DIRR	000001	DMSDSL
DIRRR	000001	DMSFCH
DIRTT	000005	DMSDOS DMSDSL DMSFCH
DIRTTR	000002	DMSFCH
DISK$SEG	000008	DMSBRD DMSFNS DMSLFS
DITCNT	000005	DMSEDI
DMPTITLE	000003	DMSDBG
DMSABNGO	000005	DMSFRE DMSITI DMSITP DMSITS
DMSABNRT	000001	DMSDBG
DMSABNSV	000001	DMSFNC
DMSABW	000011	DMSABN DMSDBG DMSFRE DMSITI DMSITP DMSITS
DMSARD	000001	DMSARX
DMSASD	000001	DMSASM
DMSBWR	000002	DMSFNC
DMSCAT	000004	DMSABN DMSCRD DMSFNC
DMSCCB	000002	DMSXCP
DMSCIOSI	000002	DMSFNC
DMSCITA	000001	DMSCWR
DMSCITB	000002	DMSCRD DMSCWR
DMSCITDB	000003	DMSABN DMSFNC
DMSCPF	000003	DMSFNC DMSINT
DMSCRD	000005	DMSABN DMSFNC
DMSCWR	000005	DMSDBG DMSERR DMSFNC DMSITE
DMSCWT	000006	DMSABN DMSDBG DMSERR DMSFNC DMSITS
DMSDBD	000001	DMSDBG
DMSDBG	000014	DMSABN DMSFNC DMSINS DMSINT DMSIOW DMSITE DMSNUC DMSQRY DMSSET DMSSMN DMSSTG DMSSVN DMSSVT
DMSDBGP	000001	DMSINI
DMSEDC	000001	DMSSEG
DMSEDI	000001	DMSSEG

```
LABEL     COUNT     REFERENCES

DMSERR    000086    DMSABN   DMSBWR   DMSCIT   DMSCRD   DMSCWR   DMSDBG   DMSERS   DMSFET   DMSFNC   DMSFNS   DMSFRE   DMSITP
                    DMSITS   DMSLIO   DMSMOD   DMSSTT
DMSERT    000002    DMSERR
DMSEXC    000002    DMSFNC
DMSEXCAB  000001    DMSABN
DMSEXT    000001    DMSSEG
DMSFCH    000003    DMSDOS
DMSFET    000002    DMSFNC
DMSFNC    000001    DMSITS
DMSFNC3   000001    DMSITS
DMSFREB   000002    DMSFNC
DMSFREES  000002    DMSFNC
DMSFREEX  000002    DMSFNC
DMSFRES   000005    DMSABN   DMSFNC   DMSINS
DMSFRETS  000002    DMSFNC
DMSFRETX  000001    DMSFNC
DMSFRT    000002    DMSFRE
DMSGIO    000002    DMSSCR   DMSSEG
DMSINALT  000001    DMSNUC
DMSINA1S  000001    DMSNUC
DMSINS    000001    DMSINI
DMSINSE   000001    DMSINI
DMSINTAB  000001    DMSABN
DMSIOWR   000001    DMSDBG
DMSITET   000002    DMSFNC
DMSITP    000001    DMSDBG
DMSITSK   000001    DMSFNC
DMSITSR   000001    DMSABN
DMSITSXS  000001    DMSFNC
DMSITS1   000001    DMSINI
DMSLAD    000005    DMSBWR   DMSERS   DMSINS   DMSLFS   DMSSTT
DMSLADAD  000003    DMSABN   DMSFNC
DMSLADN   000003    DMSABN   DMSLFS
DMSLADW   000002    DMSERS   DMSSTT
DMSLDRA   000002    DMSFNC
DMSLDRB   000001    DMSLOA
DMSLDRC   000001    DMSLSB
DMSLDRD   000003    DMSLGT   DMSLIB   DMSLSB
DMSLFS    000005    DMSBRD   DMSEXC   DMSINT   DMSPNT   DMSSTT
DMSLFSW   000005    DMSEWR   DMSERS   DMSFNS   DMSSTT
DMSLGT    000002    DMSSEG   DMSSVT
DMSLGTA   000003    DMSLDR   DMSOLD   DMSSTG
DMSLGTB   000002    DMSLDR   DMSOLD
DMSLIB    000004    DMSLDR   DMSOLD   DMSSEG   DMSTMA
DMSLIO    000001    DMSLDR
DMSLOA    000005    DMSFNC   DMSINS
```

LABEL	COUNT	REFERENCES			
DMSLSB	000002	DMSSEG	DMSSVT		
DMSLSBA	000002	DMSLDR	DMSOLD		
DMSLSBB	000002	DMSLDR	DMSOLD		
DMSLSBC	000002	DMSLDR	DMSOLD		
DMSLSBD	000002	DMSLDR	DMSOLD		
DMSLSY	000003	DMSLDR	DMSOLD	DMSSEG	
DMSMOD	000005	DMSFNC	DMSITS		
DMSNUCU	000001	DMSFRE			
DMSOLD	000002	DMSSEG	DMSSLN		
DMSOVS	000001	DMSOVR			
DMSPIO	000002	DMSFNC			
DMSPIOCC	000002	DMSFNC			
DMSPIOSI	000002	DMSFNC			
DMSREA	000002	DMSIFC			
DMSSAB	000004	DMSSEG	DMSSVT		
DMSSBD	000002	DMSSBS	DMSSEG		
DMSSBDFR	000001	DMSSVT			
DMSSBS	000004	DMSSBD	DMSSEG	DMSSOP	DMSSVT
DMSSBSRT	000001	DMSSBD			
DMSSCNN	000002	DMSINS	DMSINT		
DMSSCR	000002	DMSEDI	DMSSEG		
DMSSCT	000002	DMSSEG	DMSSVT		
DMSSCTCE	000002	DMSSOP	DMSSQS		
DMSSCTCK	000003	DMSSOP	DMSSQS		
DMSSCTNP	000001	DMSSOP			
DMSSEB	000005	DMSSBS	DMSSEG	DMSSQS	
DMSSLN	000002	DMSSEG	DMSSVT		
DMSSLN3	000002	DMSSVT			
DMSSLN42	000002	DMSSVT			
DMSSLN6	000002	DMSSVT			
DMSSLN7	000002	DMSSVT			
DMSSLN8	000002	DMSSVT			
DMSSLN9	000002	DMSSVT			
DMSSMN	000002	DMSSEG	DMSSVT		
DMSSMNSB	000001	DMSSLN			
DMSSMN10	000002	DMSSVT			
DMSSMN4	000002	DMSSVT			
DMSSMN5	000002	DMSSVT			
DMSSOP	000002	DMSSEG	DMSSVT		
DMSSOP19	000002	DMSSVT			
DMSSOP20	000002	DMSSVT			
DMSSOP22	000002	DMSSVT			
DMSSOP23	000002	DMSSVT			
DMSSQS	000002	DMSSEG	DMSSVT		
DMSSQSGT	000001	DMSSOP			
DMSSQSPT	000001	DMSSOP			

```
LABEL     COUNT      REFERENCES

DMSSQSUP 000001     DMSSOP
DMSSTGAT 000002     DMSFNC
DMSSTGCL 000001     DMSFNC
DMSSTGSB 000005     DMSABN    DMSFNC    DMSINT    DMSLDR    DMSMOD
DMSSTGSV 000003     DMSFNC
DMSSTTR  000001     DMSLFS
DMSSVN   000002     DMSSEG    DMSSVT
DMSSVN1  000002     DMSSVT
DMSSVN2  000002     DMSSVT
DMSSVN93 000002     DMSSVT
DMSSVN94 000002     DMSSVT
DMSSVT   000001     DMSSEG
DMSVSR   000002     DMSFNC
DMSXCP   000001     DMSDOS
DOSBLKSZ 000005     DMSBOP
DOSBUFF  000012     DMSBOP    DMSXCP
DOSBUFSP 000004     DMSDLB    DMSQRY    DMSXCP
DOSBYTE  000014     DMSXCP
DOSCBID  000002     DMSDLB    DMSXCP
DOSCMS   000002     DMSDLB
DOSCOMP  000005     DMSFET    DMSLDR
DOSCOUT  000002     DMSXCP
DOSDD    000027     DMSAMS    DMSBOP    DMSCLS    DMSDLB    DMSDLK    DMSDSV    DMSOPL    DMSQRY    DMSRRV    DMSSRV    DMSSVT    DMSVIP
                    DMSXCP
DOSDDCAT 000006     DMSDLB
DOSDEV   000018     DMSAMS    DMSBOP    DMSDLB    DMSDLK    DMSQRY    DMSRRV    DMSSRV    DMSVIP    DMSXCP
DOSDIRC  000005     DMSSOP    DMSSVT
DOSDOS   000004     DMSDLB    DMSQRY
DOSDSK   000006     DMSDLB    DMSDLK    DMSEXT    DMSRRV    DMSSRV    DMSXCP
DOSDSMD  000027     DMSAMS    DMSBOP    DMSDLB    DMSVIP    DMSXCP
DOSDSNAM 000009     DMSCLS    DMSDLB    DMSQRY    DMSXCP
DOSDSTYP 000004     DMSCLS    DMSDLB    DMSQRY    DMSXCP
DOSDUM   000013     DMSAMS    DMSBOP    DMSDLB    DMSQRY    DMSVIP    DMSXCP
DOSEND   000001     DMSDLB
DOSENSIZ 000006     DMSDLB
DOSEXT   000004     DMSBOP
DOSEXTCT 000002     DMSBOP
DOSEXTCX 000004     DMSXCP
DOSEXTNO 000013     DMSAMS    DMSDLB    DMSQRY    DMSVIP    DMSXCP
DOSEXTTB 000009     DMSAMS    DMSDLB    DMSQRY    DMSVIP    DMSXCP
DOSFIRST 000027     DMSABN    DMSAMS    DMSBOP    DMSCLS    DMSDLB    DMSDLK    DMSDSV    DMSFCH    DMSOPL    DMSQRY    DMSROS    DMSRRV
                    DMSSRV    DMSSVT    DMSVIP    DMSXCP
DOSFLAGS 000161     DMSABN    DMSALU    DMSAMS    DMSASM    DMSASN    DMSBOP    DMSCPY    DMSDLB    DMSDLK    DMSDOS    DMSDSL    DMSDSV
                    DMSEDI    DMSEDX    DMSEXT    DMSFCH    DMSFET    DMSHDI    DMSHDS    DMSIFC    DMSINT    DMSITE    DMSITP    DMSITS
                    DMSLDR    DMSLDS    DMSLLU    DMSMOD    DMSMVE    DMSOPT    DMSPIO    DMSPRV    DMSQRY    DMSROS    DMSRRV    DMSSET
                    DMSSRT    DMSSRV    DMSSTG    DMSTPD    DMSUPD    DMSVIP    DMSVSR    DMSXCP    DMSZAP
```

```
LABEL     COUNT     REFERENCES


DOSFORM   000009    DMSBOP   DMSXCP
DOSINIT   000027    DMSBOP   DMSDLB   DMSQRY   DMSXCP
DOSITEM   000008    DMSXCP
DOSJCAT   000006    DMSDLB
DOSKPART  000006    DMSFCH   DMSQRY   DMSSET   DMSSTG
DOSLBSV   000004    DMSGLB
DOSLIBL   000007    DMSFCH   DMSGLB   DMSQRY   DMSSOP   DMSSVT
DOSMODE   000041    DMSABN   DMSALU   DMSAMS   DMSASN   DMSDLB   DMSDLK   DMSDSV   DMSEXT   DMSFET   DMSINT   DMSITP   DMSLDR
                    DMSLLU   DMSMOD   DMSOPT   DMSPRV   DMSQRY   DMSRRV   DMSSET   DMSSRV   DMSVSR
DOSNEXT   000011    DMSAMS   DMSBOP   DMSCLS   DMSDLB   DMSOPL   DMSSVT   DMSVIP   DMSXCP
DOSNUM    000014    DMSABN   DMSBOP   DMSDLB   DMSQRY   DMSXCP
DOSOP     000037    DMSBOP   DMSDLK   DMSRRV   DMSSRV   DMSXCP
DOSOS     000006    DMSDLB   DMSQRY
DOSOSDSN  000008    DMSDLB   DMSQRY   DMSXCP
DOSOSFST  000009    DMSBOP   DMSDLB   DMSDLK   DMSRRV   DMSSRV   DMSXCP
DOSPERM   000004    DMSDLB   DMSQRY
DOSRC     000015    DMSAMS   DMSBAB   DMSBOP   DMSDOS   DMSFET   DMSLDR   DMSVIP
DOSREAD   000010    DMSFCH   DMSXCP
DOSSAVE   000009    DMSIFC   DMSXCP
DOSSECT   000029    DMSAMS   DMSBOP   DMSCLS   DMSDLB   DMSDLK   DMSDSV   DMSOPL   DMSQRY   DMSRRV   DMSSRV   DMSSVT   DMSVIP
                    DMSXCP
DOSSENSE  000008    DMSXCP
DOSSVC    000057    DMSABN   DMSAMS   DMSASM   DMSCPY   DMSDLB   DMSDLK   DMSDSL   DMSEDI   DMSEDX   DMSEXT   DMSFCH   DMSFET
                    DMSHDI   DMSHDS   DMSIFC   DMSINT   DMSITE   DMSITP   DMSITS   DMSLDR   DMSLDS   DMSMOD   DMSMVE   DMSQRY
                    DMSROS   DMSSET   DMSSRT   DMSTPD   DMSUPD   DMSVIP   DMSVSR   DMSZAP
DOSSYS    000004    DMSBOP   DMSDLB   DMSOPL   DMSQRY
DOSTAPID  000002    DMSXCP
DOSTRANS  000013    DMSABN   DMSBOP   DMSCLS   DMSDOS   DMSFCH   DMSSET
DOSTYPE   000011    DMSDLB   DMSQRY   DMSXCP
DOSUCAT   000006    DMSBOP   DMSDLB
DOSUCNAM  000011    DMSBOP   DMSDLB   DMSQRY   DMSXCP
DOSVOLNO  000015    DMSAMS   DMSDLB   DMSQRY   DMSVIP   DMSXCP
DOSVOLTB  000009    DMSAMS   DMSDLB   DMSQRY   DMSVIP   DMSXCP
DOSVSAM   000010    DMSASN   DMSBOP   DMSDOS   DMSFCH   DMSSET   DMSSTG
DOSWORK   000006    DMSXCP
DOSXXX    000002    DMSDLB   DMSQRY
DOSYSXXX  000015    DMSAMS   DMSBOP   DMSCLS   DMSDLB   DMSVIP   DMSXCP
DOUBLE    000017    DMSBOP   DMSCLS   DMSDIO   DMSDLB   DMSLBM   DMSLBT
DSKAD     000002    DMSLIO
DSKADR    000006    DMSACF   DMSACM   DMSAUD   DMSERS
DSKLIN    000066    DMSEXT   DMSLIO   DMSMOD   DMSSLN
DSKLOC    000010    DMSACF   DMSACM   DMSAUD   DMSERS   DMSFNS   DMSMOD
DSKLST    000021    DMSACF   DMSACM   DMSAUD   DMSERS   DMSFNS   DMSLLU   DMSMOD   DMSPRV   DMSRRV   DMSSRV
DSYM      000002    DMSLSY
DTAD      000034    DMSACC   DMSACM   DMSAMS   DMSARE   DMSASN   DMSDIC   DMSFOR   DMSINS   DMSQRY   DMSROS
DTADT     000018    DMSACM   DMSASN   DMSAUD   DMSDIO   DMSQRY   DMSTQQ
```

LABEL	COUNT	REFERENCES										
DTAS	000003	DMSAMS										
DUALNOS	000008	DMSEDC										
DUMCOM	000004	DMSITS	DMSSLN									
DUMMY	000020	DMSASM	DMSFLD	DMSQRY	DMSSBD	DMSSEB	DMSVPD					
DUMPLIST	000002	DMSDBG	DMSSVT									
DYLD	000012	DMSLDR	DMSLIO	DMSOLD	DMSSLN	DMSSTG						
DYLIBO	000004	DMSSLN	DMSSTG									
DYMBRNM	000005	DMSLIB	DMSSLN	DMSSTG								
DYNAEND	000004	DMSLDR	DMSOLD	DMSSLN								
EDCB	000005	DMSEDC	DMSEDI	DMSEDX	DMSGIO	DMSSCR						
EDCBEND	000001	DMSEDX										
EDCBLTH	000002	DMSEDX										
EDCT	000026	DMSEDI										
EDISK	000002	DMSNUC										
EDIT	000066	DMSBTP	DMSDLB	DMSEDI	DMSIFC	DMSINA	DMSQRY	DMSVPD				
EDLIN	000013	DMSEDI	DMSEDX									
EDMSK	000003	DMSSCR										
EDRET	000003	DMSEDI	DMSEDX									
EDWORK	000002	DMSEDX										
EFPRS	000008	DMSITS	DMSOVS	DMSSVT								
EGPRS	000019	DMSABN	DMSITS	DMSOVS	DMSSAB	DMSSLN						
EGPR0	000064	DMSACC	DMSDLB	DMSDOS	DMSFLD	DMSITS	DMSOVS	DMSSAB	DMSSLN	DMSSOP	DMSSVN	DMSSVT
EGPR1	000039	DMSDOS	DMSLDR	DMSSAB	DMSSLN	DMSSMN	DMSSOP	DMSSVN	DMSSVT			
EGPR11	000002	DMSITS	DMSSAB									
EGPR12	000003	DMSSAB	DMSSTG									
EGPR13	000008	DMSSLN	DMSSVT									
EGPR14	000007	DMSDOS	DMSSAB	DMSSLN	DMSSTG	DMSSVT						
EGPR15	000039	DMSDOS	DMSIFC	DMSITS	DMSOVS	DMSSAB	DMSSLN	DMSSMN	DMSSCP	DMSSTG	DMSSVN	DMSSVT
EGPR2	000006	DMSITS	DMSSOP	DMSSVT								
EGPR5	000003	DMSXCP										
EGPR9	000004	DMSDOS	DMSSAB									
ENDBLOC	000003	DMSEDI	DMSEDX									
ENDCDADR	000006	DMSLDR	DMSLSB	DMSOLD								
ENDFREE	000002	DMSEXT	DMSLBT									
ENDTABS	000006	DMSEDI	DMSEDX									
ENTADR	000008	DMSLDR	DMSOLD									
ENTNAME	000005	DMSLDR	DMSLSB	DMSOLD								
EOCADR	000006	DMSDMP	DMSSMN	DMSSTG								
EQCHK	000002	DMSBOP	DMSFCH									
ERBIT	000008	DMSACF	DMSERS	DMSRNM								
ERBL	000001	DMSERR										
ERDSECT	000002	DMSERR										
ERF1BF	000002	DMSERR										
ERF1HD	000003	DMSERR										
ERF1SBN	000005	DMSERR										
ERF1SB1	000003	DMSERR										

LABEL	COUNT	REFERENCES
ERF1TX	000002	DMSERR
ERF2CM	000004	DMSERR
ERF2DI	000001	DMSERR
ERF2DT	000001	DMSERR
ERF2PR	000001	DMSERR
ERF2SI	000001	DMSERR
ERLET	000001	DMSERR
ERMESS	000002	DMSERR
ERNUM	000002	DMSERR
ERPAS13	000001	DMSERR
ERPBFA	000002	DMSERR
ERPCS	000001	DMSERR
ERPF1	000013	DMSERR
ERPF2	000010	DMSERR
ERPHDR	000001	DMSERR
ERPLET	000001	DMSERR
ERPNUM	000001	DMSERR
ERPSBA	000004	DMSERR
ERPTXA	000003	DMSERR
ERR$202	000004	DMSEXT
ERRCODE	000065	DMSACC DMSARN DMSDIO DMSHDI DMSHDS DMSLBM DMSSAB DMSSYN
ERRCOD0	000012	DMSACM
ERRCOD1	000020	DMSACF DMSERS DMSRNM
ERRET	000036	DMSCIO DMSINT DMSITS DMSPIO DMSPRT DMSPUN DMSVIP
ERRMSG	000023	DMSAMS DMSCIO DMSERS DMSEXT DMSFCH DMSPIC DMSUPD DMSXCP
ERRNUM	000002	DMSINT
ERROR	000196	DMSACM DMSARN DMSARX DMSASM DMSBTP DMSCMP DMSDLK DMSDSK DMSDSL DMSDSV DMSEDI DMSEDX DMSFCH DMSGRN DMSIFC DMSLBM DMSLIO DMSLLU DMSMOD DMSNCP DMSOVR DMSPRV DMSRDC DMSRNE DMSRRV DMSSCR DMSSET DMSSLN DMSSRV DMSSYN DMSTMA DMSTPD DMSTPE DMSUPD DMSVPD DMSXCP DMSZAP
ERSAVE	000007	DMSERR
ERSBD	000013	DMSERR
ERSBF	000010	DMSERR
ERSBL	000005	DMSERR
ERSECT	000001	DMSERR
ERSFA	000004	DMSERR
ERSFL	000005	DMSERR
ERSFLAG	000050	DMSERS DMSRNM
ERSFLST	000002	DMSERR
ERSSZ	000002	DMSERR
ERTEXT	000004	DMSERR
ERTPL	000004	DMSERR
ERTPLA	000006	DMSERR
ERTPLL	000008	DMSERR
ERTSIZE	000002	DMSERR
ERT1	000008	DMSERR

```
LABEL     COUNT      REFERENCES


ERT2      000013     DMSERR
ESD1ST    000011     DMSDLK    DMSLDR    DMSOLD
ESIDTB    000040     DMSLDR    DMSOLD
EXADD     000008     DMSEXC    DMSEXT
EXAMLC    000005     DMSDBG
EXAMLG    000006     DMSDBG
EXECFLAG  000003     DMSEXC
EXECRUN   000004     DMSEXC    DMSGRN
EXENACTE  000009     DMSVIP
EXENADDR  000002     DMSVIP
EXLEODF   000004     DMSVIP
EXLEODL   000001     DMSVIP
EXLEODP   000001     DMSVIP
EXLEVEL   000006     DMSEXC    DMSEXT
EXLJRN    000002     DMSVIP
EXLJRNL   000004     DMSVIP
EXLLEN    000009     DMSVIP
EXLLERF   000004     DMSVIP
EXLLERL   000001     DMSVIP
EXLLERP   000001     DMSVIP
EXLSYNF   000004     DMSVIP
EXLSYNL   000002     DMSVIP
EXLSYNP   000001     DMSVIP
EXNUM     000003     DMSEXC
EXSAVE    000007     DMSITE    DMSMVE
EXSAVE1   000009     DMSITE
EXTFLAG   000006     DMSIOW    DMSITE    DMSSVN
EXTM      000001     DMSQRY
EXTNPSW   000001     DMSINI
EXTOPSW   000021     DMSDBG    DMSITE
EXTPSW    000005     DMSINT    DMSITE
EXTRET    000007     DMSITE
EXTSECT   000013     DMSINS    DMSINT    DMSIOW    DMSITE    DMSQRY    DMSSET    DMSSTG    DMSSVN    DMSSVT
FCBBLKSZ  000005     DMSFLD    DMSMVE    DMSROS    DMSSOP
FCBBUFF   000045     DMSARN    DMSARX    DMSASM    DMSSBS    DMSSEB    DMSSOP    DMSSQS    DMSSVT
FCBBYTE   000052     DMSARN    DMSARX    DMSASM    DMSSBD    DMSSBS    DMSSEB    DMSSOP    DMSSQS    DMSSVT
FCBCASE   000004     DMSFLD    DMSSEB    DMSSOP
FCBCATML  000019     DMSARN    DMSARX    DMSASM    DMSFLD    DMSSBS    DMSSCT    DMSSOP    DMSSVT
FCBCLEAV  000004     DMSSOP
FCBCLOSE  000011     DMSARN    DMSARX    DMSASM    DMSSCT    DMSSOP    DMSSQS
FCBCON    000003     DMSFLD    DMSSOP
FCBCOUT   000026     DMSSBS    DMSSCT    DMSSEB    DMSSOP    DMSSQS    DMSSVT
FCBDCBCT  000004     DMSSOP
FCBDD     000022     DMSARN    DMSARX    DMSASM    DMSFCH    DMSFLD    DMSMVE    DMSQRY    DMSSAB    DMSSOP    DMSSVT
FCBDEV    000054     DMSARN    DMSARX    DMSASM    DMSFCH    DMSFLD    DMSMVE    DMSQRY    DMSSAB    DMSSBS    DMSSCT    DMSSEB    DMSSOP
                     DMSSQS    DMSSVT
```

LABEL	COUNT	REFERENCES											
FCBDOSL	000007	DMSFLD	DMSSOP	DMSSVT									
FCBDSK	000012	DMSARX	DMSASM	DMSFCH	DMSFLD	DMSMVE	DMSSOP	DMSSVT					
FCBDSMD	000035	DMSALU	DMSFLD	DMSMVE	DMSROS	DMSSBS	DMSSEB	DMSSOP	DMSSQS				
FCBDSNAM	000052	DMSARX	DMSASM	DMSFCH	DMSFLD	DMSMVE	DMSQRY	DMSROS	DMSSBS	DMSSCT	DMSSOP	DMSSVT	
FCBDSORG	000004	DMSFLD											
FCBDSTYP	000016	DMSFLD	DMSQRY	DMSROS	DMSSEB	DMSSOP	DMSSVT						
FCBDUM	000005	DMSFLD	DMSSAB	DMSSOP	DMSSVT								
FCBEND	000001	DMSFLD											
FCBENSIZ	000006	DMSFLD											
FCBFIRST	000016	DMSABN	DMSALU	DMSFLD	DMSQRY	DMSROS	DMSSAB	DMSSOP	DMSSVT				
FCBFORM	000012	DMSARN	DMSARX	DMSASM	DMSSEB	DMSSOP	DMSSVT						
FCBINIT	000069	DMSARN	DMSARX	DMSASM	DMSFCH	DMSFLD	DMSMVE	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVT
FCBIO	000001	DMSSEB											
FCBIORD	000003	DMSSQS											
FCBIOSW	000033	DMSARN	DMSARX	DMSASM	DMSFLD	DMSSCT	DMSSEB	DMSSOP	DMSSQS				
FCBIOSW2	000024	DMSDSL	DMSLDS	DMSMVE	DMSROS	DMSSEB	DMSSOP	DMSSVT					
FCBIOWR	000003	DMSSQS											
FCBITEM	000062	DMSARN	DMSARX	DMSASM	DMSDSL	DMSMVE	DMSSBD	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVT
FCBKEYS	000009	DMSSBD	DMSSOP	DMSSVT									
FCBLRECL	000006	DMSFLD	DMSMVE	DMSROS	DMSSOP								
FCBMEMBR	000013	DMSFLD	DMSLDS	DMSROS	DMSSEB	DMSSOP							
FCBMMV	000004	DMSMVE	DMSSVT										
FCBMODE	000006	DMSFLD	DMSSBS	DMSSEB	DMSSOP								
FCBMVFIL	000002	DMSMVE	DMSSEB										
FCBMVPDS	000017	DMSDSL	DMSLDS	DMSMVE	DMSROS	DMSSEB	DMSSOP	DMSSVT					
FCBNEXT	000004	DMSALU	DMSFLD	DMSROS									
FCBNUM	000013	DMSABN	DMSFLD	DMSQRY									
FCBOP	000119	DMSFCH	DMSMVE	DMSROS	DMSSBD	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVT		
FCBOPCB	000005	DMSMVE	DMSSEB										
FCBOS	000017	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSVT							
FCBOSDSN	000017	DMSFLD	DMSLDS	DMSROS									
FCBOSFST	000020	DMSALU	DMSFCH	DMSMVE	DMSROS	DMSSCT	DMSSOP	DMSSVT					
FCBPCH	000002	DMSFLD											
FCBPDS	000011	DMSSBS	DMSSCT	DMSSOP	DMSSVT								
FCBPROC	000009	DMSARN	DMSFLD	DMSROS	DMSSEB	DMSSOP							
FCBPROCC	000005	DMSARN	DMSARX	DMSASM	DMSSOP								
FCBPROCO	000003	DMSARN	DMSSOP										
FCBPRPU	000006	DMSSEB											
FCBPTR	000002	DMSFLD											
FCBPVMB	000003	DMSSQS											
FCBRDR	000005	DMSARX	DMSASM	DMSFLD	DMSSOP								
FCBREAD	000022	DMSARN	DMSARX	DMSASM	DMSSBS	DMSSEB	DMSSQS						
FCBRECFM	000007	DMSFLD	DMSMVE	DMSROS	DMSSBD	DMSSEB	DMSSOP						
FCBRECL	000005	DMSSEB	DMSSOP										
FCBR13	000002	DMSSCT	DMSSEB										
FCBSECT	000043	DMSALU	DMSARN	DMSARX	DMSASM	DMSDSL	DMSFCH	DMSFLD	DMSLDS	DMSMVE	DMSQRY	DMSROS	DMSSAB

LABEL	COUNT	REFERENCES											
		DMSSBD	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVN	DMSSVT				
FCBTAB	000001	DMSSVT											
FCBTAP	000010	DMSARX	DMSASM	DMSFLD	DMSMVE	DMSSBS	DMSSCT	DMSSOP	DMSSVT				
FCBTAPID	000006	DMSFLD	DMSMVE	DMSQRY	DMSSEB								
FCBTBSP	000004	DMSSBS	DMSSVT										
FCBTCLOS	000003	DMSSOP											
FCBXTENT	000011	DMSFLD	DMSSBD	DMSSBS	DMSSOP	DMSSVT							
FCHAPHNM	000002	DMSFET											
FCHLENG	000003	DMSDOS	DMSFET										
FCHOPT	000002	DMSFET											
FCHTAB	000008	DMSDOS	DMSFET										
FDISK	000003	DMSLDR	DMSNUC	DMSOLD									
FFD	000005	DMSACM	DMSAUD	DMSEXC									
FFE	000002	DMSACM	DMSAUD										
FFF	000004	DMSACM	DMSAUD										
FFS	000005	DMSGRN											
FILE	000080	DMSACM	DMSARX	DMSASM	DMSBOP	DMSCLS	DMSCMP	DMSDLB	DMSDSK	DMSDSL	DMSEDI	DMSEDX	DMSFLD
		DMSGLB	DMSGND	DMSIFC	DMSLBM	DMSLBT	DMSLGT	DMSLIB	DMSLIC	DMSLKD	DMSMOD	DMSNCP	DMSPRT
		DMSPUN	DMSRDC	DMSRNM	DMSSLN	DMSSTT	DMSSYN	DMSTPD	DMSTPE	DMSTYP	DMSZAP		
FILEBUFF	000023	DMSEXC	DMSPRT	DMSPUN	DMSRDC	DMSROS	DMSSVT	DMSTPD					
FILEBYTE	000009	DMSEXC	DMSROS	DMSSOP	DMSSVT								
FILECOUT	000002	DMSSVT											
FILEITEM	000007	DMSSVT											
FILEMODE	000013	DMSEXC	DMSNCP	DMSPRT	DMSPUN	DMSRDC	DMSSOP	DMSSVT	DMSTPD				
FILEMS	000006	DMSEDI											
FILENAME	000048	DMSINT	DMSNCP	DMSPRT	DMSPUN	DMSRDC	DMSROS	DMSSCT	DMSSCP	DMSSVT	DMSTPD		
FILEREAD	000002	DMSROS	DMSSOP										
FILETYPE	000013	DMSBOP	DMSCLS	DMSINT	DMSPRT	DMSPUN	DMSSOP	DMSSVT	DMSTPD				
FINIS	000066	DMSARN	DMSFNC	DMSFRE	DMSLBT	DMSLDR	DMSLIB	DMSLLU	DMSOLD	DMSSBD	DMSSRT	DMSTMA	DMSTPE
FINISLST	000004	DMSAUD	DMSFNS	DMSINT									
FIRSTDMP	000002	DMSDBG											
FLAG	000136	DMSEDI	DMSEDX	DMSEXT	DMSFOR	DMSLST	DMSMVE	DMSSCR	DMSSRT	DMSSVT	DMSTPD		
FLAGLOC	000004	DMSEDX	DMSSCR										
FLAGS	000164	DMSFRE	DMSITS	DMSLBM	DMSLBT	DMSLDR	DMSLIB	DMSLSB	DMSLST	DMSOLD	DMSOVS	DMSTPE	DMSZAP
FLAG1	000077	DMSARX	DMSASM	DMSEXT	DMSFLD	DMSLDR	DMSLIC	DMSLSB	DMSOLD				
FLAG2	000137	DMSARX	DMSASM	DMSASN	DMSEDI	DMSEDX	DMSFLD	DMSLDR	DMSLIB	DMSLIO	DMSLSB	DMSOLD	DMSSCR
		DMSTPD											
FLAG3	000019	DMSASN	DMSFLD	DMSLDR	DMSOLD								
FLCLN	000011	DMSFRE											
FLGSAVE	000002	DMSALU											
FLHC	000008	DMSFRE											
FLNU	000007	DMSFRE											
FLPA	000016	DMSFRE											
FMODE	000047	DMSEDI	DMSEDX	DMSEXT	DMSLBT	DMSLDS	DMSLGT	DMSLIB	DMSLST	DMSRDC	DMSRNE	DMSSCR	DMSTYP
FNAME	000062	DMSDSK	DMSEDI	DMSEDX	DMSEXT	DMSLGT	DMSLIB	DMSLIO	DMSLST	DMSPRV	DMSRNE	DMSRRV	DMSSCR
		DMSSRV	DMSTYP	DMSUPD	DMSVPD								

LABEL	COUNT	REFERENCES											
FNBIT	000004	DMSFNS											
FPRLOG	000003	DMSDBG											
FPTR	000008	DMSEDI	DMSUPD										
FRDSECT	000005	DMSFRE	DMSSET										
FREEAD	000003	DMSUPD											
FREEFLG1	000028	DMSFRE											
FREEFLG2	000036	DMSFRE											
FREEHN	000007	DMSFRE											
FREEHU	000009	DMSFRE											
FREELEN	000006	DMSEDI	DMSEDX	DMSUPD									
FREELN	000014	DMSBOP	DMSCLS	DMSFRE									
FREELOWE	000050	DMSABN	DMSARX	DMSASM	DMSDLK	DMSDOS	DMSDSV	DMSFCH	DMSFRE	DMSINS	DMSINT	DMSLBM	DMSLDR
		DMSLSB	DMSMOD	DMSNCP	DMSOLD	DMSSET	DMSSLN	DMSSMN	DMSSTG				
FREELOW1	000006	DMSFRE	DMSSET										
FREELU	000006	DMSFRE											
FREENEXT	000001	DMSEXT											
FREER0	000003	DMSDIO											
FREESAVE	000013	DMSFRE											
FRERESPG	000007	DMSFCH	DMSINS	DMSSET	DMSSMN	DMSSTG							
FRF1B	000002	DMSFRE											
FRF1C	000003	DMSFRE											
FRF1E	000003	DMSFRE											
FRF1H	000006	DMSFRE											
FRF1L	000006	DMSFRE											
FRF1M	000004	DMSFRE											
FRF1N	000003	DMSFRE											
FRF1V	000003	DMSFRE											
FRF2CKE	000003	DMSFRE											
FRF2CKT	000007	DMSFRE											
FRF2CKX	000003	DMSFRE											
FRF2CL	000012	DMSFRE											
FRF2NOI	000010	DMSFRE											
FRF2SVP	000003	DMSFRE											
FRSTLOC	000008	DMSMOD	DMSSLN										
FRSTSDID	000002	DMSLDR	DMSLSB										
FSCBBUFF	000007	DMSDLK	DMSIFC	DMSZAP									
FSCBD	000020	DMSBRD	DMSDLK	DMSIFC	DMSZAP								
FSCBFLG	000005	DMSBRD											
FSCBFM	000006	DMSDLK	DMSGRN	DMSIFC									
FSCBFN	000027	DMSDLK	DMSGRN	DMSIFC	DMSZAP								
FSCBFT	000007	DMSGRN	DMSZAP										
FSCBFV	000005	DMSBRD	DMSDLK	DMSIFC	DMSZAP								
FSCBITNO	000011	DMSDLK											
FSIZE	000009	DMSEDI	DMSEXT	DMSLBT	DMSRNE								
FSTBKWD	000001	DMSERS											
FSTD	000012	DMSCPY	DMSEDX	DMSEXC	DMSFNS	DMSGND	DMSNCP	DMSSOP	DMSTPE				

LABEL	COUNT	REFERENCES
FSTDATEW	000001	DMSGND
FSTDBC	000007	DMSDSK DMSERS DMSTPE
FSTFAP	000001	DMSSTT
FSTFAR	000001	DMSSTT
FSTFAW	000002	DMSCPY DMSSTT
FSTFB	000008	DMSCPY DMSDLK DMSFNS DMSSTT DMSZAP
FSTFCL	000003	DMSERS DMSTPE
FSTFINRD	000012	DMSCAT DMSCIT DMSCRD DMSEDX DMSEXT DMSINT DMSSVN
FSTFLAGS	000003	DMSSOP
FSTFMODE	000008	DMSACC DMSEDX DMSNCP DMSSOP
FSTFNAME	000003	DMSACC
FSTFRO	000001	DMSSTT
FSTFROX	000001	DMSSTT
FSTFRW	000003	DMSDLK DMSSTT DMSZAP
FSTFRWX	000002	DMSDLK DMSSTT
FSTFTYPE	000007	DMSACC
FSTFV	000023	DMSAMS DMSARX DMSASM DMSBRD DMSBWR DMSCPY DMSDLK DMSDSK DMSIFC DMSLBM DMSLKD DMSMVE DMSTPE DMSUPD DMSZAP
FSTFWDP	000002	DMSERS
FSTIC	000018	DMSACF DMSBOP DMSBRD DMSCPY DMSDLK DMSDSK DMSFNS DMSLBM DMSTPE DMSXCP DMSZAP
FSTIL	000025	DMSAMS DMSARX DMSASM DMSBWR DMSCPY DMSDLK DMSDSK DMSIFC DMSLBM DMSLKD DMSMVE DMSTPE DMSUPD DMSXCP DMSZAP
FSTITAV	000003	DMSBRD DMSCPY
FSTL	000005	DMSARN DMSARX DMSASM DMSDSL DMSLAF
FSTLRECL	000001	DMSEXC
FSTM	000028	DMSAMS DMSARN DMSARX DMSASM DMSBOP DMSCPY DMSDLK DMSDSK DMSERS DMSFNS DMSIFC DMSLBM DMSLKD DMSRNM DMSSTT DMSTPE DMSUPD DMSZAP
FSTN	000014	DMSAMS DMSCPY DMSDSK DMSERS DMSFNS DMSRNM DMSTPE
FSTNOIT	000001	DMSBRD
FSTRECAV	000002	DMSBRD
FSTRECCT	000001	DMSEDX
FSTRECFM	000001	DMSEDX
FSTRP	000004	DMSACF DMSBRD DMSFNS DMSTPE
FSTRWDSK	000001	DMSSOP
FSTSECT	000059	DMSACF DMSAMS DMSARN DMSARX DMSASM DMSBOP DMSBRD DMSBWR DMSCPY DMSDLK DMSDSK DMSDSL DMSERS DMSFNS DMSGND DMSIFC DMSLAF DMSLBM DMSLKD DMSMVE DMSRNM DMSSTT DMSTPE DMSUPD DMSXCP DMSZAP
FSTT	000009	DMSACF DMSDSK DMSERS DMSFNS DMSRNM DMSTPE
FSTWP	000010	DMSACF DMSBWR DMSFNS DMSTPE
FSTXRDSK	000002	DMSSOP
FSTXTADR	000007	DMSLDR DMSLOA DMSLSB DMSOLD
FSTYR	000006	DMSCPY DMSFNS
FTRDCONV	000001	DMSTPE
FTRDLDNS	000004	DMSTPE
FTRTRANS	000001	DMSTPE
FTRUCS	000001	DMSASN

LABEL	COUNT	REFERENCES											
FTR35MB	000001	DMSASN											
FTR7TRK	000001	DMSTPE											
FTYPE	000019	DMSEDI	DMSEDX	DMSLDR	DMSLGT	DMSLIB	DMSLST	DMSOLD	DMSPRV	DMSRRV	DMSSCR	DMSSRV	DMSTYP
FV	000014	DMSEDI	DMSEDX	DMSSCR									
FVS	000002	DMSINS	DMSITE										
FVSDSKA	000002	DMSACM	DMSAUD										
FVSECT	000065	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCIT	DMSCRD
		DMSCWR	DMSCWT	DMSDIO	DMSDOS	DMSDSK	DMSERS	DMSFNS	DMSINT	DMSITE	DMSITI	DMSITP	DMSITS
		DMSLAD	DMSLFS	DMSMOD	DMSPNT	DMSQRY	DMSRNM	DMSSLN	DMSSCP	DMSSTT	DMSTPE	DMSTQQ	
FVSERAS0	000013	DMSERS	DMSRNM										
FVSERAS1	000012	DMSERS	DMSRNM										
FVSERAS2	000004	DMSERS	DMSRNM										
FVSFSTAD	000004	DMSMOD	DMSPUN	DMSSTT									
FVSFSTCL	000001	DMSMOD											
FVSFSTDT	000002	DMSSTT											
FVSFSTFV	000001	DMSMOD											
FVSFSTIC	000003	DMSACM	DMSBTB	DMSMOD									
FVSFSTIL	000003	DMSACM	DMSBTB	DMSMOD									
FVSFSTM	000002	DMSDSK	DMSSTT										
FVSFSTN	000001	DMSSTT											
FXD	000023	DMSDSL	DMSSEB	DMSSOP	DMSSQS	DMSTMA	DMSTPD						
F0	000025	DMSDBG	DMSINS	DMSITE	DMSITS								
F1	000011	DMSDLK	DMSDSV	DMSEXT	DMSSTG								
F15	000005	DMSDBG											
F2	000015	DMSDLK	DMSITE										
F255	000002	DMSCRD											
F256	000008	DMSCWR	DMSHDI	DMSHDS									
F3	000009	DMSAUD	DMSDLK										
F4	000016	DMSDLK	DMSITE	DMSTQQ									
F4096	000002	DMSAMS	DMSDBD										
F5	000008	DMSDLK	DMSXCP										
F6	000033	DMSDBG	DMSDLK	DMSITE	DMSITS	DMSSOP							
F65535	000007	DMSACF	DMSDSK	DMSMOD	DMSPNT	DMSSLN	DMSTQQ						
F7	000006	DMSEOP	DMSOR3	DMSXCP									
F800	000004	DMSACM	DMSAUD	DMSDSK									
GDISK	000001	DMSNUC											
GETFLAG	000007	DMSEDI											
GET1	000002	DMSLSY											
GIOPLIST	000001	DMSSCR											
GPRLOG	000011	DMSDBG	DMSITS										
GPRSAV	000004	DMSLDR	DMSOLD										
GRAFDEV	000001	DMSINS											
HALF	000002	DMSEDI	DMSLDS										
HEX	000041	DMSCPY	DMSDBG	DMSDLK	DMSDOS	DMSDSV	DMSEDI	DMSFNS	DMSPRT	DMSSSK	DMSTPE	DMSTYP	DMSZAP
HEXHEX	000010	DMSDBG											
HIPHAS	000006	DMSFCH	DMSFET										

```
LABEL     COUNT     REFERENCES


HIPROG    000002    DMSFCH
HOLD      000012    DMSBOP    DMSDSK    DMSITI
HOLDFLAG  000015    DMSSCR
IADT      000003    DMSACC    DMSDSK    DMSLAD
IC        000003    DMSBOP    DMSDBG
IHADEB    000020    DMSFCH    DMSMVE    DMSSBS    DMSSCT    DMSSOP    DMSSQS    DMSSVT
IHADECB   000006    DMSSBD    DMSSBS    DMSSCT    DMSSEB    DMSSVT
IHAJFCB   000001    DMSSVT
IJBABTAB  000004    DMSBAB    DMSDOS    DMSITP
IJBBOX    000001    DMSSTG
IJBCCWT   000001    DMSDOS
IJBFLG04  000001    DMSBOP
IJBFTTAB  000004    DMSDOS    DMSFET
IKQACB    000007    DMSBOP    DMSCLS    DMSVIP
IKQEXLST  000003    DMSVIP
IKQRPL    000006    DMSVIP
INCRNO    000003    DMSEDI
INHIBIT   000002    DMSDIO
INPUT     000068    DMSARN    DMSBOP    DMSCPY    DMSDBD    DMSDBG    DMSDSL    DMSDSV    DMSEDI    DMSFCH    DMSGRN    DMSITE    DMSMVE
                    DMSNCP    DMSNUC    DMSOR1    DMSPRV    DMSQRY    DMSRRV    DMSSRV    DMSTPE    DMSXCP    DMSZAP
INPUTSIZ  000002    DMSDBG
INPUT1    000002    DMSDBG
INSIZE    000006    DMSLBM    DMSSRT
INSTALID  000005    DMSINI    DMSPRT
INTINFO   000006    DMSDOS    DMSITP
INTREQ    000001    DMSFCH
INVLD     000003    DMSEDI
INVLDHDR  000001    DMSEDX
IOAD      000002    DMSEDX
IOAREA    000002    DMSRDC    DMSTYP
IOBBCSW   000003    DMSSBS    DMSSEB
IOBBECBC  000002    DMSSEB
IOBBECBP  000003    DMSSBS    DMSSEB
IOBBFLG   000002    DMSSBS    DMSSCT
IOBCSW    000006    DMSARN    DMSARX    DMSASM    DMSSBS    DMSSCT
IOBDCBPT  000001    DMSSOP
IOBECB    000004    DMSIFC    DMSSQS
IOBECBPT  000003    DMSSQS
IOBEND    000001    DMSSOP
IOBIN     000032    DMSARN    DMSARX    DMSASM    DMSSBD    DMSSBS    DMSSEB    DMSSOP    DMSSQS    DMSSVT
IOBIOFLG  000045    DMSARN    DMSARX    DMSASM    DMSSBD    DMSSBS    DMSSCT    DMSSEB    DMSSCP    DMSSQS    DMSSVT
IOBNXTAD  000003    DMSSOP
IOBOUT    000007    DMSSBS    DMSSCT    DMSSQS
IOBSTART  000008    DMSSOP    DMSSQS
IOBUPD    000004    DMSSQS
IOCOMM    000007    DMSDIO
```

LABEL	COUNT	REFERENCES						
IOID	000005	DMSEDI	DMSEDX					
IOLIST	000051	DMSEDI	DMSEDX					
IOMODE	000003	DMSEDI	DMSEDX					
IONPSW	000006	DMSINI	DMSINS	DMSIOW	DMSITE			
IONTABL	000012	DMSABN	DMSHDI	DMSINT	DMSIOW	DMSITI		
IOOLD	000002	DMSDIO	DMSITI					
IOOPSW	000027	DMSCIT	DMSDBG	DMSDIO	DMSINI	DMSIOW	DMSITE	DMSITI
IOPSW	000001	DMSITI						
IOSAVE	000005	DMSITI						
IOSECT	000004	DMSABN	DMSHDI	DMSINT	DMSITI			
IPLADDR	000003	DMSBTP	DMSINS					
IPLCCW1	000001	DMSINI						
IPLPSW	000009	DMSABN	DMSDBG	DMSINI	DMSINS			
ITEM	000073	DMSBRD	DMSEDI	DMSEDX	DMSSCR	DMSUPD		
ITSBIT	000007	DMSITS						
JAR	000003	DMSEDI	DMSEDX					
JCSW2	000001	DMSDOS						
JCSW3	000016	DMSOPT	DMSSET					
JCSW4	000005	DMSDOS	DMSOPT	DMSSET				
JFCBIND2	000002	DMSFLD	DMSSOP					
JFCBMASK	000022	DMSSOP	DMSSVT					
JFCBUFNO	000001	DMSFLD						
JFCDSORG	000002	DMSSOP						
JFCKEYLE	000003	DMSFLD	DMSSOP					
JFCLIMCT	000003	DMSFLD	DMSSOP					
JFCLRECL	000001	DMSSVT						
JFCOPTCD	000008	DMSFLD	DMSSOP					
JFIRST	000009	DMSHDS	DMSITS					
JFLAGS	000014	DMSDBG						
JLAST	000010	DMSHDS	DMSITS					
JNUMB	000012	DMSHDS	DMSINT	DMSITS				
JOBDATE	000004	DMSDLK	DMSDOS	DMSSET				
JR1	000008	DMSITE						
JSR0	000012	DMSACF	DMSACM					
JSYM	000002	DMSLSY						
KEYCHNG	000006	DMSSBD	DMSSVT					
KEYCOUT	000004	DMSSBD	DMSSVT					
KEYFORM	000002	DMSSVT						
KEYLNGTH	000010	DMSSBD	DMSSVT					
KEYMAX	000002	DMSITS						
KEYNAME	000007	DMSSBD	DMSSVT					
KEYOP	000009	DMSSBD	DMSSVT					
KEYP	000008	DMSITS						
KEYS	000003	DMSBTP	DMSITS					
KEYSECT	000002	DMSSBD	DMSSVT					
KEYTABLE	000011	DMSSVT						

```
LABEL     COUNT     REFERENCES

KEYTBLAD 000009    DMSSBD   DMSSVT
KEYTBLNO 000016    DMSSBD   DMSSVT
KEYTYPE  000002    DMSSVT
KXFLAG   000020    DMSABN   DMSACC   DMSAUD   DMSBWR   DMSCIT   DMSCRD   DMSCWR   DMSCWT   DMSDIO   DMSDSK   DMSERS   DMSFNS
                   DMSITI   DMSITS   DMSRNM   DMSTPE
KXWANT   000013    DMSABN   DMSACC   DMSAUD   DMSBWR   DMSCIT   DMSDIO   DMSDSK   DMSERS   DMSFNS   DMSITI   DMSITS   DMSRNM
                   DMSTPE
KXWSVC   000005    DMSCRD   DMSCWR   DMSCWT   DMSITS
LABLEN   000003    DMSDLK   DMSEXT
LASTALOC 000004    DMSITS
LASTCMND 000011    DMSEXT   DMSINT
LASTCYL  000003    DMSDIO
LASTDMP  000001    DMSDBG
LASTEXEC 000002    DMSEXT
LASTHED  000003    DMSDIO
LASTLINE 000012    DMSDBD   DMSZAP
LASTLMOD 000002    DMSMOD   DMSSLN
LASTLOC  000001    DMSFET
LASTREC  000014    DMSCLS   DMSDIO   DMSZAP
LASTTMOD 000008    DMSITS   DMSLSB   DMSMOD   DMSSLN
LASTUSER 000003    DMSITE   DMSSAB   DMSSOP
LDMSROS  000004    DMSABN   DMSACM   DMSALU
LDRADDR  000014    DMSLDR   DMSLIO   DMSLOA   DMSOLD
LDRFLAGS 000019    DMSLDR   DMSLOA   DMSMOD   DMSOLD   DMSSLN
LDRRTCD  000003    DMSLDR   DMSOLD
LDRST    000009    DMSLDR   DMSLGT   DMSLIB   DMSLIO   DMSLSB   DMSOLD
LENOVS   000003    DMSITS   DMSOVR
LINE     000053    DMSBTP   DMSDBD   DMSDBG   DMSEDI   DMSEDX   DMSITE   DMSNUC
LINELOC  000002    DMSEDX   DMSSCR
LINENO   000002    DMSEDI
LINE1    000002    DMSDBD   DMSLIO
LINE1A   000001    DMSDBD
LINE1B   000001    DMSDBD
LINE1C   000001    DMSDBD
LINKLAST 000007    DMSSAB   DMSSLN   DMSSTG
LINKLEN  000004    DMSEXT
LINKSTRT 000009    DMSSLN   DMSSTG   DMSSVT
LMCURR   000005    DMSEDI
LMINCR   000005    DMSEDI
LMSTART  000010    DMSEDI   DMSEDX
LOADLIST 000001    DMSIFC
LOADSTRT 000004    DMSINS   DMSSET
LOC      000156    DMSABN   DMSACC   DMSACF   DMSACM   DMSALU   DMSAMS   DMSAUD   DMSBCP   DMSBWR   DMSCIT   DMSCLS   DMSCMP
                   DMSCRD   DMSDIO   DMSDLB   DMSDMP   DMSDOS   DMSEDX   DMSERS   DMSEXC   DMSEXT   DMSFCH   DMSFET   DMSFLD
                   DMSFNC   DMSFNS   DMSFOR   DMSFRE   DMSGIO   DMSGLE   DMSHDI   DMSHDS   DMSIFC   DMSINS   DMSINT   DMSITE
                   DMSITP   DMSITS   DMSLAD   DMSLAF   DMSLDR   DMSLGT   DMSLIB   DMSLSB   DMSMOD   DMSOLD   DMSOPL   DMSOR1
```

LABEL	COUNT	REFERENCES											
		DMSOVR	DMSPRT	DMSPUN	DMSQRY	DMSRNE	DMSROS	DMSSAB	DMSSET	DMSSLN	DMSSOP	DMSSQS	DMSSTG
		DMSSVN	DMSSVT	DMSSYN	DMSTPE	DMSTYP	DMSUPD	DMSVIP	DMSVPD	DMSVSR	DMSXCP	DMSZAP	
LOCCNT	000039	DMSACM	DMSBTB	DMSEDX	DMSFET	DMSFRE	DMSINS	DMSINT	DMSLDR	DMSLOA	DMSMOD	DMSOLD	DMSSET
		DMSSLN	DMSSMN	DMSSTG									
LOCCT	000025	DMSLDR	DMSLSB	DMSOLD									
LOWSAVE	000007	DMSDBG	DMSSVT										
LSTFINRD	000005	DMSCIT	DMSCRD	DMSSVN									
LTK	000009	DMSAMS	DMSDOS	DMSITP	DMSSET								
LUB	000004	DMSDLK	DMSDSV										
LUBCLB	000002	DMSDSV											
LUBP	000002	DMSDSV											
LUBPR	000002	DMSDLK	DMSDSV										
LUBPT	000016	DMSAMS	DMSBOP	DMSCLS	DMSDLB	DMSFCH	DMSLLU	DMSOPL	DMSPRV	DMSRRV	DMSSET	DMSSRV	DMSXCP
LUBRES	000003	DMSDLK	DMSDSV										
LUBRLB	000003	DMSDLK	DMSDSV										
LUBSLB	000001	DMSDSV											
LUB014	000002	DMSDLK	DMSDSV										
LUNDEF	000012	DMSLDR	DMSOLD										
MACDIRC	000011	DMSABN	DMSSCT	DMSSOP	DMSSTG	DMSSVT							
MACLBSV	000004	DMSGLB											
MACLIBL	000009	DMSGLB	DMSQRY	DMSSCT	DMSSOP	DMSSTG	DMSSVT						
MACRO	000003	DMSEDI											
MAINAD	000003	DMSEDX											
MAINHIGH	000037	DMSARX	DMSASM	DMSDOS	DMSFCH	DMSFRE	DMSINS	DMSLDR	DMSLOA	DMSLSE	DMSSET	DMSSMN	DMSSTG
MAINLIST	000012	DMSDOS	DMSFCH	DMSSMN	DMSSTG								
MAINSTRT	000008	DMSDOS	DMSFCH	DMSSMN	DMSSTG								
MAX	000013	DMSASM	DMSFRE										
MAXCODE	000001	DMSFRE											
MCKM	000014	DMSINI	DMSINS	DMSITS									
MCKNPSW	000001	DMSINI											
MDPCALL	000004	DMSMDP	DMSMOD										
MEMBOUND	000008	DMSLDR	DMSOLD										
MISFLAGS	000043	DMSABN	DMSACC	DMSAMS	DMSARN	DMSARX	DMSASM	DMSCAT	DMSCIT	DMSCPY	DMSCRD	DMSEDI	DMSEXC
		DMSINS	DMSINT	DMSITI	DMSITS	DMSLBM	DMSLPT	DMSLKD	DMSQRY	DMSSET	DMSSRT	DMSSTG	DMSUPD
MODDISP	000001	DMSZAP											
MODFLGS	000027	DMSACM	DMSINS	DMSLDR	DMSLSB	DMSMDP	DMSMOD	DMSOLD	DMSSET				
MODGNALL	000002	DMSMOD											
MODGNDOS	000003	DMSMOD											
MODLIST	000002	DMSITS	DMSSLN										
MSGFLAGS	000025	DMSCAT	DMSCIT	DMSCRD	DMSCWR	DMSEDI	DMSEXT	DMSINS	DMSINT	DMSQRY	DMSSET	DMSTYP	
MVCNT	000001	DMSDBG											
MVCNT1	000004	DMSDBD	DMSDBG	DMSITE	DMSNUC								
MVCNT2	000001	DMSDBG											
NDIKQLAB	000002	DMSXCP											
NEED	000007	DMSEXT	DMSLDR	DMSOLD									
NEGITS	000013	DMSCAT	DMSEXC	DMSINT	DMSITS	DMSQRY	DMSSET						

LABEL	COUNT	REFERENCES							
NEWPLKS	000005	DMSSVT							
NEWMODE	000009	DMSEDI	DMSRNM						
NEWNAME	000020	DMSEDI	DMSRNM	DMSUPD					
NEWTYPE	000005	DMSEDI	DMSRNM						
NEXTO	000001	DMSITI							
NICCIBM	000008	DMSNCP							
NICCTLR	000001	DMSNCP							
NICDISA	000004	DMSNCP							
NICEPMD	000002	DMSNCP							
NICGRAF	000004	DMSNCP							
NICLBSC	000001	DMSNCP							
NICLGRP	000002	DMSNCP							
NICLINE	000003	DMSNCP							
NICLPT	000005	DMSBOP	DMSCLS	DMSDLB	DMSLLU	DMSXCP			
NICMLTP	000001	DMSNCP							
NICRCPU	000028	DMSNCP							
NICRSPL	000006	DMSNCP							
NICSDLC	000001	DMSNCP							
NICSWCH	000001	DMSNCP							
NICSWEP	000001	DMSNCP							
NICTELE	000008	DMSNCP							
NICTERM	000003	DMSNCP							
NOABBREV	000006	DMSINA	DMSINT	DMSQRY	DMSSET				
NOAUTO	000007	DMSDLK	DMSLDR	DMSLIB	DMSLOA	DMSLSB	DMSOLD		
NODUP	000007	DMSLDR	DMSLSB	DMSOLD					
NOERASE	000008	DMSARN	DMSARX	DMSASM	DMSLIO	DMSLOA	DMSMOD	DMSUPD	
NOIMPCP	000007	DMSINT	DMSQRY	DMSSET					
NOIMPEX	000004	DMSINT	DMSQRY	DMSSET					
NOINV	000005	DMSLDR	DMSLOA	DMSLSB	DMSOLD				
NOLIBE	000009	DMSLBT	DMSLDR	DMSLIB	DMSLOA	DMSLSB	DMSOLD		
NOMAP	000007	DMSDLK	DMSLIO	DMSLOA	DMSLSB				
NOMAPFLG	000003	DMSMOD							
NOP	000014	DMSINI	DMSXCP						
NOPAGREL	000005	DMSABN	DMSINT	DMSQRY	DMSSET				
NORDYMSG	000002	DMSSET							
NORDYTIM	000006	DMSINT	DMSQRY	DMSSET					
NOREP	000006	DMSLDR	DMSLOA	DMSLSB	DMSOLD	DMSUPD			
NOSLCADR	000006	DMSLDR	DMSOLD						
NOSTDSYN	000005	DMSINA	DMSQRY	DMSSYN					
NOSYS	000002	DMSEXC							
NOTEXT	000009	DMSDOS	DMSFCH	DMSFET					
NOTIME	000002	DMSPUN							
NOTYPING	000011	DMSCAT	DMSCIT	DMSCRD	DMSCWR	DMSEDI	DMSEXT	DMSINT	DMSTYP
NOVMREAD	000003	DMSINS	DMSINT	DMSSET					
NRMRET	000010	DMSABN	DMSITS	DMSVIP					
NRMSAV	000019	DMSITS							

```
LABEL     COUNT      REFERENCES


NRMUSAV   000001     DMSITS
NUCCODE   000004     DMSFRE
NUCKEY    000002     DMSFRE   DMSSET
NUCON     000428     DMSABN   DMSACC   DMSACF   DMSACM   DMSALU   DMSAMS   DMSARE   DMSARN   DMSARX   DMSASM   DMSASN   DMSAUD
                     DMSBAB   DMSBOP   DMSBRD   DMSBTB   DMSBTP   DMSBWR   DMSCAT   DMSCIC   DMSCIT   DMSCLS   DMSCMP   DMSCPF
                     DMSCPY   DMSCRD   DMSCWR   DMSCWT   DMSDBD   DMSDBG   DMSDIO   DMSDLB   DMSDLK   DMSDMP   DMSDOS   DMSDSK
                     DMSDSL   DMSDSV   DMSEDI   DMSEDX   DMSERR   DMSERS   DMSEXC   DMSEXT   DMSFCH   DMSFET   DMSFLD   DMSFNS
                     DMSFOR   DMSFRE   DMSGIO   DMSGLB   DMSGND   DMSHDI   DMSHDS   DMSIFC   DMSINA   DMSINI   DMSINM   DMSINS
                     DMSINT   DMSIOW   DMSITE   DMSITI   DMSITP   DMSITS   DMSLAD   DMSLAF   DMSLBM   DMSLBT   DMSLDR   DMSLDS
                     DMSLFS   DMSLGT   DMSLIB   DMSLIO   DMSLKD   DMSLLU   DMSLOA   DMSLSB   DMSLST   DMSLSY   DMSMDP   DMSMOD
                     DMSMVE   DMSNCP   DMSOLD   DMSOPL   DMSOPT   DMSOR1   DMSOVR   DMSOVS   DMSPIO   DMSPNT   DMSPRT   DMSPRV
                     DMSPUN   DMSQRY   DMSRDC   DMSRNE   DMSRNM   DMSROS   DMSRRV   DMSSAB   DMSSBS   DMSSCN   DMSSCT   DMSSEB
                     DMSSET   DMSSLN   DMSSMN   DMSSOP   DMSSQS   DMSSRT   DMSSRV   DMSSSK   DMSSTG   DMSSTT   DMSSVN   DMSSVT
                     DMSSYN   DMSTIO   DMSTPD   DMSTPE   DMSTQQ   DMSTYP   DMSUPD   DMSVIB   DMSVIP   DMSVSR   DMSXCP   DMSZAP
NUCRSV3   000001     DMSDOS
NUM       000574     DMSABN   DMSACC   DMSAMS   DMSARE   DMSARN   DMSARX   DMSASM   DMSASN   DMSBOP   DMSBTB   DMSBTP   DMSBWR
                     DMSCIO   DMSCLS   DMSCMP   DMSCPY   DMSDIO   DMSDLB   DMSDLK   DMSDMP   DMSDOS   DMSDSK   DMSDSL   DMSDSV
                     DMSEDI   DMSEDX   DMSERS   DMSEXC   DMSFET   DMSFLD   DMSFNC   DMSFNS   DMSFOR   DMSFRE   DMSGND   DMSIFC
                     DMSINA   DMSINS   DMSITP   DMSITS   DMSLBM   DMSLBT   DMSLDR   DMSLDS   DMSLIO   DMSLST   DMSMOD   DMSMVE
                     DMSNCP   DMSOLD   DMSOPL   DMSOR1   DMSOVR   DMSPIO   DMSPRT   DMSPUN   DMSQRY   DMSRDC   DMSREA   DMSRNM
                     DMSSCR   DMSSCT   DMSSET   DMSSOP   DMSSRT   DMSSSK   DMSSVT   DMSSYN   DMSTPD   DMSTPE   DMSTYP   DMSUPD
                     DMSSCR   DMSSCT   DMSSET   DMSSOP   DMSSRT   DMSSSK   DMSSVT   DMSSYN   DMSTPD   DMSTPE   DMSTYP   DMSUPD
                     DMSVIB   DMSVIP   DMSVPD   DMSZAP
NUMBYTE   000005     DMSLDR   DMSLIB   DMSOLD
NUMFINRD  000014     DMSABN   DMSBTP   DMSCAT   DMSCIT   DMSCRD   DMSSVN
NUMLOC    000002     DMSEDX   DMSSCR
NUMPNDWR  000016     DMSCIT   DMSCRD   DMSCWR   DMSCWT   DMSITE   DMSSVN
NXTSYM    000004     DMSLDR   DMSLSY   DMSOLD
OFF       000042     DMSBTP   DMSCLS   DMSDBG   DMSEDI   DMSEXT   DMSITS   DMSOVR   DMSSET   DMSUPD   DMSXCP
OLDEST    000001     DMSITI
OLDPSW    000071     DMSABN   DMSBAB   DMSDOS   DMSERR   DMSIFC   DMSITS   DMSOVS   DMSSAB   DMSSLN   DMSSTG   DMSSVT   DMSVIP
ON        000047     DMSBOP   DMSEDI   DMSEDX   DMSERS   DMSEXT   DMSITS   DMSLDS   DMSOR1   DMSOVR   DMSOVS   DMSRNM   DMSSET
                     DMSUPD   DMSXCP
OPSECT    000029     DMSABN   DMSARX   DMSASM   DMSCPY   DMSCRD   DMSCWR   DMSCWT   DMSDBG   DMSEXC   DMSEXT   DMSINS   DMSINT
                     DMSROS   DMSSBD   DMSSBS   DMSSCT   DMSSEB   DMSSOP   DMSSQS   DMSSVN   DMSSVT
OPSW      000016     DMSITP
OPTFLAGS  000030     DMSABN   DMSINA   DMSINS   DMSINT   DMSQRY   DMSSET   DMSSYN
OPTNBYTE  000001     DMSSTG
ORG       000004     DMSDBG
OSADTDSK  000009     DMSLDS   DMSROS
OSADTFST  000005     DMSABN   DMSALU   DMSROS
OSADTVTA  000008     DMSACM   DMSLDS   DMSROS
OSADTVTB  000008     DMSLDS   DMSROS
OSFST     000013     DMSABN   DMSALU   DMSBOP   DMSDLK   DMSFCH   DMSMVE   DMSROS   DMSRRV   DMSSOP   DMSSRV   DMSSTT
OSFSTALT  000009     DMSROS
OSFSTBLK  000005     DMSMVE   DMSROS   DMSSOP
OSFSTCHR  000014     DMSROS   DMSSOP
```

```
LABEL     COUNT     REFERENCES


OSFSTDBK 000002     DMSROS
OSFSTDSK 000006     DMSDLK   DMSFCH   DMSROS   DMSRRV   DMSSRV
OSFSTDSN 000002     DMSROS
OSFSTEND 000007     DMSROS
OSFSTEX4 000006     DMSROS
OSFSTFLG 000023     DMSROS   DMSSTT
OSFSTFM  000007     DMSBOP   DMSROS   DMSSTT
OSFSTFVF 000002     DMSROS
OSFSTLRL 000005     DMSMVE   DMSROS   DMSSOP
OSFSTLTH 000005     DMSABN   DMSALU   DMSROS
OSFSTMEM 000001     DMSROS
OSFSTMVL 000001     DMSROS
OSFSTNTE 000011     DMSROS
OSFSTNXT 000004     DMSABN   DMSALU   DMSROS
OSFSTRFM 000012     DMSBOP   DMSMVE   DMSROS   DMSSOP
OSFSTRSW 000009     DMSROS
OSFSTTRK 000008     DMSROS
OSFSTTYP 000003     DMSROS
OSFSTUMV 000001     DMSROS
OSFSTXNO 000005     DMSBOP   DMSROS
OSFSTXTN 000013     DMSBOP   DMSDLK   DMSFCH   DMSROS   DMSRRV   DMSSRV
OSIOTYPE 000016     DMSARX   DMSASM   DMSSBS   DMSSOP   DMSSQS   DMSSVT
OSMODLDW 000013     DMSABN   DMSINS   DMSSET
OSRESET  000010     DMSEXT   DMSINT   DMSLDR   DMSOLD   DMSSLN   DMSSVT
OSSFLAGS 000059     DMSARN   DMSARX   DMSASM   DMSCIT   DMSEXT   DMSIFC   DMSINT   DMSITE   DMSLDR   DMSLIB   DMSLIO   DMSOLD
                    DMSSLN   DMSSMN   DMSSTG   DMSSVN   DMSSVT
OSSMNU   000005     DMSSMN
OSTEMP   000029     DMSBAB   DMSDOS   DMSSLN   DMSSVT
OSWAIT   000006     DMSCIT   DMSITE   DMSSVN
OUTBUF   000053     DMSLDR   DMSLGT   DMSLIB   DMSLIO   DMSLSB   DMSOLD   DMSRRV   DMSSRV
OUTPT1   000010     DMSDBG
OUTPUT   000034     DMSDLK   DMSDSL   DMSGRN   DMSLDR   DMSLIO   DMSMVE   DMSOLD   DMSCR1   DMSOVS   DMSQRY   DMSTPE   DMSXCP
OVAPF    000004     DMSOVR   DMSOVS
OVBPF    000005     DMSOVR   DMSOVS
OVF1F    000002     DMSOVR   DMSOVS
OVF1FS   000002     DMSOVR   DMSOVS
OVF1GA   000002     DMSOVR   DMSOVS
OVF1GB   000003     DMSOVR   DMSOVS
OVF1GS   000002     DMSOVR   DMSOVS
OVF1ON   000011     DMSOVR   DMSOVS
OVF1PA   000002     DMSOVR   DMSOVS
OVF2CM   000003     DMSOVR   DMSOVS
OVF2NR   000003     DMSOVR   DMSOVS
OVF2OS   000003     DMSOVR   DMSOVS
OVF2ST   000001     DMSOVS
OVF2WA   000002     DMSOVR
```

LABEL	COUNT	REFERENCES
OVSAFT	000011	DMSITS DMSOVS
OVSECT	000003	DMSITS DMSOVR
OVSHO	000004	DMSCIT DMSOVR DMSOVS
OVSON	000008	DMSCIT DMSITS DMSOVR DMSOVS
OVSSO	000006	DMSCIT DMSOVR DMSOVS
OVSTAT	000029	DMSCIT DMSITE DMSITS DMSOVR DMSOVS
PACK	000029	DMSASN DMSBOP DMSBTP DMSCIT DMSCPY DMSDLK DMSEDI DMSFLD DMSLIO DMSRNE DMSTMA
PADBUF	000017	DMSEDI DMSEDX
PADCHAR	000007	DMSEDI DMSEDX
PARMLIST	000013	DMSGRN DMSLDR DMSLIO DMSOLD
PCPTR	000004	DMSBAB DMSDOS DMSITP
PCTVSAM	000002	DMSFCH DMSSTG
PDSBLKSI	000008	DMSSVT
PDSDIR	000003	DMSSVT
PDSSECT	000002	DMSSTG DMSSVT
PENDREAD	000022	DMSCIT DMSCRD DMSCWR DMSCWT DMSITE DMSSVN
PENDWRIT	000011	DMSCIT DMSCWR DMSSVN
PGMNPSW	000006	DMSABN DMSINS DMSITP
PGMOPSW	000017	DMSABN DMSDBG DMSITP DMSSAB
PGMSECT	000006	DMSITP DMSSAB DMSSLN DMSSTG DMSSVT
PIBADR	000010	DMSBAB DMSDOS DMSITP
PIBFLG	000001	DMSDOS
PIBPT	000022	DMSAMS DMSBAB DMSBOP DMSCLS DMSDOS DMSITP DMSSET
PIBSAVE	000015	DMSBAB DMSDOS DMSITP
PIB2PTR	000003	DMSDOS DMSVSR
PICADDR	000004	DMSITP DMSSTG
PIE	000002	DMSITP
PIK	000014	DMSBAB DMSDOS DMSITP DMSVSR
PLIST	000123	DMSBOP DMSBRD DMSBWR DMSCLS DMSDIO DMSDLK DMSDMP DMSDSV DMSEDI DMSEDX DMSEXC DMSINT DMSLBM DMSMDP DMSMVE DMSRNE DMSSOP DMSSVT DMSTIO DMSUPD
PLISTSAV	000018	DMSLDR DMSLIO DMSOLD
PNOTFND	000008	DMSDOS DMSFCH DMSFET
PO	000015	DMSDLK DMSDSL DMSFCH DMSLDS DMSNCP DMSRCS DMSSBS DMSSEB DMSSOP
POINTER	000026	DMSFRE
POU	000001	DMSLDS
PPBEG	000002	DMSDOS
PPEND	000018	DMSDOS DMSFCH DMSSET DMSSMN DMSSTG DMSVSR
PREVCMND	000005	DMSEXT DMSINT
PREVEXEC	000001	DMSEXT
PREVIOUS	000017	DMSLBM DMSSBS DMSSOP DMSSQS DMSSVT
PREXIST	000006	DMSLDR DMSOLD
PRFPOFF	000009	DMSDBG DMSFRE DMSITS DMSQRY DMSSET
PRFTSYS	000006	DMSINS DMSITS DMSLDR DMSMOD DMSSLN
PRFUSYS	000005	DMSASM DMSITS DMSLDR DMSMOD DMSSLN
PRHOLD	000003	DMSLDR DMSLOA
PRINTER1	000001	DMSDBD

LABEL	COUNT	REFERENCES											
PRINTLST	000001	DMSSEB											
PROCERR	000004	DMSGRN	DMSLKD										
PROTFLAG	000020	DMSASM	DMSDBG	DMSFRE	DMSINS	DMSITS	DMSLDR	DMSMOD	DMSQRY	DMSSET	DMSSLN		
PRVCNT	000012	DMSLDR	DMSOLD										
PS	000019	DMSDSL	DMSFCH	DMSMVE	DMSROS	DMSSBD	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVN	DMSSVT
PSAVE	000011	DMSITP											
PSW	000003	DMSLDR											
PTR1	000015	DMSEDI	DMSEDX	DMSSCR	DMSUPD								
PTR2	000038	DMSEDI	DMSEDX	DMSSCR	DMSUPD								
PTR3	000008	DMSEDI	DMSEDX										
PUBADR	000017	DMSBOP	DMSCLS	DMSDLK	DMSDSV	DMSLLU	DMSPRV	DMSXCP					
PUBCUU	000013	DMSBOP	DMSCLS	DMSDLK	DMSDSV	DMSLLU	DMSPRV	DMSXCP					
PUBDEVT	000044	DMSBOP	DMSCLS	DMSDLK	DMSLLU	DMSXCP							
PUBDSKM	000002	DMSLLU	DMSXCP										
PUBPT	000017	DMSAMS	DMSASN	DMSBOP	DMSCLS	DMSDLB	DMSDLK	DMSDSV	DMSFCH	DMSLLU	DMSPRV	DMSRRV	DMSSET
		DMSSRV	DMSXCP										
PUBTAPM1	000005	DMSBOP	DMSCLS	DMSXCP									
PUBTAPM2	000016	DMSBOP											
PUBTAP7	000001	DMSBOP											
PUNCHLST	000001	DMSSEB											
PWAIT	000001	DMSPIO											
QQDSK1	000007	DMSACM	DMSDIO	DMSFNS	DMSFOR	DMSITI	DMSNUC						
QQDSK2	000007	DMSDIO											
QQTRK	000006	DMSDIO	DMSTQQ										
QS	000003	DMSNCP	DMSSOP										
QSWITCH	000003	DMSCRD	DMSINT										
RA	000047	DMSDLK											
RADD	000005	DMSLBT	DMSLGT	DMSLIB									
RANGE	000012	DMSEDI											
RDBUFF	000002	DMSSEB											
RDBUFLN	000001	DMSNCP											
RDBUFNO	000001	DMSNCP											
RDCCW	000001	DMSSEB											
RDCONS	000001	DMSINI											
RDCOUNT	000004	DMSPRV	DMSRRV	DMSSEB	DMSSRV								
RDDATA	000027	DMSINI	DMSPRV	DMSRRV	DMSSRV								
READ	000044	DMSBOP	DMSBRD	DMSCLS	DMSCMP	DMSDIO	DMSDLB	DMSDSK	DMSDSL	DMSDSV	DMSFCH	DMSRDC	DMSSBS
		DMSTPE											
READBLK	000003	DMSROS	DMSSVT										
READBUF	000031	DMSLDR	DMSLGT	DMSLIB	DMSNCP	DMSOLD							
READCNT	000015	DMSBRD	DMSEXT	DMSFCH									
READLST	000002	DMSDLK	DMSSEB										
REALTIMR	000006	DMSIOW	DMSITE	DMSSVN									
RECS	000002	DMSEDX											
REDERRID	000005	DMSCWR	DMSINT	DMSQRY	DMSSET								
REFCMD	000004	DMSLDR	DMSCLD										

LABEL	COUNT	REFERENCES											
REFLG1	000008	DMSLDR	DMSOLD										
REFLG2	000004	DMSLDR	DMSOLD										
REFLIB	000006	DMSLDR	DMSOLD										
REFUND	000004	DMSLDR	DMSOLD										
REGSAV	000025	DMSEDI	DMSINS	DMSUPD	DMSVSR								
REGSAVX	000007	DMSEDI											
REGSAV0	000030	DMSACF	DMSACM	DMSALU	DMSAUD	DMSLAD	DMSLFS						
REGSAV1	000012	DMSACF	DMSERS	DMSRNM									
REGSAV3	000036	DMSBRD	DMSBWR	DMSFNS	DMSMOD	DMSPNT	DMSSTT						
REG13SAV	000003	DMSLDR	DMSOLD										
RELPAGES	000020	DMSABN	DMSAMS	DMSARN	DMSARX	DMSASM	DMSCPY	DMSEDI	DMSINT	DMSLBM	DMSLBT	DMSLKD	DMSSRT
		DMSSTG	DMSUPD										
RELPHSE	000002	DMSFCH											
REPCNT	000010	DMSEDI	DMSEDX										
RESET	000103	DMSACC	DMSAMS	DMSARN	DMSARX	DMSASM	DMSBOP	DMSBTB	DMSBTP	DMSBWR	DMSCLS	DMSCPY	DMSDLB
		DMSDLK	DMSDSL	DMSDSV	DMSEDI	DMSFLD	DMSFOR	DMSIFC	DMSITE	DMSITP	DMSLBM	DMSLBT	DMSLDR
		DMSLDS	DMSLSB	DMSMVE	DMSOLD	DMSOPT	DMSPRV	DMSRRV	DMSSAB	DMSSCT	DMSSET	DMSSOP	DMSSRT
		DMSSRV	DMSSVT	DMSTPE	DMSUPD	DMSVIB	DMSVIP	DMSZAP					
RETREG	000009	DMSDIO	DMSLDR	DMSLST	DMSOLD								
RETRYBIT	000002	DMSSAB											
RETSAV	000006	DMSDBG	DMSVIP										
RETT	000005	DMSLSB											
RFIX	000001	DMSLGT											
RFPRS	000001	DMSOVS											
RGPRS	000007	DMSINS	DMSITS	DMSOVS	DMSSET								
RGPR11	000002	DMSITS											
RGPR8	000001	DMSOVS											
RITEM	000007	DMSLBT	DMSLGT	DMSLIB									
RLDCONST	000008	DMSLDR	DMSOLD										
RLENG	000002	DMSLGT	DMSLIB										
RMSGBUF	000011	DMSINT											
RMSROPEN	000001	DMSBOP											
RNUM	000002	DMSLGT	DMSLIB										
RPLACB	000003	DMSVIP											
RPLAREA	000001	DMSVIP											
RPLARG	000001	DMSVIP											
RPLASY	000002	DMSVIP											
RPLBUFL	000001	DMSVIP											
RPLCHAIN	000006	DMSVIP											
RPLECBPR	000004	DMSVIP											
RPLEOFDS	000001	DMSVIP											
RPLFDBKC	000003	DMSVIP											
RPLFLAG	000004	DMSVIP											
RPLIST	000005	DMSEDI	DMSRDC										
RPLKEYL	000001	DMSVIP											
RPLNUP	000001	DMSVIP											

```
LABEL      COUNT      REFERENCES

RPLOPT1    000004     DMSVIP
RPLOPT2    000001     DMSVIP
RPLRLEN    000001     DMSVIP
RPLRTNCD   000006     DMSVIP
RPLST      000002     DMSVIP
RPLSTRID   000001     DMSVIP
RPLUPD     000001     DMSVIP
RPLVLERR   000001     DMSVIP
RSTNPSW    000002     DMSDBG
RUN        000003     DMSCLS    DMSGRN
RWCCW      000003     DMSDIO
RWCNT      000004     DMSACF    DMSAUD    DMSMOD
RWFSTRG    000009     DMSAUD    DMSBRD    DMSBWR    DMSFNS
RWMFD      000010     DMSACM    DMSAUD
R0         002423     DMSABN    DMSACC    DMSACF    DMSACM    DMSALU    DMSAMS    DMSARE    DMSARN    DMSARX    DMSASM    DMSASN    DMSAUD
                      DMSBAB    DMSBOP    DMSBRD    DMSBTB    DMSBTP    DMSBWR    DMSCAT    DMSCIO    DMSCIT    DMSCLS    DMSCMP    DMSCPF
                      DMSCPY    DMSCRD    DMSCWR    DMSCWT    DMSDBD    DMSDBG    DMSDIO    DMSDLB    DMSDLK    DMSDMP    DMSDOS    DMSDSK
                      DMSDSL    DMSDSV    DMSEDC    DMSEDI    DMSEDX    DMSERR    DMSERS    DMSEXC    DMSEXT    DMSFCH    DMSFET    DMSFLD
                      DMSFNC    DMSFNS    DMSFOR    DMSFRE    DMSGIO    DMSGLB    DMSGND    DMSGRN    DMSHDI    DMSHDS    DMSIFC    DMSINA
                      DMSINI    DMSINM    DMSINS    DMSINT    DMSIOW    DMSITE    DMSITI    DMSITP    DMSITS    DMSLAD    DMSLAF    DMSLBM
                      DMSLBT    DMSLDR    DMSLDS    DMSLFS    DMSLGT    DMSLIB    DMSLIO    DMSLKD    DMSLLU    DMSLOA    DMSLSB    DMSLST
                      DMSLSY    DMSMDP    DMSMOD    DMSMVE    DMSNCP    DMSOLD    DMSOPL    DMSOPT    DMSOR1    DMSOVR    DMSOVS    DMSPNT
                      DMSPRT    DMSPRV    DMSPUN    DMSQRY    DMSRDC    DMSREA    DMSRNE    DMSRNM    DMSROS    DMSRRV    DMSSAB    DMSSBD
                      DMSSBS    DMSSCN    DMSSCR    DMSSCT    DMSSEB    DMSSET    DMSSMN    DMSSCP    DMSSQS    DMSSRT    DMSSRV    DMSSSK
                      DMSSTG    DMSSTT    DMSSVN    DMSSVT    DMSSYN    DMSTIC    DMSTMA    DMSTPD    DMSTPE    DMSTQQ    DMSTRK    DMSTYP
                      DMSUPD    DMSVIB    DMSVIP    DMSVPD    DMSVSR    DMSXCP    DMSZAP
R1         006574     DMSABN    DMSACC    DMSACF    DMSACM    DMSALU    DMSAMS    DMSARE    DMSARN    DMSARX    DMSASM    DMSASN    DMSAUD
                      DMSBAB    DMSBOP    DMSBRD    DMSBTB    DMSBTP    DMSBWR    DMSCAT    DMSCIO    DMSCIT    DMSCLS    DMSCMP    DMSCPF
                      DMSCPY    DMSCRD    DMSCWR    DMSCWT    DMSDBD    DMSDBG    DMSDIO    DMSDLB    DMSDLK    DMSDMP    DMSDOS    DMSDSK
                      DMSDSL    DMSDSV    DMSEDC    DMSEDI    DMSEDX    DMSERR    DMSERS    DMSEXC    DMSEXT    DMSFCH    DMSFET    DMSFLD
                      DMSFNS    DMSFOR    DMSFRE    DMSGIO    DMSGLB    DMSGND    DMSGRN    DMSHDI    DMSHDS    DMSIFC    DMSINA    DMSINI
                      DMSINM    DMSINS    DMSINT    DMSIOW    DMSITE    DMSITI    DMSITP    DMSITS    DMSLAD    DMSLAF    DMSLBM    DMSLBT
                      DMSLDR    DMSLDS    DMSLFS    DMSLGT    DMSLIB    DMSLIO    DMSLKD    DMSLLU    DMSLOA    DMSLSB    DMSLST    DMSLSY
                      DMSMDP    DMSMOD    DMSMVE    DMSNCP    DMSOLD    DMSOPL    DMSOPT    DMSOR1    DMSOR2    DMSOR3    DMSOVR    DMSOVS
                      DMSPIO    DMSPNT    DMSPRT    DMSPRV    DMSPUN    DMSQRY    DMSRDC    DMSREA    DMSRNE    DMSRNM    DMSROS    DMSRRV
                      DMSSAB    DMSSBD    DMSSBS    DMSSCN    DMSSCR    DMSSCT    DMSSEB    DMSSET    DMSSMN    DMSSOP    DMSSQS    DMSSRT
                      DMSSRV    DMSSSK    DMSSTG    DMSSTT    DMSSVN    DMSSVT    DMSSYN    DMSTIC    DMSTMA    DMSTPD    DMSTPE    DMSTQQ
                      DMSTRK    DMSTYP    DMSUPD    DMSVIB    DMSVIP    DMSVPD    DMSVSR    DMSXCP    DMSZAP
R10        001820     DMSACC    DMSACF    DMSACM    DMSALU    DMSAMS    DMSARE    DMSARN    DMSARX    DMSASM    DMSASN    DMSAUD    DMSBAB
                      DMSBOP    DMSBRD    DMSBTP    DMSBWR    DMSCIO    DMSCLS    DMSCMP    DMSCPF    DMSCPY    DMSCWR    DMSCWT    DMSDBD
                      DMSDBG    DMSDIO    DMSDLB    DMSDOS    DMSDSL    DMSDSV    DMSEDC    DMSEDI    DMSEDX    DMSERR    DMSERS    DMSEXC
                      DMSEXT    DMSFCH    DMSFLD    DMSFNS    DMSFOR    DMSFRE    DMSGIO    DMSGRN    DMSHDI    DMSHDS    DMSINT    DMSINM
                      DMSINS    DMSINT    DMSIOW    DMSITE    DMSITI    DMSITP    DMSITS    DMSLAD    DMSLBM    DMSLBT    DMSLDR    DMSLDS
                      DMSLFS    DMSLGT    DMSLIO    DMSLKD    DMSLLU    DMSLSB    DMSLST    DMSMOD    DMSMVE    DMSNCP    DMSOLD    DMSOPT
                      DMSPIO    DMSPRT    DMSPRV    DMSPUN    DMSQRY    DMSRDC    DMSRNE    DMSRNM    DMSROS    DMSRRV    DMSSAB    DMSSBD
                      DMSSCR    DMSSET    DMSSMN    DMSSOP    DMSSQS    DMSSRV    DMSSTG    DMSSTT    DMSSVN    DMSSVT    DMSTMA    DMSTPD
```

```
LABEL     COUNT     REFERENCES

                    DMSTPE DMSTRK DMSTYP DMSUPD DMSVIP DMSXCP DMSZAP
R11       000746    DMSACC DMSACF DMSACM DMSALU DMSAMS DMSARE DMSARN DMSARX DMSASM DMSASN DMSAUD DMSBOP
                    DMSBRD DMSBTP DMSBWR DMSCIO DMSCLS DMSCMP DMSCPY DMSCRD DMSCWR DMSCWT DMSDBD DMSDIO
                    DMSDLB DMSDOS DMSDSV DMSERS DMSEXC DMSFCH DMSFLD DMSFNS DMSFOR DMSFRE DMSGLB DMSGND
                    DMSGRN DMSINI DMSINS DMSINT DMSIOW DMSITE DMSITI DMSITP DMSITS DMSLAF DMSLBM DMSLBT
                    DMSLDR DMSLDS DMSLFS DMSLIB DMSLIO DMSLKD DMSLLU DMSLSB DMSLST DMSMOD DMSNCP DMSOLD
                    DMSOPT DMSPIO DMSPNT DMSPRT DMSPUN DMSQRY DMSRDC DMSRNM DMSROS DMSRRV DMSSAB DMSSBD
                    DMSSBS DMSSCR DMSSCT DMSSEB DMSSET DMSSOP DMSSQS DMSSVT DMSSYN DMSTIO DMSTMA DMSTPD
                    DMSTPE DMSTQQ DMSTRK DMSUPD DMSVIP DMSVPD DMSXCP DMSZAP
R12       000716    DMSABN DMSACC DMSACF DMSACM DMSALU DMSAMS DMSARE DMSARN DMSARX DMSASM DMSASN DMSAUD
                    DMSBAB DMSBOP DMSBRD DMSBTB DMSBTP DMSBWR DMSCAT DMSCIO DMSCIT DMSCLS DMSCMP DMSCPF
                    DMSCPY DMSCRD DMSCWR DMSCWT DMSDIO DMSDLB DMSDMP DMSDOS DMSDSL DMSDSV DMSEDX DMSERR
                    DMSERS DMSEXC DMSFCH DMSFET DMSFLD DMSFNS DMSFOR DMSFRE DMSGLB DMSGND DMSGRN DMSHDI
                    DMSHDS DMSIFC DMSINI DMSINS DMSINT DMSITE DMSITI DMSITP DMSITS DMSLAD DMSLAF DMSLBT
                    DMSLDR DMSLDS DMSLFS DMSLGT DMSLIB DMSLKD DMSLLU DMSLOA DMSLSB DMSLST DMSMOD DMSMVE
                    DMSNCP DMSOLD DMSOPL DMSOPT DMSOR1 DMSOR2 DMSOR3 DMSOVR DMSOVS DMSPIO DMSPNT DMSPRT
                    DMSPRV DMSPUN DMSQRY DMSREA DMSRNE DMSRNM DMSROS DMSRRV DMSSAB DMSSBD DMSSBS DMSSCN
                    DMSSCR DMSSCT DMSSET DMSSMN DMSSOP DMSSQS DMSSRT DMSSRV DMSSSK DMSSTG DMSSTT DMSSVN
                    DMSSVT DMSSYN DMSTIO DMSTMA DMSTPD DMSTPE DMSTQQ DMSTRK DMSUPD DMSVIB DMSVIP DMSVPD
                    DMSVSR DMSXCP DMSZAP
R13       000828    DMSABN DMSACC DMSACF DMSACM DMSALU DMSAMS DMSARN DMSARX DMSASM DMSASN DMSAUD DMSBAB
                    DMSBRD DMSBTP DMSBWR DMSCIO DMSCIT DMSCLS DMSCPY DMSCRD DMSCWR DMSDBG DMSDIO DMSDLB
                    DMSDOS DMSDSK DMSDSV DMSEDC DMSEDI DMSEDX DMSERR DMSERS DMSEXC DMSFCH DMSFLD DMSFNS
                    DMSFOR DMSFRE DMSGIO DMSGLB DMSGRN DMSHDI DMSHDS DMSIFC DMSINI DMSINS DMSINT DMSITE
                    DMSITI DMSITP DMSITS DMSLAD DMSLAF DMSLBT DMSLDR DMSLDS DMSLFS DMSLGT DMSLIB DMSLIO
                    DMSLSB DMSLST DMSMOD DMSMVE DMSNCP DMSOLD DMSOVS DMSPIO DMSPNT DMSPRT DMSPUN DMSQRY
                    DMSREA DMSRNE DMSRNM DMSSAB DMSSBS DMSSCR DMSSCT DMSSEB DMSSMN DMSSOP DMSSQS DMSSTG
                    DMSSTT DMSSVN DMSSVT DMSTIO DMSTPE DMSTRK DMSUPD DMSVIP DMSVSR DMSXCP DMSZAP
R14       003284    DMSABN DMSACC DMSACF DMSACM DMSALU DMSAMS DMSARE DMSARN DMSARX DMSASM DMSASN DMSAUD
                    DMSBAB DMSBOP DMSBRD DMSBTB DMSBTP DMSBWR DMSCAT DMSCIO DMSCIT DMSCLS DMSCMP DMSCPF
                    DMSCPY DMSCRD DMSCWR DMSCWT DMSDBD DMSDBG DMSDIO DMSDLB DMSDOS DMSDSK DMSDSL DMSDSV
                    DMSEDC DMSEDI DMSEDX DMSERR DMSERS DMSEXC DMSEXT DMSFCH DMSFET DMSFLD DMSFNS DMSFOR
                    DMSFRE DMSGIO DMSGLB DMSGND DMSGRN DMSHDI DMSHDS DMSIFC DMSINA DMSINI DMSINM DMSINS
                    DMSINT DMSIOW DMSITE DMSITI DMSITP DMSITS DMSLAD DMSLAF DMSLBM DMSLBT DMSLDR DMSLDS
                    DMSLFS DMSLGT DMSLIB DMSLIO DMSLKD DMSLLU DMSLOA DMSLSB DMSLST DMSLSY DMSMDP DMSMOD
                    DMSMVE DMSNCP DMSOLD DMSOPT DMSOR3 DMSOVR DMSOVS DMSPIO DMSPNT DMSPRT DMSPRV DMSPUN
                    DMSQRY DMSRDC DMSREA DMSRNE DMSRNM DMSROS DMSRRV DMSSAB DMSSBD DMSSBS DMSSCN DMSSCR
                    DMSSCT DMSSEB DMSSET DMSSMN DMSSOP DMSSQS DMSSRT DMSSRV DMSSSK DMSSTG DMSSTT DMSSVN
                    DMSSVT DMSSYN DMSTIO DMSTMA DMSTPD DMSTPE DMSTQQ DMSTRK DMSTYP DMSUPD DMSVIB DMSVIP
                    DMSVPD DMSVSR DMSXCP DMSZAP
R15       005371    DMSABN DMSACC DMSACF DMSACM DMSALU DMSAMS DMSARE DMSARN DMSARX DMSASM DMSASN DMSAUD
                    DMSBAB DMSBOP DMSBRD DMSBTB DMSBTP DMSBWR DMSCAT DMSCIO DMSCIT DMSCLS DMSCMP DMSCPF
                    DMSCPY DMSCRD DMSCWR DMSCWT DMSDBD DMSDBG DMSDIO DMSDLB DMSDOS DMSDSK DMSDSL DMSDSV
                    DMSEDC DMSEDI DMSEDX DMSERR DMSERS DMSEXC DMSEXT DMSFCH DMSFET DMSFLD DMSFNS DMSFOR
                    DMSFRE DMSGIO DMSGLB DMSGND DMSGRN DMSHDI DMSHDS DMSIFC DMSINA DMSINI DMSINM DMSINS
                    DMSINT DMSIOW DMSITE DMSITI DMSITP DMSITS DMSLAD DMSLAF DMSLBM DMSLBT DMSLDR DMSLDS
```

```
LABEL     COUNT     REFERENCES

                    DMSLFS  DMSLGT  DMSLIB  DMSLIO  DMSLKD  DMSLLU  DMSLOA  DMSLSB  DMSLST  DMSLSY  DMSMDP  DMSMOD
                    DMSMVE  DMSNCP  DMSOLD  DMSOPL  DMSOPT  DMSOR1  DMSOVR  DMSOVS  DMSPIO  DMSPNT  DMSPRT  DMSPRV
                    DMSPUN  DMSQRY  DMSRDC  DMSREA  DMSRNE  DMSRNM  DMSROS  DMSRRV  DMSSAB  DMSSBD  DMSSBS  DMSSCN
                    DMSSCR  DMSSCT  DMSSEB  DMSSET  DMSSMN  DMSSOP  DMSSQS  DMSSRT  DMSSRV  DMSSSK  DMSSTG  DMSSTT
                    DMSSVN  DMSSVT  DMSSYN  DMSTIO  DMSTMA  DMSTPD  DMSTPE  DMSTQQ  DMSTRK  DMSTYP  DMSUPD  DMSVIB
                    DMSVIP  DMSVPD  DMSVSR  DMSXCP  DMSZAP
R2        003771    DMSABN  DMSACC  DMSACF  DMSACM  DMSALU  DMSAMS  DMSARE  DMSARN  DMSARX  DMSASM  DMSASN  DMSAUD
                    DMSBAB  DMSBOP  DMSBRD  DMSBTB  DMSBTP  DMSBWR  DMSCAT  DMSCIC  DMSCIT  DMSCLS  DMSCMP  DMSCPF
                    DMSCPY  DMSCRD  DMSCWR  DMSDBD  DMSDBG  DMSDIC  DMSDLB  DMSDLK  DMSDMP  DMSDOS  DMSDSK  DMSDSL
                    DMSDSV  DMSEDC  DMSEDI  DMSEDX  DMSERR  DMSERS  DMSEXC  DMSEXT  DMSFCH  DMSFET  DMSFLD  DMSFNS
                    DMSFOR  DMSFRE  DMSGIO  DMSGLB  DMSGND  DMSGRN  DMSHDI  DMSHDS  DMSIFC  DMSINA  DMSINI  DMSINM
                    DMSINS  DMSINT  DMSIOW  DMSITE  DMSITP  DMSITS  DMSLAD  DMSLAF  DMSLBM  DMSLBT  DMSLDR  DMSLDS
                    DMSLFS  DMSLIO  DMSLKD  DMSLLU  DMSLOA  DMSLSB  DMSLST  DMSMDP  DMSMOD  DMSMVE  DMSNCP  DMSOLD
                    DMSOPL  DMSOPT  DMSOR1  DMSPIO  DMSPNT  DMSPRT  DMSPRV  DMSPUN  DMSQRY  DMSRDC  DMSREA  DMSRNE
                    DMSRNM  DMSROS  DMSRRV  DMSSAB  DMSSBD  DMSSBS  DMSSCN  DMSSCR  DMSSCT  DMSSEB  DMSSET  DMSSMN
                    DMSSOP  DMSSQS  DMSSRT  DMSSRV  DMSSSK  DMSSTG  DMSSTT  DMSSVN  DMSSVT  DMSSYN  DMSTMA  DMSTPD
                    DMSTPE  DMSTQQ  DMSTRK  DMSTYP  DMSUPD  DMSVIB  DMSVIP  DMSVPD  DMSVSR  DMSXCP  DMSZAP
R3        003780    DMSABN  DMSACC  DMSACF  DMSACM  DMSALU  DMSAMS  DMSARE  DMSARN  DMSARX  DMSASM  DMSASN  DMSAUD
                    DMSBAB  DMSBOP  DMSBRD  DMSBTB  DMSBTP  DMSBWR  DMSCAT  DMSCIC  DMSCIT  DMSCLS  DMSCMP  DMSCPF
                    DMSCPY  DMSCRD  DMSCWR  DMSDBD  DMSDBG  DMSDLB  DMSDLK  DMSDMP  DMSDOS  DMSDSK  DMSDSL  DMSDSV
                    DMSEDC  DMSEDI  DMSEDX  DMSERR  DMSERS  DMSEXC  DMSEXT  DMSFCH  DMSFET  DMSFLD  DMSFOR  DMSFRE
                    DMSGIO  DMSGLB  DMSGND  DMSGRN  DMSHDI  DMSHDS  DMSIFC  DMSINA  DMSINI  DMSINM  DMSINS  DMSINT
                    DMSITE  DMSITI  DMSITP  DMSITS  DMSLAD  DMSLAF  DMSLBM  DMSLBT  DMSLDR  DMSLDS  DMSLFS  DMSLGT
                    DMSLIO  DMSLKD  DMSLLU  DMSLSB  DMSLST  DMSMDP  DMSMOD  DMSMVE  DMSNCP  DMSOLD  DMSOPL  DMSOVR
                    DMSOVS  DMSPIO  DMSPRT  DMSPRV  DMSPUN  DMSQRY  DMSRDC  DMSREA  DMSRNE  DMSRNM  DMSROS  DMSRRV
                    DMSSAB  DMSSBD  DMSSBS  DMSSCN  DMSSCR  DMSSCT  DMSSEB  DMSSET  DMSSMN  DMSSOP  DMSSQS  DMSSRT
                    DMSSRV  DMSSSK  DMSSTG  DMSSTT  DMSSVN  DMSSVT  DMSSYN  DMSTMA  DMSTPD  DMSTPE  DMSTRK  DMSTYP
                    DMSUPD  DMSVIB  DMSVIP  DMSVPD  DMSVSR  DMSXCP  DMSZAP
R4        002961    DMSABN  DMSACC  DMSACF  DMSACM  DMSALU  DMSAMS  DMSARE  DMSARN  DMSARX  DMSASM  DMSASN  DMSAUD
                    DMSBAB  DMSBOP  DMSBRD  DMSBTB  DMSBTP  DMSBWR  DMSCAT  DMSCIC  DMSCIT  DMSCLS  DMSCMP  DMSCPF
                    DMSCPY  DMSCRD  DMSCWR  DMSDBD  DMSDBG  DMSDIC  DMSDLB  DMSDLK  DMSDMP  DMSDOS  DMSDSK  DMSDSL
                    DMSDSV  DMSEDC  DMSEDI  DMSEDX  DMSERR  DMSERS  DMSEXC  DMSEXT  DMSFCH  DMSFET  DMSFLD  DMSFOR
                    DMSFRE  DMSGIO  DMSGLB  DMSGND  DMSGRN  DMSHDI  DMSHDS  DMSIFC  DMSINA  DMSINI  DMSINM  DMSINS
                    DMSINT  DMSIOW  DMSITI  DMSITP  DMSITS  DMSLAD  DMSLAF  DMSLBM  DMSLBT  DMSLDR  DMSLDS  DMSLFS
                    DMSLGT  DMSLIO  DMSLKD  DMSLLU  DMSLSB  DMSLST  DMSMDP  DMSMOD  DMSMVE  DMSNCP  DMSOLD  DMSOPL
                    DMSOVR  DMSOVS  DMSPIO  DMSPNT  DMSPRT  DMSPUN  DMSQRY  DMSRDC  DMSREA  DMSRNE  DMSRNM  DMSROS
                    DMSRRV  DMSSAB  DMSSBD  DMSSBS  DMSSCN  DMSSCR  DMSSCT  DMSSET  DMSSMN  DMSSOP  DMSSQS  DMSSRT
                    DMSSRV  DMSSSK  DMSSTG  DMSSTT  DMSSVN  DMSSVT  DMSSYN  DMSTMA  DMSTPD  DMSTPE  DMSTQQ  DMSTRK
                    DMSTYP  DMSUPD  DMSVIP  DMSVPD  DMSVSR  DMSXCP  DMSZAP
R5        003094    DMSABN  DMSACC  DMSACF  DMSACM  DMSALU  DMSAMS  DMSARE  DMSARN  DMSARX  DMSASM  DMSASN  DMSAUD
                    DMSBAB  DMSBOP  DMSBRD  DMSBTB  DMSBTP  DMSBWR  DMSCIO  DMSCIT  DMSCLS  DMSCMP  DMSCPF  DMSCPY
                    DMSCRD  DMSCWR  DMSDBD  DMSDBG  DMSDIO  DMSDLB  DMSDLK  DMSDMP  DMSDOS  DMSDSK  DMSDSL  DMSDSV
                    DMSEDC  DMSEDI  DMSEDX  DMSERR  DMSERS  DMSEXC  DMSEXT  DMSFCH  DMSFET  DMSFLD  DMSFNS  DMSFOR
                    DMSFRE  DMSGIO  DMSGLB  DMSGND  DMSGRN  DMSHDI  DMSHDS  DMSIFC  DMSINA  DMSINI  DMSINM  DMSINS
                    DMSINT  DMSIOW  DMSITI  DMSITP  DMSITS  DMSLAD  DMSLAF  DMSLBM  DMSLBT  DMSLDR  DMSLDS  DMSLFS
                    DMSLGT  DMSLIB  DMSLKD  DMSLLU  DMSLSB  DMSLST  DMSMOD  DMSMVE  DMSNCP  DMSOLD  DMSOPL  DMSOR1
```

```
LABEL     COUNT     REFERENCES

                    DMSOVR  DMSOVS  DMSPIO  DMSPNT  DMSPRT  DMSPUN  DMSQRY  DMSRDC  DMSREA  DMSRNE  DMSRNM  DMSROS
                    DMSRRV  DMSSAB  DMSSBD  DMSSBS  DMSSCN  DMSSCR  DMSSCT  DMSSET  DMSSMN  DMSSOP  DMSSQS  DMSSRT
                    DMSSRV  DMSSSK  DMSSTG  DMSSTT  DMSSVN  DMSSVT  DMSSYN  DMSTMA  DMSTPD  DMSTPE  DMSTRK  DMSTYP
                    DMSUPD  DMSVIB  DMSVIP  DMSVPD  DMSVSR  DMSXCP  DMSZAP
R6        002670    DMSABN  DMSACC  DMSACF  DMSACM  DMSALU  DMSAMS  DMSARE  DMSARN  DMSARX  DMSASM  DMSASN  DMSAUD
                    DMSBAB  DMSBOP  DMSBRD  DMSBTP  DMSBWR  DMSCIC  DMSCIT  DMSCLS  DMSCMP  DMSCPF  DMSCPY  DMSCRD
                    DMSCWR  DMSDBD  DMSDBG  DMSDIO  DMSDLB  DMSDLK  DMSDMP  DMSDOS  DMSDSK  DMSDSV  DMSEDC  DMSEDI
                    DMSEDX  DMSERR  DMSERS  DMSEXC  DMSEXT  DMSFCH  DMSFET  DMSFLD  DMSFNS  DMSFOR  DMSFRE  DMSGND
                    DMSGRN  DMSHDI  DMSHDS  DMSIFC  DMSINA  DMSINI  DMSINS  DMSINT  DMSIOW  DMSITI  DMSITP  DMSITS
                    DMSLAD  DMSLBM  DMSLBT  DMSLDR  DMSLDS  DMSLFS  DMSLGT  DMSLKD  DMSLLU  DMSLOA  DMSLSB  DMSLST
                    DMSMOD  DMSMVE  DMSNCP  DMSOLD  DMSOPL  DMSOR1  DMSOVR  DMSOVS  DMSPIO  DMSPNT  DMSPRT  DMSPUN
                    DMSQRY  DMSRDC  DMSREA  DMSRNE  DMSRNM  DMSROS  DMSRRV  DMSSAB  DMSSBD  DMSSBS  DMSSCN  DMSSCR
                    DMSSCT  DMSSET  DMSSMN  DMSSOP  DMSSQS  DMSSRT  DMSSSK  DMSSTG  DMSSTT  DMSSVN  DMSSVT  DMSSYN
                    DMSTMA  DMSTPD  DMSTPE  DMSTQQ  DMSTRK  DMSTYP  DMSUPD  DMSVIP  DMSVPD  DMSVSR  DMSXCP  DMSZAP
R7        002449    DMSABN  DMSACC  DMSACF  DMSACM  DMSALU  DMSAMS  DMSARE  DMSARN  DMSARX  DMSASM  DMSASN  DMSAUD
                    DMSBOP  DMSBRD  DMSBTP  DMSBWR  DMSCIO  DMSCIT  DMSCLS  DMSCMP  DMSCPF  DMSCPY  DMSCWR  DMSDBD
                    DMSDBG  DMSDIO  DMSDLB  DMSDLK  DMSDMP  DMSDOS  DMSDSK  DMSDSV  DMSEDC  DMSEDI  DMSEDX  DMSERR
                    DMSERS  DMSEXC  DMSEXT  DMSFCH  DMSFET  DMSFLD  DMSFNS  DMSFOR  DMSFRE  DMSGLB  DMSGRN  DMSHDI
                    DMSHDS  DMSIFC  DMSINA  DMSINI  DMSINS  DMSINT  DMSIOW  DMSITE  DMSITI  DMSITP  DMSITS  DMSLAD
                    DMSLBM  DMSLBT  DMSLDR  DMSLDS  DMSLFS  DMSLGT  DMSLIB  DMSLKD  DMSLLU  DMSLSB  DMSLST  DMSMOD
                    DMSMVE  DMSOLD  DMSOPL  DMSOVR  DMSOVS  DMSPIO  DMSPRT  DMSPUN  DMSQRY  DMSRDC  DMSREA  DMSRNE
                    DMSRNM  DMSROS  DMSRRV  DMSSAB  DMSSBD  DMSSCN  DMSSCR  DMSSCT  DMSSET  DMSSMN  DMSSOP  DMSSQS
                    DMSSTG  DMSSVT  DMSSYN  DMSTMA  DMSTPD  DMSTPE  DMSTRK  DMSTYP  DMSUPD  DMSVIP  DMSVPD  DMSVSR
                    DMSXCP  DMSZAP
R8        002110    DMSABN  DMSACC  DMSACF  DMSACM  DMSALU  DMSAMS  DMSARE  DMSARN  DMSARX  DMSASM  DMSASN  DMSAUD
                    DMSBAB  DMSBOP  DMSBRD  DMSBTB  DMSBTP  DMSBWR  DMSCIO  DMSCIT  DMSCLS  DMSCMP  DMSCPF  DMSCPY
                    DMSCRD  DMSCWR  DMSDBD  DMSDBG  DMSDIO  DMSDLB  DMSDLK  DMSDOS  DMSDSK  DMSDSL  DMSDSV  DMSEDC
                    DMSEDI  DMSEDX  DMSERR  DMSERS  DMSEXC  DMSEXT  DMSFCH  DMSFLD  DMSFNS  DMSFOR  DMSFRE  DMSGLB
                    DMSGRN  DMSHDI  DMSHDS  DMSIFC  DMSINA  DMSINI  DMSINM  DMSIOW  DMSITE  DMSITI  DMSITP  DMSITS
                    DMSLAD  DMSLBM  DMSLBT  DMSLDR  DMSLDS  DMSLFS  DMSLGT  DMSLLU  DMSLSB  DMSLST  DMSMOD  DMSMVE
                    DMSNCP  DMSOLD  DMSOPL  DMSOVR  DMSOVS  DMSPIO  DMSPRT  DMSPUN  DMSQRY  DMSRDC  DMSRNM  DMSROS
                    DMSRRV  DMSSAB  DMSSBD  DMSSBS  DMSSCN  DMSSCT  DMSSEB  DMSSET  DMSSMN  DMSSOP  DMSSSK  DMSSTG
                    DMSSVN  DMSSVT  DMSSYN  DMSTMA  DMSTPD  DMSTPE  DMSTRK  DMSTYP  DMSUPD  DMSVIP  DMSVSR  DMSXCP
                    DMSZAP
R9        001869    DMSABN  DMSACC  DMSACF  DMSACM  DMSALU  DMSAMS  DMSARE  DMSARN  DMSARX  DMSASM  DMSASN  DMSAUD
                    DMSBAB  DMSBOP  DMSBRD  DMSBTP  DMSBWR  DMSCIT  DMSCLS  DMSCMP  DMSCPF  DMSCPY  DMSCRD  DMSCWR
                    DMSCWT  DMSDBD  DMSDBG  DMSDIO  DMSDLB  DMSDLK  DMSDOS  DMSDSK  DMSDSV  DMSEDC  DMSEDI  DMSEDX
                    DMSERR  DMSERS  DMSEXC  DMSEXT  DMSFCH  DMSFLD  DMSFNS  DMSFOR  DMSFRE  DMSGIO  DMSGND  DMSGRN
                    DMSHDI  DMSHDS  DMSIFC  DMSINA  DMSINI  DMSINS  DMSINT  DMSIOW  DMSITI  DMSITP  DMSITS  DMSLAD
                    DMSLBM  DMSLBT  DMSLDR  DMSLDS  DMSLFS  DMSLGT  DMSLKD  DMSLSB  DMSLST  DMSMOD  DMSMVE  DMSNCP
                    DMSOLD  DMSOPL  DMSPIO  DMSPRT  DMSPUN  DMSQRY  DMSRDC  DMSRNM  DMSROS  DMSRRV  DMSSAB  DMSSBD
                    DMSSCR  DMSSCT  DMSSET  DMSSMN  DMSSOP  DMSSRV  DMSSSK  DMSSTG  DMSSTT  DMSSVT  DMSTMA  DMSTPD
                    DMSTPE  DMSTRK  DMSTYP  DMSUPD  DMSVIP  DMSVPD  DMSXCP  DMSZAP
SAVCNT    000005    DMSEDI  DMSSCR
SAVCWD    000022    DMSEDI
SAVE      000015    DMSCMP  DMSEDI  DMSGRN  DMSLBT  DMSNCP  DMSRDC
```

LABEL	COUNT	REFERENCES										
SAVEADT	000002	DMSDIO										
SAVEAR	000010	DMSEDC	DMSSCR									
SAVEREGS	000040	DMSASM	DMSROS									
SAVER0	000021	DMSDSV	DMSIFC	DMSREA	DMSVIP							
SAVER1	000048	DMSIFC	DMSREA	DMSSOP	DMSTPE	DMSVIP						
SAVER10	000002	DMSTMA										
SAVER14	000059	DMSIFC	DMSREA	DMSSCT	DMSSEB	DMSTPE	DMSVIP					
SAVER15	000013	DMSIFC	DMSREA	DMSSOP								
SAVER2	000011	DMSREA	DMSVIP									
SAVESIZE	000001	DMSZAP										
SAVEXT	000002	DMSDLB	DMSITE									
SAVE1	000020	DMSBOP	DMSDBD	DMSDBG	DMSDSL	DMSRRV	DMSSRV					
SAVE2	000021	DMSBOP	DMSDBG	DMSIFC								
SAV67	000006	DMSLDR	DMSOLD									
SCAW	000003	DMSDBG	DMSITE									
SCBPTR	000015	DMSITP	DMSSAB	DMSSLN	DMSSTG	DMSSVT						
SCBSAV12	000004	DMSSAB										
SCBWORK	000006	DMSSAB	DMSSTG									
SCLNO	000002	DMSSCR										
SCRBUFAD	000002	DMSEDX	DMSSCR									
SCRFLGS	000036	DMSEDI	DMSSCR									
SCRFLG2	000019	DMSEDI	DMSSCR									
SDISK	000005	DMSALU	DMSINI	DMSINS	DMSNUC							
SEARCH	000035	DMSFCH	DMSINI	DMSLIB	DMSLST	DMSMOD	DMSPRV	DMSQRY	DMSRRV	DMSSET	DMSSRV	DMSSVT
SEBSAV	000009	DMSSBD	DMSSEB									
SECTNUM	000006	DMSACM	DMSDIO	DMSFNS	DMSFOR	DMSITI	DMSNUC					
SEEK	000037	DMSDSV	DMSFCH	DMSINI	DMSOPL	DMSPRV	DMSROS	DMSRRV	DMSSET	DMSSRV	DMSXCP	
SEEKADR	000013	DMSACM	DMSDIO	DMSFNS	DMSFOR	DMSITI	DMSNUC					
SENCCW	000002	DMSDIO	DMSPIO									
SENSB	000008	DMSACM	DMSDIO	DMSFNS	DMSFOR	DMSITI	DMSNUC					
SENSE	000019	DMSBOP	DMSCLS	DMSFOR	DMSPRV	DMSRRV	DMSSRV					
SEQNAME	000004	DMSEDI	DMSEDX									
SERSAV	000002	DMSEDI										
SERTSEQ	000003	DMSEDI										
SERTSW	000003	DMSEDI										
SETLIB	000002	DMSLIB										
SETSEC	000002	DMSINI										
SETUP	000013	DMSSAB										
SETUP2	000002	DMSSAB										
SF	000007	DMSDLK	DMSDSL	DMSFCH	DMSNCP							
SFLAG	000009	DMSITS										
SFNUC	000002	DMSITS										
SFREN	000001	DMSITS										
SFSYS	000005	DMSITS										
SFTRN	000002	DMSITS										
SIGNAL	000057	DMSACM	DMSEDI	DMSERS								

LABEL	COUNT	REFERENCES											
SILI	000209	DMSDBD	DMSDBG	DMSFOR	DMSINI	DMSINS	DMSITE	DMSNUC	DMSPIC	DMSTIO	DMSXCP		
SIZE	000022	DMSFRE	DMSLKD										
SKEY	000003	DMSFRE	DMSSBD										
SKIP	000010	DMSBOP	DMSEXT	DMSROS	DMSSRT	DMSXCP							
SM	000001	DMSERR											
SOB1	000002	DMSOPT	DMSSET										
SPARES	000015	DMSEDI	DMSEDX	DMSUPD									
SPEC	000198	DMSLDR	DMSLGT	DMSLIB	DMSOLD								
SPECLF	000002	DMSINS	DMSINT										
SPIESAV	000002	DMSINT											
SSAVE	000060	DMSABN	DMSACC	DMSBAB	DMSDBG	DMSDLB	DMSDOS	DMSERR	DMSFLD	DMSFRE	DMSIFC	DMSITP	DMSITS
		DMSLDR	DMSOVS	DMSSAB	DMSSLN	DMSSMN	DMSSOP	DMSSTG	DMSSVN	DMSSVT	DMSVIP	DMSXCP	
SSAVENXT	000004	DMSITS											
SSAVEPRV	000008	DMSITS	DMSSAB	DMSVIP									
SSAVESZ	000006	DMSITS											
STACKAT	000002	DMSEDI											
STACKATL	000005	DMSEDI											
STAEBIT	000003	DMSSAB											
STAESAV	000002	DMSINT											
STAIBIT	000002	DMSSAB											
STARS	000001	DMSINT											
START	000023	DMSFET	DMSFNC	DMSFOR	DMSGRN	DMSITS	DMSLDR	DMSLSB	DMSOVS	DMSTYP			
STATEFST	000022	DMSALU	DMSBRD	DMSERS	DMSFNS	DMSGND	DMSINT	DMSPUN	DMSRNM	DMSSTT			
STATER0	000003	DMSBRD	DMSSOP	DMSSTT									
STATER1	000005	DMSDSK	DMSERS										
STIMEXIT	000009	DMSITE	DMSSTG	DMSSVN	DMSSVT								
STOP	000006	DMSTPD											
STOPAT	000002	DMSDBG											
STRTADDR	000034	DMSFET	DMSITS	DMSLDR	DMSLOA	DMSLSB	DMSMOD	DMSOLD	DMSSET	DMSSLN			
STRTNO	000005	DMSEDI	DMSRNE										
SUBACT	000004	DMSEDX	DMSINT	DMSLOA	DMSSLN								
SUBFLAG	000028	DMSABN	DMSEDX	DMSEXT	DMSFNS	DMSINT	DMSLOA	DMSMOD	DMSSLN				
SUBINIT	000001	DMSFNS											
SUBREJ	000003	DMSEDX	DMSINT										
SUBSECT	000004	DMSABN	DMSINM	DMSINT									
SVC$202	000004	DMSEXT											
SVCAB	000008	DMSFRE	DMSITS										
SVCOPSW	000026	DMSITS											
SVCOUNT	000003	DMSITS	DMSOVS										
SVCSAVE	000012	DMSITS											
SVCSECT	000021	DMSCIT	DMSFRE	DMSHDS	DMSINT	DMSITE	DMSITS	DMSLAD	DMSLFS	DMSOVR	DMSOVS	DMSSLN	
SVCSTOP	000001	DMSITS											
SVC12SAV	000004	DMSDOS											
SVEARA	000007	DMSBAB	DMSDOS	DMSITP									
SVEPSW	000007	DMSBAB	DMSDOS	DMSITP									
SVEPSW2	000008	DMSBAB	DMSDOS	DMSITP									

```
LABEL    COUNT     REFERENCES

SVEROF   000004    DMSBAB   DMSDOS
SVER00   000015    DMSBAB   DMSDOS   DMSITP
SVER01   000001    DMSBAB
SVER09   000009    DMSBAB   DMSDOS   DMSITP
SVLAD    000006    DMSLAD
SVLADW   000003    DMSLAD
SVLFS    000006    DMSLFS
SWTCH    000001    DMSACM
SWTCHSAV 000002    DMSINT
SYMTABLE 000003    DMSDBG
SYMTBG   000004    DMSDBG
SYSADDR  000003    DMSINI
SYSCODE  000005    DMSDLB   DMSFRE   DMSSET
SYSCOM   000017    DMSBAB   DMSBOP   DMSDOS   DMSFET   DMSITP   DMSQRY   DMSSTG   DMSSYN
SYSLINE  000003    DMSDLK   DMSQRY   DMSSET
SYSLOAD  000010    DMSACM   DMSINS   DMSLDR   DMSLSB   DMSCLD   DMSSET
SYSNAME  000006    DMSETP   DMSINS
SYSNAMES 000037    DMSAMS   DMSBOP   DMSBTP   DMSDOS   DMSEDX   DMSEXC   DMSINS   DMSINT   DMSITS   DMSQRY   DMSSFT   DMSVIB
                   DMSVSR
SYSNEND  000014    DMSAMS   DMSBOP   DMSBTP   DMSDOS   DMSEDX   DMSEXC   DMSINS   DMSINT   DMSITS   DMSQRY   DMSSET   DMSVIB
                   DMSVSR
SYSREF   000004    DMSINS   DMSLOA   DMSSET
SYSTEM   000012    DMSASN   DMSDLB   DMSMOD   DMSSET   DMSSLN   DMSSSK   DMSXCP
SYSTEMID 000005    DMSINI   DMSINS
SYSUT1   000027    DMSARX   DMSASM   DMSDLK   DMSLDR   DMSLKD   DMSCLD
TABEND   000007    DMSFLD   DMSZAP
TABLIN   000016    DMSEDI   DMSSCR
TABS     000023    DMSEDI   DMSEDX
TAIEIAD  000002    DMSCIT
TAIEMSGL 000001    DMSCIT
TAIERSAV 000002    DMSCIT
TAPE     000017    DMSCLS   DMSLLU   DMSSEB   DMSTIO   DMSIMA   DMSTPE   DMSXCP
TAPEBUFF 000001    DMSSEB
TAPECOUT 000002    DMSSEB
TAPEDEV  000003    DMSSBS   DMSSEB   DMSSOP
TAPELIST 000003    DMSSBS   DMSSEB   DMSSOP
TAPEMASK 000003    DMSSBS   DMSSEB   DMSSOP
TAPEOPER 000010    DMSSBS   DMSSEB   DMSSOP
TAPESIZE 000002    DMSSEB
TAPE1    000002    DMSASN
TAPE4    000002    DMSASN
TAXEADDR 000010    DMSCIT   DMSITE   DMSITI   DMSSTG   DMSSVT
TAXEDEF  000001    DMSSVT
TAXEEXIT 000002    DMSCIT   DMSSVT
TAXEEXTS 000001    DMSCIT
TAXEFREQ 000006    DMSCIT   DMSITE   DMSITI
```

```
LABEL     COUNT     REFERENCES


TAXEIOL   000003    DMSCIT   DMSITI
TAXEIOWS  000002    DMSCIT
TAXELNK   000006    DMSCIT   DMSITE   DMSITI   DMSSVT
TAXERTNA  000002    DMSCIT
TAXESTAT  000005    DMSCIT   DMSITE   DMSITI
TAXETAIE  000002    DMSCIT
TAXETSOF  000002    DMSCIT
TBENT     000028    DMSACM   DMSBTB   DMSFET   DMSGND   DMSLDR   DMSLOA   DMSMDP   DMSMOD   DMSOLD   DMSSET   DMSSLN
TBLCT     000019    DMSLDR   DMSLIB   DMSOLD
TBLEND    000004    DMSDBD   DMSDBG   DMSITE   DMSNUC
TBLLNGTH  000005    DMSSBD   DMSSVT
TBLREF    000020    DMSLDR   DMSLIB   DMSOLD
TCODE     000001    DMSFRE
TEMPBYTE  000003    DMSSVT
TEMPSAVE  000014    DMSBOP   DMSUPD
TEMPST    000008    DMSLDR   DMSOLD
TEMPTAB   000004    DMSEDI
TEMP02    000002    DMSITS
TEXT      000553    DMSABN   DMSACC   DMSACM   DMSAMS   DMSARE   DMSARN   DMSARX   DMSASM   DMSASN   DMSBOP   DMSBTB   DMSBTP
                    DMSBWR   DMSCIT   DMSCLS   DMSCMP   DMSCPY   DMSCRD   DMSCWR   DMSDBD   DMSDBG   DMSDIO   DMSDLB   DMSDLK
                    DMSDMP   DMSDOS   DMSDSK   DMSDSL   DMSDSV   DMSEDI   DMSEDX   DMSEXC   DMSFCH   DMSFET   DMSFLD   DMSFNS
                    DMSFOR   DMSGLB   DMSGND   DMSGRN   DMSIFC   DMSINS   DMSITS   DMSLBM   DMSLBT   DMSLDR   DMSLDS   DMSLGT
                    DMSLIO   DMSLKD   DMSLLU   DMSLOA   DMSMDP   DMSMOD   DMSMVE   DMSNCP   DMSOPL   DMSOPT   DMSOR1   DMSOVR
                    DMSOVS   DMSPRV   DMSQRY   DMSRDC   DMSREA   DMSRNE   DMSRNM   DMSRCS   DMSRRV   DMSSCR   DMSSET   DMSSLN
                    DMSSMN   DMSSRT   DMSSRV   DMSSSK   DMSSTT   DMSSYN   DMSTMA   DMSTPD   DMSTPE   DMSTYP   DMSUPD   DMSVIB
                    DMSVIP   DMSVPD   DMSXCP   DMSZAP
TEXTA     000058    DMSACC   DMSAMS   DMSBWR   DMSCIO   DMSDLK   DMSDOS   DMSERS   DMSGRN   DMSLBM   DMSLBT   DMSLST   DMSMOD
                    DMSOVS   DMSPIO   DMSPRT   DMSPUN   DMSSVT   DMSUPD
TEXT3     000001    DMSSVT
TIC       000054    DMSDSV   DMSFCH   DMSINI   DMSOPL   DMSPRV   DMSRRV   DMSSET   DMSSRV   DMSXCP
TIMBUF    000013    DMSEXT   DMSINM   DMSSVT
TIMCCW    000005    DMSITE   DMSQRY   DMSSET
TIMCHAR   000024    DMSINS   DMSINT   DMSIOW   DMSITE   DMSNUC   DMSQRY   DMSSET   DMSSMN   DMSSTG   DMSSVN   DMSSVT
TIMER     000016    DMSINS   DMSINT   DMSIOW   DMSITE   DMSSET   DMSSVN   DMSSVT
TIMINIT   000011    DMSINS   DMSINT   DMSIOW   DMSITE   DMSSET   DMSSVN
TIN       000004    DMSEDI   DMSEDX
TMPLOC    000008    DMSLDR   DMSLSB   DMSOLD
TOOBIG    000003    DMSDIO
TOTLIBS   000003    DMSGLB   DMSSMN
TOUT      000004    DMSEDI
TPFACB    000004    DMSSOP
TPFERT    000003    DMSITS
TPFNS     000009    DMSITS
TPFR01    000002    DMSITS
TPFSVO    000005    DMSDOS   DMSITS   DMSOVS   DMSVIP
TPFUSR    000011    DMSDBG   DMSITP   DMSITS   DMSLDR   DMSSAB
```

LABEL	COUNT	REFERENCES
TRAP	000002	DMSFNC DMSITE
TRKLSAVE	000002	DMSTQQ
TRNCNUM	000006	DMSEDI
TRNCODE	000001	DMSFRE
TRUN	000001	DMSOR1
TRUNCOL	000016	DMSEDI DMSEDX DMSSCR
TSOATCNL	000017	DMSCIT DMSCRD DMSITE DMSITI DMSITS DMSSEB DMSSVN
TSOBLKS	000001	DMSSET
TSOFLAGS	000017	DMSCIT DMSCRD DMSITE DMSITI DMSITS DMSSEB DMSSVN
TSYM	000005	DMSDBG
TVERCOL1	000002	DMSEDI
TVERCOL2	000001	DMSEDI
TWITCH	000088	DMSEDI DMSEDX DMSSCR
TXLIBSV	000004	DMSGLB
TXTDIRC	000009	DMSGLB DMSIFC DMSLDR DMSLGT DMSLIB DMSOLD
TXTLIBS	000005	DMSGLB DMSIFC DMSLGT DMSLIB DMSQRY
TYPE	000092	DMSACC DMSACF DMSACM DMSAUD DMSBOP DMSBRD DMSBTB DMSBWR DMSCAT DMSCLS DMSCMP DMSCPY DMSDIO DMSDLK DMSDMP DMSDSK DMSDSV DMSEDI DMSEDX DMSERS DMSEXC DMSFLD DMSFNS DMSFOR DMSFRE DMSIFC DMSINA DMSINS DMSINT DMSITE DMSITP DMSITS DMSLAD DMSLAF DMSLFS DMSLGT DMSLIB DMSLIO DMSLOA DMSLSB DMSLST DMSOPL DMSOR1 DMSOVR DMSOVS DMSRNE DMSROS DMSSAB DMSSCR DMSSEB DMSSET DMSSOP DMSSVT DMSSYN DMSUPD DMSVIB DMSVIP DMSXCP DMSZAP
TYPEAD	000001	DMSLIO
TYPFLAG	000034	DMSDBG DMSDOS DMSITP DMSITS DMSLDR DMSOVS DMSSAB DMSSCP DMSVIP
TYPFLG	000002	DMSEDI
TYPLIN	000040	DMSEXT DMSFNC DMSLBT DMSLIO DMSTYP
TYPLIST	000007	DMSEXT DMSITE DMSTMA
TYPPUN	000001	DMSPUN
TYPRDR	000001	DMSRDC
TYPSCR	000009	DMSEDX DMSSCR
TYP1403	000002	DMSASN DMSPRT
TYP2305	000001	DMSINI
TYP2311	000001	DMSINI
TYP2314	000006	DMSASN DMSBOP DMSDIO DMSINI
TYP2401	000002	DMSASN DMSTPE
TYP2415	000001	DMSASN
TYP2420	000002	DMSASN DMSTPE
TYP2501	000001	DMSASN
TYP2540P	000001	DMSASN
TYP2540R	000001	DMSASN
TYP3203	000002	DMSASN DMSPRT
TYP3210	000001	DMSINI
TYP3211	000002	DMSASN DMSPRT
TYP3277	000001	DMSEDX
TYP3278	000001	DMSEDX
TYP3330	000005	DMSBOP DMSDIO DMSDOS DMSINI
TYP3340	000004	DMSASN DMSBOP DMSDOS DMSINI

LABEL	COUNT	REFERENCES											
TYP3350	000007	DMSBOP	DMSDIO	DMSDOS	DMSINI	DMSROS							
TYP3420	000003	DMSASN	DMSTPE										
TYP3525	000001	DMSASN											
UCASE	000003	DMSCRD											
UE	000001	DMSCIT											
UFDBUSY	000045	DMSABN	DMSACC	DMSACF	DMSACM	DMSAUD	DMSBTE	DMSBWR	DMSCIT	DMSDIO	DMSDOS	DMSDSK	DMSERS
		DMSFNS	DMSITE	DMSITI	DMSITP	DMSITS	DMSRNM	DMSTPE					
UND	000019	DMSROS	DMSSBS	DMSSEB	DMSSOP	DMSSQS							
UNPACK	000013	DMSCPY	DMSEXT	DMSLIO									
UNRES	000005	DMSLDR	DMSLOA	DMSOLD									
UPBIT	000006	DMSACM	DMSAUD	DMSDSK									
UPSI	000004	DMSSET											
UPTMID	000002	DMSSET											
UPTSWS	000002	DMSSET											
USARCODE	000002	DMSFRE											
USAVE	000003	DMSITS											
USAVEPTR	000025	DMSITS	DMSSAB	DMSSLN	DMSSOP	DMSSTG	DMSSVT						
USAVESZ	000005	DMSITS											
USERCODE	000004	DMSFRE	DMSSET										
USERKEY	000012	DMSABN	DMSDBG	DMSFRE	DMSITS	DMSLDR	DMSSET						
UTILFLAG	000020	DMSEDI	DMSSCR										
VAR	000033	DMSOR1	DMSROS	DMSSBD	DMSSBS	DMSSEB	DMSSOP	DMSSQS	DMSSVT	DMSTPD	DMSXCP		
VCADTLKP	000029	DMSACC	DMSACM	DMSALU	DMSARE	DMSASN	DMSBOP	DMSDIO	DMSDLB	DMSDSL	DMSEXT	DMSFOR	DMSLDS
		DMSLLU	DMSLST	DMSQRY	DMSRNM	DMSSET	DMSSVT	DMSUPD	DMSXCP				
VCADTLKW	000007	DMSAMS	DMSARN	DMSEXT	DMSRNE	DMSSRT	DMSUPD						
VCADTNXT	000009	DMSACC	DMSALU	DMSARE	DMSLDS	DMSLST	DMSQRY	DMSROS					
VCFSTLKP	000005	DMSACC	DMSDSK	DMSEDX	DMSTPE	DMSXCP							
VCFSTLKW	000004	DMSRNM	DMSTPE										
VERCOL1	000009	DMSEDI	DMSEDX	DMSSCR									
VERCOL2	000004	DMSEDI	DMSEDX										
VERLEN	000007	DMSEDI	DMSEDX	DMSSCR									
VIPINIT	000009	DMSCLS	DMSDOS	DMSEXT	DMSINT	DMSSTG	DMSVIB	DMSVSR					
VIPSOP	000008	DMSBOP	DMSCLS	DMSVIP									
VIPTCLOS	000004	DMSCLS	DMSVIP										
VIRTUAL	000021	DMSACC	DMSAMS	DMSARN	DMSBWR	DMSCMP	DMSDLB	DMSEDX	DMSFCH	DMSFNS	DMSLBM	DMSLIO	DMSNCP
		DMSQRY	DMSSET	DMSSMN	DMSTMA	DMSTPD	DMSVIB	DMSVIP	DMSVPD	DMSZAP			
VMCOMP	000002	DMSDSV											
VMDISP	000004	DMSDSV											
VMDISP1	000005	DMSDSV											
VMSIZE	000041	DMSAMS	DMSBOP	DMSBRD	DMSBWR	DMSDBG	DMSDOS	DMSFRE	DMSHDI	DMSHDS	DMSINS	DMSLDR	DMSOVS
		DMSSET	DMSSSK	DMSSVT	DMSVIB								
VSAMFLG1	000051	DMSABN	DMSAMS	DMSBAB	DMSBOP	DMSCLS	DMSDLB	DMSDOS	DMSEXT	DMSFCH	DMSINT	DMSITP	DMSSTG
		DMSVIB	DMSVIP	DMSVSR									
VSAMRUN	000010	DMSABN	DMSBOP	DMSDOS	DMSFCH	DMSSTG	DMSVIB	DMSVSR					
VSAMSERV	000015	DMSAMS	DMSBAB	DMSBOP	DMSCLS	DMSDLB	DMSDOS	DMSFCH	DMSITP	DMSSTG	DMSVSR		
VSAMSOS	000006	DMSABN	DMSAMS	DMSVSR									

LABEL	COUNT	REFERENCES											
VSJOBCAT	000003	DMSDLB											
VSMINSTL	000005	DMSFCH	DMSFET										
VSTRANGE	000001	DMSITI											
WAIT	000033	DMSABN	DMSCIO	DMSCIT	DMSCRD	DMSCWR	DMSCWT	DMSDOS	DMSFNC	DMSINI	DMSINS	DMSITE	DMSITI
		DMSPIO	DMSSVT										
WAITEND	000003	DMSSVN											
WAITING	000003	DMSVIP											
WAITLIST	000002	DMSDBG	DMSSVT										
WAITLST	000003	DMSCRD	DMSCWR	DMSCWT									
WAITRD	000004	DMSDBG	DMSFNC	DMSFOR									
WAITSAVE	000007	DMSCIT	DMSDBG	DMSIOW									
WORKFILE	000005	DMSCLS	DMSOLD										
WRBIT	000012	DMSACC	DMSBWR	DMSDSK	DMSTPE								
WRDATA	000022	DMSINI											
WRITE	000028	DMSBOP	DMSCLS	DMSDIO	DMSDLK	DMSDSL	DMSINI	DMSSBS	DMSTPE	DMSVPD			
WRITE1	000007	DMSINI											
WRTKF	000003	DMSDIO											
WTM	000011	DMSBOP	DMSCLS	DMSTPE									
WTRDCNT	000002	DMSDBG											
XAREA	000001	DMSEDI											
XCOUNT	000002	DMSOVS											
XGPR0	000002	DMSOVS											
XGPR1	000001	DMSOVS											
XGPR15	000002	DMSOVS											
XPSW	000013	DMSDBG	DMSITE										
XRSAVE	000003	DMSDIO											
XXXCWD	000042	DMSEDI											
XYCNT	000008	DMSEDI											
XYFLAG	000003	DMSEDI											
YAREA	000001	DMSEDI											
YDISK	000003	DMSINI	DMSINS	DMSNUC									
YYDDD	000003	DMSINS											
Y2	000001	DMSSCR											
ZDISK	000001	DMSNUC											
ZEROES	000014	DMSINI	DMSOR1	DMSROS									
ZONE1	000011	DMSEDI	DMSEDX										
ZONE2	000016	DMSEDI	DMSEDX										

CMS Diagnostic Aids

This section contains the following information:

- A list of devices Supported by a CMS Virtual Machine
- DMSFREX Error Codes
- Abend Codes

Supported Devices

Figure 23 indicates those devices that are supported by a CMS machine.

Virtual IBM Device	Virtual Address[1]	Symbolic Name	Device Type
3210, 3215, 1052, 3066, 3270	cuu	CON1	System console
2314, 3330, 3340 3350	190	DSK0	System disk (read-only)
2314, 3330, 3340 3350	191[2]	DSK1	Primary disk (user files)
2314, 2319, 3330, 3340, 3350	cuu	DSK2	Disk (user files)
2314, 2319, 3330, 3340, 3350	cuu	DSK3	Disk (user files)
2314, 2319, 3330, 3340, 3350	192	DSK4	Disk (user files)
2314, 2319, 3330, 3340, 3350	cuu	DSK5	Disk (user files)
2314, 2319, 3330, 3340, 3350	cuu	DSK6	Disk (user files)
2314, 2319, 3330, 3340, 3350	cuu	DSK7	Disk (user files)
2314, 2319, 3330, 3340, 3350	19E	DSK8	Disk (user files)
2314, 2319, 3330, 3340, 3350	cuu	DSK9	Disk (user files)
1403, 3203, 3211, 1443	00E	PRN1	Line printer
2540, 2501, 3505	00C	RDR1	Card reader
2540, 3525	00D	PCH1	Card punch
2415, 2420, 3410, 3420	181-4	TAP1-TAP4	Tape drives

[1]The device addresses shown are those that are preassembled into the CMS resident device table. These need only be modified and a new device table made resident to change the addresses.

[2]The virtual device address (cuu) of a disk for user files can be any valid System/370 device address, and can be specified by the CMS user when he activates a disk. If the user does not activate a disk immediately after loading CMS, CMS automatically activates the primary disk at virtual address 191.

Figure 23. Devices Supported by a CMS Virtual Machine

DMSFREX Error Codes

Error Codes from DMSFREE, DMSFRES, and DMSFRET

A nonzero return code upon return from DMSFRES, DMSFREE, or DMSFRET indicates that the request could not be satisfied. Register 15 contains this return code, indicating which error has occurred. The codes below apply to the DMSFRES, DMSFREE and DMSFRET macros, described on the following pages.

Code	Error
1	(DMSFREE) Insufficient storage space is available to satisfy the request for free storage. In the case of a variable request, the minimum request could not be satisfied.
2	(DMSFREE or DMSFRET) User storage pointers destroyed.
3	(DMSFREE or DMSFRET) Nucleus storage pointers destroyed.
4	(DMSFREE) An invalid size was requested. This error exit is taken if the requested size is not greater than zero. In the case of variable requests, this error exit is taken if the minimum request is greater than the maximum request. However, the error is not detected if DMSFREE is able to satisfy the maximum request.
5	(DMSFRET) An invalid size was passed to the DMSFRET macro. This error exit is taken if the specified length is not positive.
6	(DMSFRET) The block of storage that is being released was never allocated by DMSFREE. This error occurs if one of the following errors is found: a. The block is not entirely inside either the low-core free storage area or the user program area between FREELOWE and FREEUPPR. b. The block crosses a page boundary that separates a page allocated for USER storage from a page allocated for NUCLEUS storage. c. The block overlaps another block already on the free storage chain.
7	(DMSFRET) The address given for the block being released is not a doubleword boundary address.
8	(DMSFRES) An illegal request code was passed to the DMSFRES routine. Because the DMSFRES macro generates all codes, this error code should never appear.
9	(DMSFRE, DMSFRET, or DMSFRES) Unexpected internal error.

Abend Codes

Abend Recovery

Modules Used: DMSABN

Operation of the Abend Routine, DMSABN

When the abend recovery routine is entered, it types out the abend message, followed by the line "CMS", to indicate to the user that he may type in his next command.

At this point, there are two options available to the user.

First, he may type the DEBUG command. In this case, DMSABN passes control to DMSDBG, to make the facilities of DEBUG available to him. DEBUG's PSW and registers are as they were at the time that the abend recovery routine was invoked. From DEBUG, the user may alter the PSW or registers, as he wishes, and type GO to continue processing, or type RETURN to return to DMSABN, so that abend recovery can continue.

The second option available is to type in any other command. If this is done, DMSABN performs its abend recovery function and passes control to DMSINT to execute the command that has been typed in.

The abend recovery function consists of the following steps:

1. The SVC handler, DMSITS, is reinitialized, and all stacked save areas are released.
2. "FINIS * * *" is invoked by means of SVC 202, to close all files, and to update the user file directory.
3. If the EXEC interpreter (EXECTOR module) is in storage, it is released.
4. All link blocks allocated by the OS macros simulation routine DMSSLN are freed.
5. If VSAM or Access Method Services are still active, call DMSVSR for cleanup.
6. All FCB and DOSCB pointers are zeroed out.
7. All user storage is released.
8. The amount of system free storage that should be allocated is computed. This figure is compared against the amount of free storage that is actually allocated. If the two are equal, then storage recovery can be considered successful. If they are unequal, then a message is sent to the user.

UNRECOVERABLE TERMINATION -- THE HALT OPTION OF DMSERR

There are certain times, such as when the SVC handler's pointers are modified, that the system can neither continue processing nor try to recover. In these cases, DMSERR with the option HALT=YES is specified to cause a message to be typed out, after which a disabled wait state PSW is loaded unless the NUCON field AUSERRST has been loaded.

The valid address contained in AUSERRST is assumed to be the address of an error recovery routine and will be directly branched to. The initialization routines of an application running under CMS must set this address to point to a module that might, for example, request a dump and then issue an IPL command. If the IPL command is

```
IPL CMS PARM AUTOCR
```

and the PROFILE EXEC on virtual disk 191 invokes reinitialization, the application has the capability of automatic recovery. This capability is valuable for CMS service virtual machines that run permanently disconnected and are required to stay operational.

In CP mode, the programmer can examine the PSW, whose address field contains the address of the instruction following the call to the DMSERR macro. He can also examine all the registers, which are as they were when the DMSERR macro was invoked.

Figure 24 lists the CMS ABEND codes and describes the cause of the Abend and the action required.

Abend Code	Module Name	Cause of Abend	Action
001	DMSSCT	The problem program encountered an input/output error processing an OS macro. Either the associated DCB did not have a SYNAD routine specified or the I/O error was encountered processing an OS CLOSE macro.	Message DMSSCT120S indicates the possible cause of the error. Examine the error message and take the action indicated.
034	DMSVIP	The problem program encountered an I/O error while processing a VSAM action macro under DOS/VS for which there is no OS equivalent. An internal error occurred in a DOS VSAM routine.	Refer to the <u>DOS/VS Messages Reference</u>, Order No. GC33-5379, to determine the cause of the VSAM error.
0Cx	DMSITP	The specified hardware exception occurred at a specified location. "x" is the type of exception: <u>x</u> <u>Type</u> 0 IMPRECISE 1 OPERATION 2 PRIVILEGED OPERATION 3 EXECUTE 4 PROTECTION 5 ADDRESSING 6 SPECIFICATION 7 DECIMAL DATA 8 FIXED-POINT OVERFLOW 9 FIXED-POINT DIVIDE A DECIMAL OVERFLOW B DECIMAL DIVIDE C EXPONENT OVERFLOW D EXPONENT UNDERFLOW E SIGNIFICANCE F FLOATING-POINT DIVIDE	Type DEBUG to examine the PSW and registers at the time of the exception.
0F0	DMSITS	Insufficient free storage is available to allocate a save area for an SVC call.	If the abend was caused by an error in the application program, correct it; if not, use the CP DEFINE command to increase the size of virtual storage and then restart CMS.
0F1	DMSITS	An invalid halfword code is associated with SVC 203.	Enter DEBUG and type GO. Execution continues.

Figure 24. CMS Abend Codes (Part 1 of 4)

Abend Code	Module Name	Cause of Abend	Action
0F2	DMSITS	The CMS nesting level of 20 has been exceeded.	None. abend recovery takes place when the next command is entered.
0F3	DMSITS	CMS SVC (202 or 203) instruction was executed and provision was made for an error return from the routine processing the SVC.	Enter DEBUG and type GO. Control returns to the point to which a normal return would have been made.
0F4	DMSITS	The DMSKEY key stack overflowed.	Enter DEBUG and type GO. Execution continues and the DMSKEY macro is ignored.
0F5	DMSITS	The DMSKEY key stack underflowed.	
0F6	DMSITS	The DMSKEY key stack was not empty when control returned from a command or function.	Enter DEBUG and type GO. Control returns from the command or function as if the key stack had been empty.
0F7	DMSFRE	Occurs when TYPCALL=SVC (the default) is specified in the DMSFREE or DMSFRET macro.	When a system abend occurs, use DEBUG to attempt recovery.
0F8	DMSFRE	Occurs when TYPCALL=BALR is specified in the DMSFREE or DMSFRET Macro devices.	When a system abend occurs, use DEBUG to attempt recovery.
101	DMSSVN	The wait count specified in an OS WAIT macro was larger than the number of ECBs specified.	Examine the program for excessive wait count specification.
104	DMSVIB	The OS interface to DOS/VS VSAM is unable to continue execution of the problem program.	See the additional error message accompanying the abend message, correct the error, and reexecute the program.
155	DMSSLN	Error during LOADMOD after an OS LINK, LOAD, XCTL, or ATTACH. The compiler switch is on.	See the last LOADMOD (DMSMOD) error message for error description. In the case of an I/O error, recreate the module. If the module is missing, create it.

Figure 24. CMS Abend Codes (Part 2 of 4)

Abend Code	Module Name	Cause of Abend	Action
15A	DMSSLN	Severe error during load (phase not found) after an OS LINK, LOAD, XCTL, or ATTACH. The compiler switch is on.	See last LOAD error message (DMSLIO) for the error description. In the case of an I/O error, re-create the text deck or TXTLIB. If either is missing, create it.
174	DMSVIB	The OS interace to DOS/VS VSAM is unable to continue execution of the problem program.	See the additional error message accompanying the abend message, correct the error, and reexecute the program.
177	DMSVIB DMSVIP	The OS interface to DOS/VS VSAM is unable to continue execution of the problem program.	See the additional error message accompanying the abend message, correct the error, and reexecute the program.
240	DMSSVT	No work area was provided in the parameter list for an OS RDJFCB macro.	Check RDJFCB specification.
400	DMSSVT	An invalid or unsupported form of the OS XDAP macro was issued by the problem program.	Examine program for unsupported XDAP macro or for SVC 0.
704	DMSSMN	An OS GETMAIN macro (SVC 4) was issued specifying the LC or LU operand. These operands are not supported by CMS.	Change the program so that it specifies allocation of only one area at a time.
705	DMSSMN	An OS FREEMAIN macro (SVC 5) was issued specifying the L operand. This operand is not supported by CMS.	Change the program so that it specifies the release of only one area at a time.
804 80A	DMSSMN	An OS GETMAIN macro (804 - SVC 4, 80A - SVC 10) was issued that requested either zero bytes of storage, or more storage than was available.	Check the program for a valid GETMAIN request. If more storage was requested than was available, increase the size of the virtual machine and retry.
905 90A	DMSSMN	An OS FREEMAIN macro (905 - SVC 5, 90A - SVC 10) was issued specifying an area to be released whose address was not on a doubleword boundary.	Check the program for a valid FREEMAIN request; the address may have been incorrectly specified or modified.

Figure 24. CMS Abend Codes (Part 3 of 4)

Abend Code	Module Name	Cause of Abend	Action
A05 A0A	DMSSMN	An OS FREEMAIN macro (A05 - SVC 5, A0A - SVC 10) was issued specifying an area to be released which overlaps an existing free area.	Check the program for a valid FREEMAIN request; the address and/or length may have been incorrectly specified or modified.

Figure 24. CMS Abend Codes (Part 4 of 4)

Appendix A: CMS Macro Library

The following is a list and brief description of the CMS macros applicable to Release 5.

Asterisk (*) indicates that the macro is reserved for IBM use.

CMS Macro	Function
*ADT	Generates a CSECT or DSECT for an active disk table.
*ADTGEN	Generates an active disk table (ADT) for a disk; used by ADTSECT.
*ADTSECT	Generates all the ADTs for CMS.
*AFT	Generates a DSECT for an active file table.
*AFTSECT	Generates all the AFTs for CMS.
BATLIMIT	Table of CPU, punch, and printer limits for user jobs running under CMS batch.
*CMSAVE	Equivalent to SVCSAVE macro.
*CMSCB	Generates a list of simulated OS control blocks.
*CMSCVT	Generates the communication vector table as supported by CMS.
COMPSWT	Sets the compiler switch on or off. Refer to _VM/370 CMS Command and Macro Reference_.
*CORG	Sets the origin for CSECT.
*DBGSECT	Generates a CSECT or DSECT for DEBUG environment variables.
*DEVGEN	Generates a device table for a given device; used by the DEVTAB macro.
*DEVSECT	DSECT for a device table.
*DEVTAB	Generates the device tables for the CMS nucleus.
*DIAG	Issues a specified CP Diagnose instruction.
*DIOSECT	Generates a CSECT or DSECT for all I/O information.
DISPW	Generates the calling sequence for the display terminal interface. Refer to _VM/370 System Programmer's Guide_.
DMSABN	ABEND the virtual machine. Refer to _VM/370 System Programmer's Guide_.
*DMSCCB	DSECT describes field of DOS command control block (CCB). Refer to _VM/370 Data Areas and Control Block Logic_.
*DMSABW	Allocates a work area for DMSABN.
*DMSDM	Reserved for IBM use.
*DMSERR	Sets up parameter list to type out a CMS error message; Refer to the LINEDIT macro.
*DMSERT	DMSERR work area DSECT.
DMSEXS	Execute an instruction without nucleus protection. Refer to _VM/370 System Logic and Problem Determination Guide--Volume 2_.
DMSFREE	Gets free storage. Refer to _VM/370 System Programmer's Guide_.
*DMSFRES	Calls system free storage service routines.
DMSFRET	Releases free storage. Refer to _VM/370 System Programmer's Guide_.
*DMSFREX	Calls system free storage service routines.
*DMSFRT	Generates a DSECT for free storage management work area.
*DMSFRX	Submacro called by DMSFRET.
DMSFST	Sets up a file status table for a given file. Refer to _VM/370 System Programmer's Guide_.

CMS Macro	Function
DMSKEY	Sets nucleus protection on or off. Refer to _VM/370 System Logic and Problem Determination Guide--Volume 2_.
*DMSLN	Called by DMSERR, LINEDIT macros.
*DMSLNC	Called by DMSERR, LINEDIT macros.
*DMSLND	Called by DMSERR, LINEDIT macros.
*DMSLNP	Called by DMSERR, LINEDIT macros.
*DMSLNU	Called by DMSERR, LINEDIT macros.
*DMSLNY	Called by DMSERR, LINEDIT macros.
*DMSLNZ	Called by DMSERR, LINEDIT macros.
*DMSPID	Passes a fileid in quotes into separate filename, filetype, filemode, used by FSCB, and FSPOINT.
*DMSTMS	Used by RDTAPE, WRTAPE, and TAPECTL.
*EDCB	Frees storage control blocks initialized by DMSEDX for CMS edit modules.
*EQUATES	Generates CMS equates for symbolic names.
*EXCP	Issues an SVC 0.
*EXTSECT	Defines storage for the timer interrupt.
*FCB	Generates a file control block (FCB) DSECT.
FSCB	Sets up a file system control block. Refer to _VM/370 CMS Command and Macro Reference_.
*FSCBD	DSECT that describes fields in CMS PLIST for related commands.
FSCLOSE	Closes a file. Refer to _VM/370 CMS Command and Macro Reference_.
*FSENTR	Used by CMS file system routines at entry.
FSERASE	Erases a file. Refer to _VM/370 CMS Command and Macro Reference_.
FSOPEN	Opens a file. Refer to _VM/370 CMS Command and Macro Reference_.
*FSPOINT	Executes the CMS POINT function.
FSREAD	Reads a record from a file. Refer to _VM/370 CMS Command and Macro Reference_.
FSSTATE	Checks for an existing file. Refer to _VM/370 CMS Command and Macro Reference_.
*FSTB	Generates a file status table (file directory) block.
*FSTD	Entry to the file status table (file directory) block.
FSWRITE	Writes a record into a disk file. Refer to _VM/370 CMS Command and Macro Reference_.
*FVS	Defines storage for file system variables.
*GETADT	Gets a specified active disk table.
*GETFST	Gets a specified file status table.
HNDEXT	Handles external and timer interrupts. Refer to _VM/370 CMS Command and Macro Reference_.
HNDINT	Handles interrupt on devices. Refer to _VM/370 CMS Command and Macro Reference_.
HNDSVC	Handles SVCs. Refer to _VM/370 CMS Command and Macro Reference_.
*IO	Contains PLISTs needed to access CMS I/O routines.
*IOSECT	Defines miscellaneous I/O variables.
*KEYSECT	Contains variables necessary for storage key handling.
*KXCHK	Checks to see if HX has been entered by the user.
*LDM	Loads double multiple (for floating point registers).
*LDRST	CMS Loader work area.
LINEDIT	Types a line to the terminal. Refer to _VM/370 CMS Command and Macro Reference_.
*NUCON	Generates a DSECT CMS nucleus constant area.

<u>CMS Macro</u>	<u>Function</u>
*OVSECT	DMSOVS work area.
*OSFST	Defines an OS file status table for CS ACCESS.
*PDSSECT	DSECT used for processing MACLIB files.
*PGMSECT	Defines work area for DMSITP.
PRINTL	Prints a line on the printer. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
PUNCHC	Punches a card. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
RDCARD	Reads a card from the reader. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
RDTAPE	Reads a record from tape. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
RDTERM	Reads a record from the terminal. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
REGEQU	Generates symbolic register equates. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
*RELPAGES	Sets the release pages flag.
*STDM	Storage for multiple floating-point registers.
STRINIT	Initializes storage. Refer to <u>VM/37C CMS Command and Macro Reference</u>.
*SUBSECT	CSECT or DSECT for CMS SUBSET use.
*SVCENT	Issues a DMSKEY macro before calling an instruction.
*SVCSAVE	System save area.
*SVCSECT	Defines work area for DMSITS.
*SYSLOAD	Puts in a specified register the address of a specified routine in NUCON.
*SYSNAMES	Saves system names table loaded via CMS routines.
TAPECTL	Positions a tape. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
*TSOBLKS	Contains CPPL, UPT, PSCB, and the ECT for TSO service routines.
*TSOGET	Gets the address of the TSO command processor parameter list (CPPL).
*USE	Generates assembler USING and DROP instructions, as needed.
*USERSECT	Creates user work area.
WAITD	Waits until the next interrupt occurs for the specified device. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
WAITT	Waits until all pending I/O to the terminal has completed. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
WRTAPE	Writes a record to tape. Refer to <u>VM/370 CMS Command and Macro Reference</u>.
WRTERM	Writes a record to the terminal. Refer to <u>VM/370 CMS Command and Macro Reference</u>.

Appendix B: CMS/DOS Macro Library

CMS, in this release, contains a DOS macro library with the following significant entries. A more complete list may be obtained by invoking the DOSMACRO EXEC; this EXEC produces a list of all the macros in the DOS library.

Macro	Function
CCB	Generates the DOS/VS command control block.
COMRG	Returns address of background partitions communication region; expands to SVC 33.
EOJ	Normal processing termination; expands to SVC 0.
OPENR	Activates a data file; simulated by DMSOR1, DMSOR2, DMSOR3.
STXIT	Provides/terminates supervisor linkage to user's program check routines; simulated by DMSDOS.
IKQACB	DSECT for VSAM ACB (access method control block).
IKQEXLST	DSECT for VSAM EXLST control block (contains addresses of user exit routines.
IKQRPL	DSECT for VSAM RPL (request parameter list control block).
SYSCOM	DSECT of system communication region.
ABTAB	DSECT of abnormal termination option table.
BBOX	DSECT of Boundary Box; contains beginning and ending addresses of background partitions communication region.
BGCOM	DSECT of background communication region.
FICL	DSECT, CMS/DOS first in class table.
NICL	DSECT, CMS/DOS number in class table.
PCTAB	DSECT, program check option table.
PIB2TAB	DSECT, program information block extension.
PIBTAB	DSECT, program information block.
PUBOWNER	DSECT, physical unit block ownership table.
ANCHTAB	DSECT, DOS/VS anchor table.
DOSAVE	DSECT, describes fields in the logical transient area (LTA).
FCHTAB	DOS/VS fetch table containing fetch/load parameter list.
MAPPUB	DSECT defines fields of CMS/DOS physical unit block (PUB).
PUBTAB	DSECT same usage as MAPPUB.
DOSCB	DOS simulation control block used for the simulation of the CMS file control block (FCB).
EXCPW	DSECT, work area for DMSXCP routine.
DOSCON	Creates CMS/DOS control blocks for DMSNUC.
LUBTAB	DSECT for CMS/DOS logical unit block.

Index

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

Trim Along This Line

READER'S COMMENT FORM

Title: **IBM Virtual Machine Facility/370: System Logic and Problem Determination Guide Volume 2**

Order No. **SY20-0887-1**

Please check or fill in the items; adding explanations/comments in the space provided.

Which of the following terms best describes your job?

☐ Customer Engineer	☐ Manager	☐ Programmer	☐ Systems Analyst
☐ Engineer	☐ Mathematician	☐ Sales Representative	☐ Systems Engineer
☐ Instructor	☐ Operator	☐ Student/Trainee	☐ Other (explain below)

How did you use this publication?

☐ Introductory text ☐ Reference manual ☐ Student/ ☐ Instructor text

☐ Other (explain) ____________________

Did you find the material easy to read and understand? ☐ Yes ☐ No (explain below)

Did you find the material organized for convenient use? ☐ Yes ☐ No (explain below)

Specific criticisms (explain below)

Clarifications on pages ____________________

Additions on pages ____________________

Deletions on pages ____________________

Errors on pages ____________________

Explanations and other comments:

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

SY20-0887-1

Reader's Comment Form

Cut or Fold Along Line

Fold and tape　　Please Do Not Staple　　Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Attn: VM/370 Publications

Fold and tape　　Please Do Not Staple　　Fold and tape

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

IBM Virtual Machine Facility/370: System Logic and Problem Determination Guide Vol 2　Printed in U.S.A.　SY20-0887-1

SY20-0887-1

IBM Virtual Machine Facility/370: System Logic and Problem Determination Guide Vol 2   Printed in U.S.A.   SY20-0887-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601