Licensed Material -- Property of IBM
File No. S370-36
Order No. LY20-0893-0

Program Product

IBM Virtual Machine/System Product: System Logic and Problem Determination Guide Volume 2 - CMS

Program Number 5664-167

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

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

PREREQUISITE PUBLICATIONS

IBM Virtual Machine/System Product:

Introduction, Order No. GC19-6200

Operator's Guide, Order No. SC19-6202

Terminal User's Guide, Order No. GC19-6206

CMS Command and Macro Reference,
Order No. SC19-6209

System Programmer's Guide, Order No. SC19-6203

Notice: The term VM/SP, as used in this publication, refers to VM/SP when used in conjunction with VM/370 Release 6.

First Edition (September 1980)

This first edition (LY20-0893 dated September 30, 1980) applies to the IBM Virtual Machine/System Product and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information contained herein; before using this publication in connection with the operation of IBM systems, consult the *IBM System/370 and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not annnounced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

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 Programming Publications, Dept. G60, P.O. Box 6, Endicott, New York, U.S.A. 13760. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. 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/System Product (VM/SP) when used in conjunction with VM/370 Release 6.

HOW THIS MANUAL IS ORGANIZED

This manual comprises two volumes:

- Volume 1. VM/SP Control Program (CP)
- Volume 2. Conversational Monitor System (CMS)

Each volume contains logic descriptions for the designated components of VM/SP. 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/SP. They indicate the program operation and organization in a general way to serve as a guide in understanding VM/SP. They are not meant to be a detailed analysis of VM/SP programming and cannot be used as such.

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

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

Appendix A, located in Volume 2, contains a description of the CMS macro library.

Appendix B, also located in Volume 2, describes the CMS/DOS macro library.

Appendix C, also located in Volume 2, describes CMS/DOS support modules.

Information on the Remote Spooling Communications Subsystem (RSCS), a VM/370 Release 6 component, is contained in:

> *VM/370 System Logic and Program Determination Guide, Volume 3 Remote Spooling Communications Subsystem (RSCS)*, Order No. SY20-0888

The control blocks supportive of the RSCS Logic are contained in:

> *VM/SP Data Areas and Control Blocks*, Order No. LY20-0891

Logic Information on the Interactive Problem Control System (IPCS), a VM/370 Release 6 component is totally contained in:

> *VM/SP Service Routines Program Logic*, Order No. LY20-0890

HOW TO USE THIS MANUAL

- Isolate the component of VM/370 in which the problem occurred.
- Use the list of restrictions in *VM/SP System Messages and Codes* to be certain that the operation that was being performed was valid.
- Use the directories and use the *VM/SP 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.
- "FB-512" refers to those IBM DASD devices implementing the fixed-block (512-byte blocks) architecture. Specifically, they are the IBM 3310, and the IBM 3370. Current IBM disk storage devices are referred to as count-key-data DASD when it is important to distinguish between count-key-data DASD and FB-512. Otherwise, they are collectively referred to as DASD or disk.
- "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.
- "3380" refers to the IBM 3380 Storage Facility. Information on the IBM 3380 Storage Facility is for planning purposes only until the availability of the product.
- "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, and 3279 Display Stations. A specific device type is used only when a distinction is required between device types.
- The term, System/370 processors, is also applicable to 4300 processors and 303x series processors unless indicated otherwise.
- 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.
- "3262" refers to the IBM 3262 Printer, Models 1 and 11. Information on the IBM 3262 Printer, Models 1 and 11, is for Planning purposes only, until the availability of the product.
- Unless otherwise noted, the term "VSE" refers to the combination of the DOS/VSE system control program and the VSE/Advanced Functions program product.

In certain cases, the term DOS is still used as a generic term. For example, disk packs initialized for use with VSE or any predecessor DOS or DOS/VS system may be referred to as DOS disks.

The DOS like simulation environment provided under the CMS component of the VM/System Product, continues to be referred to as CMS/DOS.

CMS COMPONENT

PREREQUISITE PUBLICATIONS

IBM Virtual Machine/System Product

Introduction, Order No. GC19-6200

Terminal User's Guide, Order No. GC19-6206

CMS Command and Macro Reference, Order No. SC19-6209

CMS User's Guide, Order No. SC19-6210

COREQUISITE PUBLICATIONS

IBM Virtual Machine/System Product

Operator's Guide, Order No. SC19-6202

CP Command Reference for General Users, Order No. SC19-6211

System Programmer's Guide, Order No. SC19-6203

System Messages and Codes, Order No. SC19-6204

OLTSEP and Error Recording Guide, Order No. SC19-6205

Operating Systems in a Virtual Machine, Order No. GC19-6212

Service Routines Program Logic, Order No. LY20-0890

Data Areas and Control Block Logic, Order No. LY20-0891

In addition, for EREP processing the following OS/VS Library publications are required: *OS/VS, DOS/VSE, VM/370 Environmental Recording Editing and Printing (EREP) Program*, Order No. GC28-0772

OS/VS, DOS/VSE, VM/370 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-, 1024-, 2048-, and 4096-byte block format; CMS's VSAM data sets are not included.

Contents

FIGURES

Conversational Monitor System (CMS)

This section contains the following information:

- Introduction to CMS
- Interrupt Handling in CMS
- Functional Information
- OS Macros Under CMS
- VSE Support Under CMS

Introduction to CMS

The Conversational Monitor System (CMS), the major subsystem of VM/SP, 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/SP 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/SP 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/SP 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 25 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.

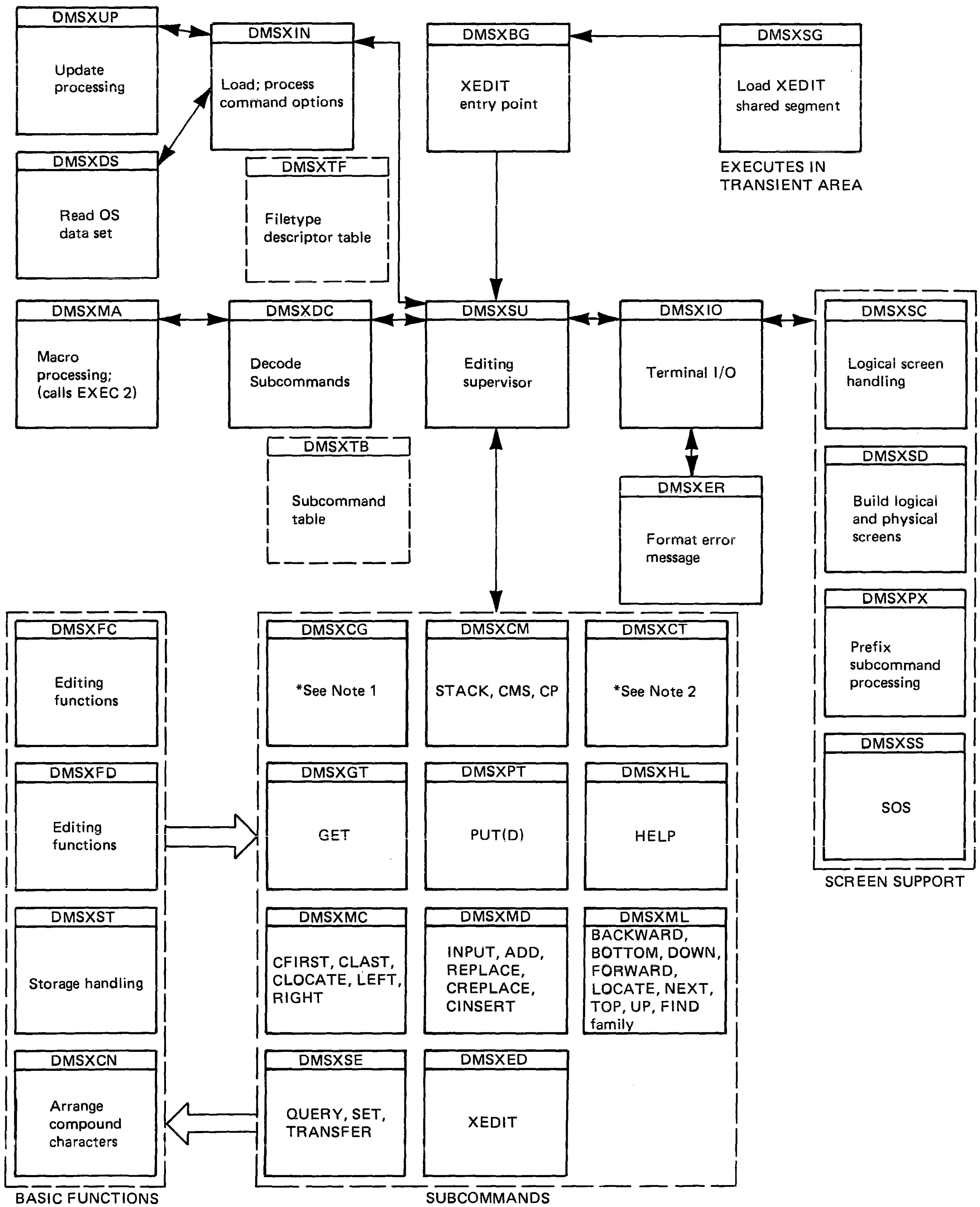
The new VM/SP System Product Editor provides full screen support for 3270 display stations. The new CMS editor coexists with the current editor. The major highlights of the new editor include:

- Multiple views of the same or different files
- Selective column viewing
- Automatic wrapping of lines larger than the screen
- Ability to issue selected commands directly from the displayed line
- Ability to define screen format
- Extended string search functions
- Column pointing for intra line editing

Additionally, the new editor provides language expansions and flexibility through the EXEC 2 processor. Figure 1 describes the modules that perform the processing for the new editor.

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.

CMS disk files contain records stored on disks as 800-, 1024-, 2048-, or 4096-byte records. For disks with 800-byte records a single user file is limited to a maximum of 65533 records and must reside on one virtual disk. The maximum number of files is limited by the file management system to 3400. For disks with 1024-, 2048-, and 4096-byte



*Note 1. CDELETE, CHANGE, COMPRESS, COPY, COUNT, COVERLAY, DELETE, DUPLICAT, EXPAND, LOWERCAS, MOVE, OVERLAY, UPPERCAS, RECOVER, SHIFT.

*Note 2. CMSG, CURSOR, EMSG, FILE, MSG, PRESERVE, PURGE, READ, RENUM, REPEAT, RESET, RESTORE, SAVE, TYPE.

Figure 1. Module Flow for the VM/SP System Product Editor

records a single user file is limited to a maximum of $2^{31}-1$ CMS blocks and must reside on one virtual disk. The maximum number of files on any one disk is limited by the file management system to $2^{31}-1$. However, the actual number of files is limited by the available disk space and the size of the user files.

All CMS disk files are written as 800-, 1024-, 2048-, or 4096-byte records chained together by a specific master file entry that is stored in a table called the file directory; a separate 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 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 CMS 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/SP 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.

Caution: 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 2 describes the file system for an 800-byte record on disk. Figure 22 shows the file system for 1K-, 2K-, and 4K-byte records on disk.

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 VSE 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 VSE 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 VSE simulation. The sequential, direct, and partitioned access methods are logically simulated; the data records are physically kept in the chained fixed-length blocks, 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 Access Method Services portion of VSE/VSAM. 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 VSE are ignored by CMS. For more information, see "OS Macro Simulation under CMS" and "VSE Support under CMS."

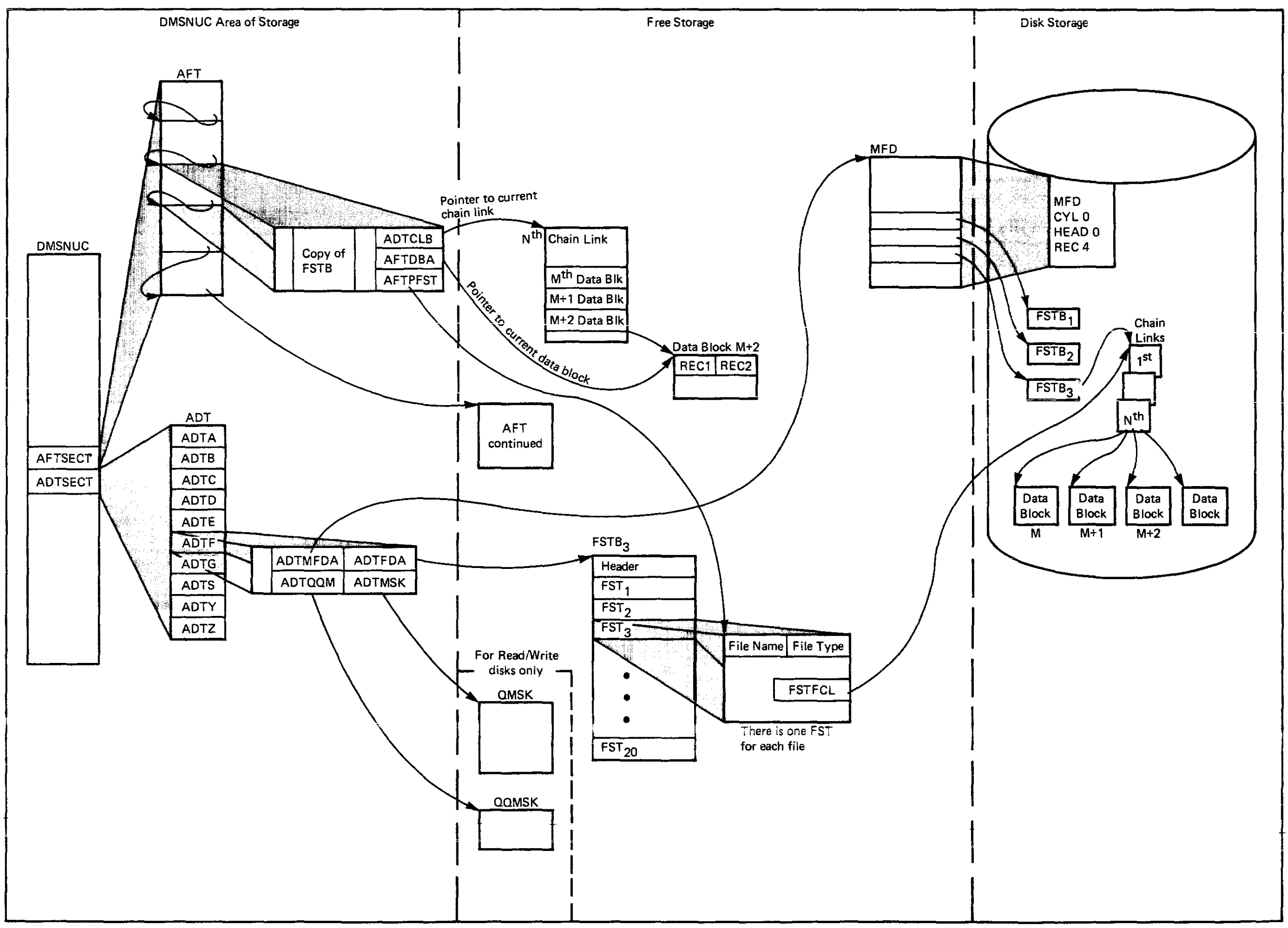


Figure 2. File System for an 800-Byte Record on Disk

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
- 26 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.

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'04000'). This area contains pointers, flags, and other data updated by the various system routines.

- CMS Nucleus First Part (X'04000' to approximately X'09000'). This area contains the following CMS Nucleus routines: DMSALU, DMSCIO, DMSVIB, DMSVSR, DMSDBD, DMSDBG, DMSFET, DMSTIO, DMSTLA, DMSTQQ, DMSITP, DMSABN, DMSITE, DMSPNT, DMSPIO, DMSLIO and DMSCPF.

- Low-Storage DMSFREE Free Storage Area (Approximately X'09000' 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'). 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 breakpcints 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.

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. In addition, a DIAGNOSE code X'10' instruction is sent to release these pages between MAINHIGH and FREELOWE. 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 returned, the MAINHIGH pointer is adjusted downward, and the freed pages are released by issuing a DIAGNOSE code X'10' instruction to CP.

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 2 for a description of CMS virtual storage usage.

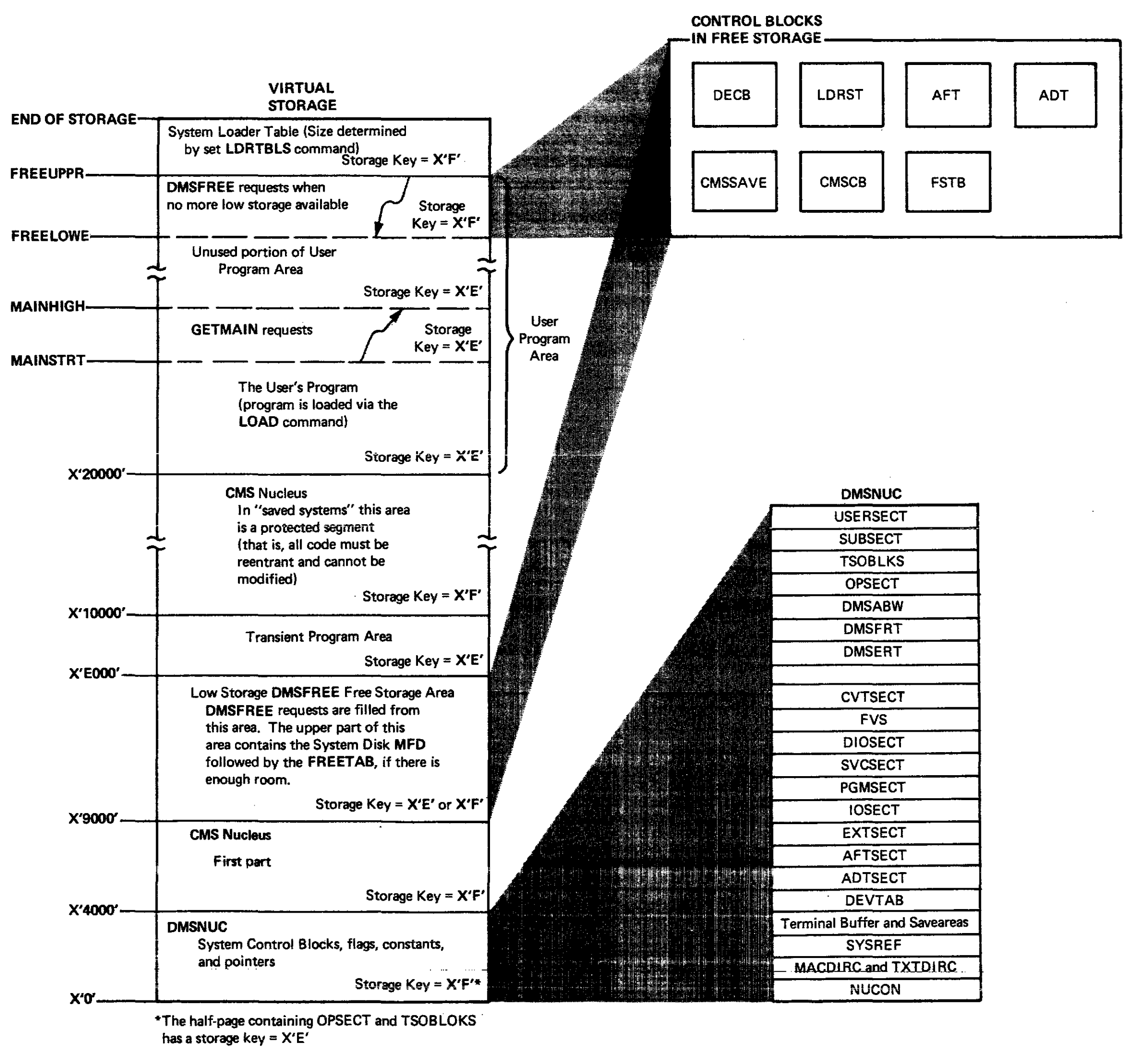


Figure 3. CMS Storage Map

The format cf 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, additional 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.

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││             │
│          │          │ └      └     ┘┘  └         └    ┘┘             │
└──────────────────────────────────────────────────────────────────────┘
```

<u>where</u>:

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= [<u>USER</u> | 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 cf 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 returned, this pointer is adjusted upward, and the freed pages are released by issuing a DIAGNOSE CODE X'10' instruction to CP.

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:

Code	Meaning
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=[SVC | 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. All the corresponding full pages in the freed blocks are released by issuing a DIAGNOSE code X'10' instruction to CP.

2. The STRINIT macro instruction releases all storage allocated by any previous GETMAIN requests. All corresponding full pages between MAINHIGH and FREELOWE are released by issuing a DIAGNOSE code X'10' instruction to CP.

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 | BALR} (SVC underscored as default)	indicates how control is passed to DMSFES. Since DMSFRES 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.

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, 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)
```

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.

Whenever SVC 202 is called, a tokenized or untokenized parameter list (PLIST) can be specified. In both cases, register 1 points to an eight-character string defining the symbolic name of the routine or command being called. The SVC handler will examine only the name and the high-order byte of register 1.

Tokenized PLIST: For a tokenized parameter list, the symbolic name of the function being called (8 character string, padded with blank characters on the right if needed) will be followed by extra arguments depending on the actual routine or command being called. These arguments must be "tokenized" (that is, have a maximum length of eight characters, padded on the right with blank characters if shorter than eight characters). Extra information on the origin of the call is provided by the high-order byte of register 1. If the contents of this byte is equal to:

X'0E' - the call is the result of a command invoked from an EXEC file with the "&CONTROL NOMSG".

X'0D' - the call is the result of a command invoked from an EXEC with "&CONTROL MSG" (that is, messages are to be displayed).

X'0C' - the command is called as a result of it's name being typed at the terminal. This flag byte may be used, for example, to recognize the need for human readable messages instead of return codes.

X'00' - the call did not originate from an EXEC file or a command typed at the terminal.

Untokenized PLIST: For an untokenized parameter list, no restriction is put on the structure of the arguments list passed to the called routine or command. The high-order byte of register 1 contains X'01' or X'02'. X'01' means a normal hierarchy search is done in the manner described under the "SEARCH HIERARCHY FOR SVC 202" section of this manual. If it contains X'02', the search for the called routine is limited to the SUBCOM list (see the section entitled "Dynamic

Linkage/SUBCOM" in this manual). Register 0 points to the untokenized PLIST which is constituted of four consecutive words:

```
¹DC A("Reserved Word")
²DC A(CMDBEG)
³DC A(CMDEND)
⁴DC A(0)
```

where the last two addresses are defined by:

```
CMDBEG EQU *
       DC C'QUERY INPUT'
CMDEND EQU *
```

CMDBEG EQU * indicates the beginning of the argument list and CMDEND EQU * indicates the end of the argument list.

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.

----

[1]The first word is reserved.
[2]The second gives the beginning address of the argument list.
[3]The third gives the address of the byte immediately following the end of the argument list.
[4]The fourth word is optional. Any words following this word are available for passing information between the calling program and the program being called.

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 VSE Macro Simulation SVC Calls

CMS supports selected SVC calls generated by OS and VSE 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 VSE 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

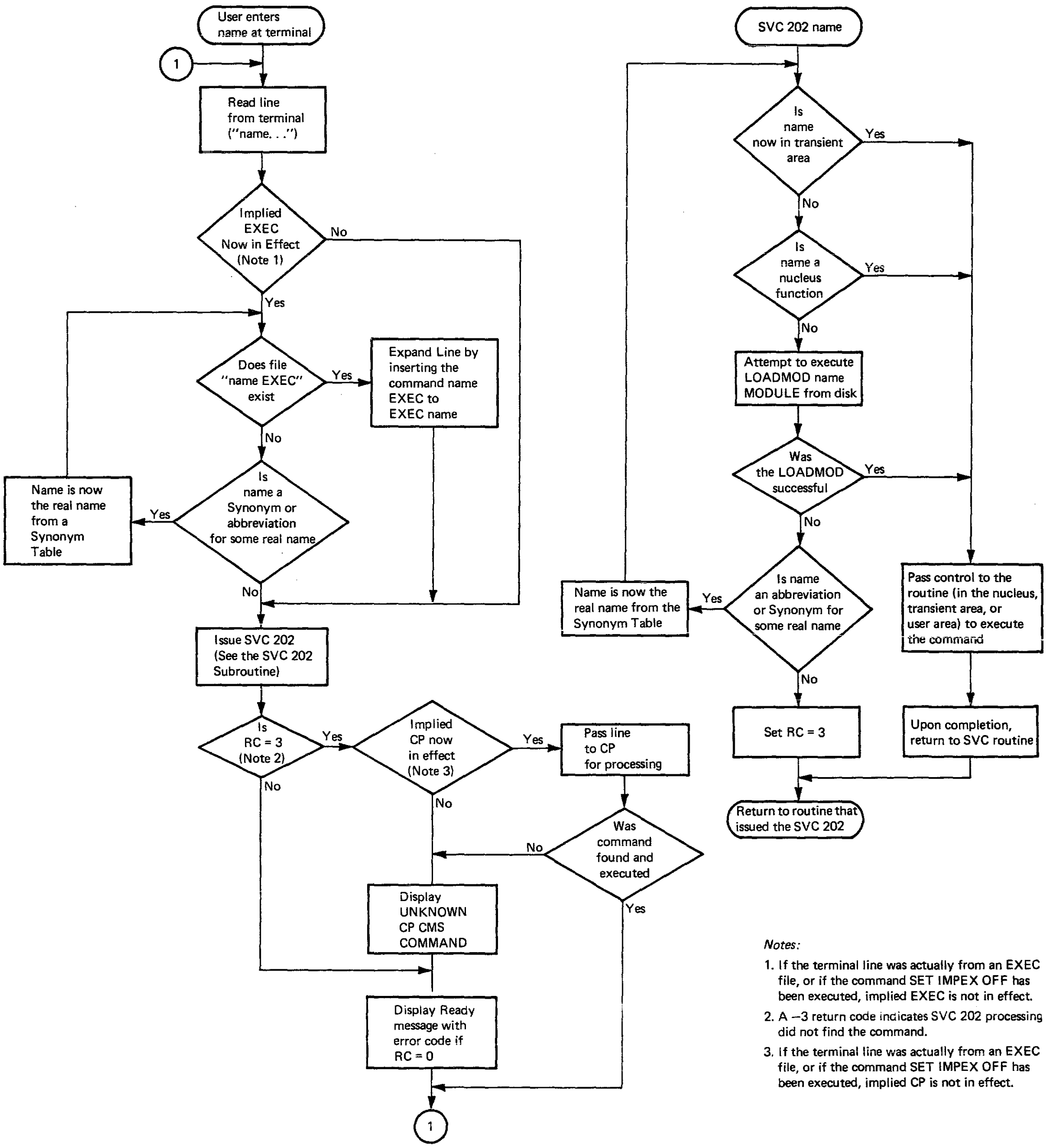


Figure 4. CMS Command (and Request) Processing

RSCS modules and their attributes, including whether they reside in the user program area or the transient area is contained in the *VM/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.

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 - VSE Nucleus resident	Disabled	System	Off
OS - VSE 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	Unpre-dictable	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/SP 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.

It is possible for programs that are already loaded to become dynamically known by name and callable via SVC 202. These programs can also make other programs dynamically known if the entry points of these other programs are known. To do this, a program or routine must invoke the create function of SUBCOM. This is done by issuing the following calling sequence from an assembler program (Register 1 must point to this calling sequence):

```
DS 0F
DC CL8'SUBCOM'
DC Cl8'[program or routine name]'
DC 4X'00' reserved
DC A ("entry point")
DC 4X 'available for user information'
```

This sequence makes the program or routine known to CMS.

SUBCOM creates an SCBLOCK control block containing the information you specified. SVC 202 uses this control block to make the communication. See the publication _VM/SP Data Areas and Control Block Logic_ for a description of the SCBLOCK control block.

Note: When a transfer to the specified entry point takes place, register 2 points to the SCBLOCK.

Future SVC 202 calls to the program or routine with the high-order byte of register 1 equal to X'02' will branch to the previously loaded copy of the program or routine at an address specified by the program or routine when it called SUBCOM.

You can also use SUBCOM to delete this potential linkage to the program or routine's SCBLOCK or to query if an SCBLOCK exists for a program or routine. To delete a program or routine's SCBLOCK, you issue:

```
DC CL8'SUBCOM'
DC CL8'[program or routine name]'
DC 8X'00'
```

To query if a SCBLOCK exists for a program or routine, you issue:

```
DC CL8'SUBCOM'
DC CL'8'[program or routine name]
DC A(0) SCBLOCK address as a returned value
DC 4X'FF'
```

Return Codes from SUBCOM are:

- 0 - Successful return code. Means new SCBLOCK was created, or specified SCBLOCK was deleted, or specified program or routine has an SCBLOCK.
- 1 - No SCBLOCK exists for the specified program or routine. This is the return code for a delete or a query.
- 25 - No more free storage available. SCBLOCK cannot be created for specified program or routine.

Note: If you set up SCBLOCKs for several programs or routines with the same name, SUBCOM will use the last submitted.

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 not 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. DMSGIO 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]
```

where:

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/SP 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, OS 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/VSE Access Method Services and VSE/VSAM and, therefore, the OS user is limited to those VSAM functions that are available under DOS/VSE.

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 tc 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
PGRLSE[1]	112	Release storage contents

[1]Simulated in the 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 LOADLIB directory or 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/SP 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.

Macro-SVC No.	Differences in Implementation
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.
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 _OS/VS Data Management Macro Instructions_, 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.

Macro-SVC No.	Differences in Implementation
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.
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. For TCLOSE, the REREAD option is REWIND, followed by a forward space file for tapes with standard labels.
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.

Macro-SVC No.	Differences in Implementation
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.
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.

Macro-SVC No.	Differences in Implementation
STAX-SVC96	Updates a queue of CMTAXEs each of which defines an attention exit level.
PGRLSE-SVC112	Release all complete pages (4K bytes) associated with the area of storage specified.
NOTE	All the options of NOTE are supported. NOTE returns the item number of the last block read or written.
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	-	J	J
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 VSE/VSAM. See the section "CMS Support for OS and DOS 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 not 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 block, 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 cption (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 ECE in place of the sense bytes. These error codes differ foı various types of devices and their meaning can be found in the _IBM VM/SP System Messages and Codes_, under DMS message 120S.

Note: If OPTCD J is specified in the FILEDEF command, the proper flag is set in the JFCOPTCD byte of the FCBSECT (simulated OS control block). During simulation of the OS OPEN macro, the FILEDEF value will be merged into DCBOPTCD. After DCBOPTCD is set, the first data byte of output lines presented to the PUT (QSAM) and WRITE (BSAM) macros is interpreted as a table reference character (TRC) byte. CP uses the TRC byte to select translate tables when printing on a 3800. The translate table determines the font type at real print time. If the virtual printer is not a 3800, the TRC byte is stripped off and the line is printed in the usual manner.

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 the CMS simulation of BDAM files uses a three-byte record identifier on 1K, 2K, and 4K format CMS minidisks. For 800-byte disks, only the last two identifiers are used.

- 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} { │DISK fn ft       │fm││ │DSN ?           │  }         │
│         │ {  nn    } │                 │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/SP CMS Command and_

Macro Reference for a complete description of the CMS ACCESS, FILEDEF, LISTDS, LKED, MOVEFILE, OSRUN, QUERY, RELEASE, and STATE commands.

For restrictions on reading OS data sets and DOS files under CMS, see the *VM/SP 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/SP 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.

VSE Support Under CMS

CMS supports interactive program development for VSE. This includes creating, compiling, testing, debugging, and executing commercial application programs. The VSE programs can be executed in a CMS virtual machine or in a CMS Batch Facility virtual machine.

VSE files and libraries can be read under CMS. VSAM data sets can be read and written under CMS.

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

- VSE linkage editor
  Fetch support
- VSE Supervisor and I/O macros
- VSE Supervisor control block support
- Transient area support
- VSE/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 VSE facilities, but does not support OS simulation. When you no longer need VSE support under CMS, you issue the SET DOS OFF command and VSE facilities are no longer available.

CMS/DOS can execute programs that use the sequential access method (SAM) and VSE/VSAM, and can access VSE 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, DOS/VS RPG II 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 VSE application program development. Because CMS/DOS is a VSE program development tool, it assumes that a VSE 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 VSE programs using VSE/VSAM CMS supports VSAM for OS programs written in VS BASIC, OS/VS COBOL, or OS PL/I programming languages; or VSE programs written in DOS/VS COBOL, DOS PL/I, DOS/VS RPG II programming languages. CMS does not support VSAM for OS or VSE 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. This compatibility with OS is limited to VSAM data sets created with physical record sizes of .5k, 1k, 2k, and 4k bytes. For further information on compatibility between OS/VS VSAM and VSE/VSAM, please refer to the VSE/VSAM General Information Manual.

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 or INITDISK (for FB-512 disks); 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 VSE/VSAM, only disks supported by DOS/VSE can be used for VSAM data sets in CMS. These disks are:

- IBM 2314 Direct Access Storage Facility
- IBM 2319 Disk Storage
- IBM 3310 Direct Access 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
- IBM 3370 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.

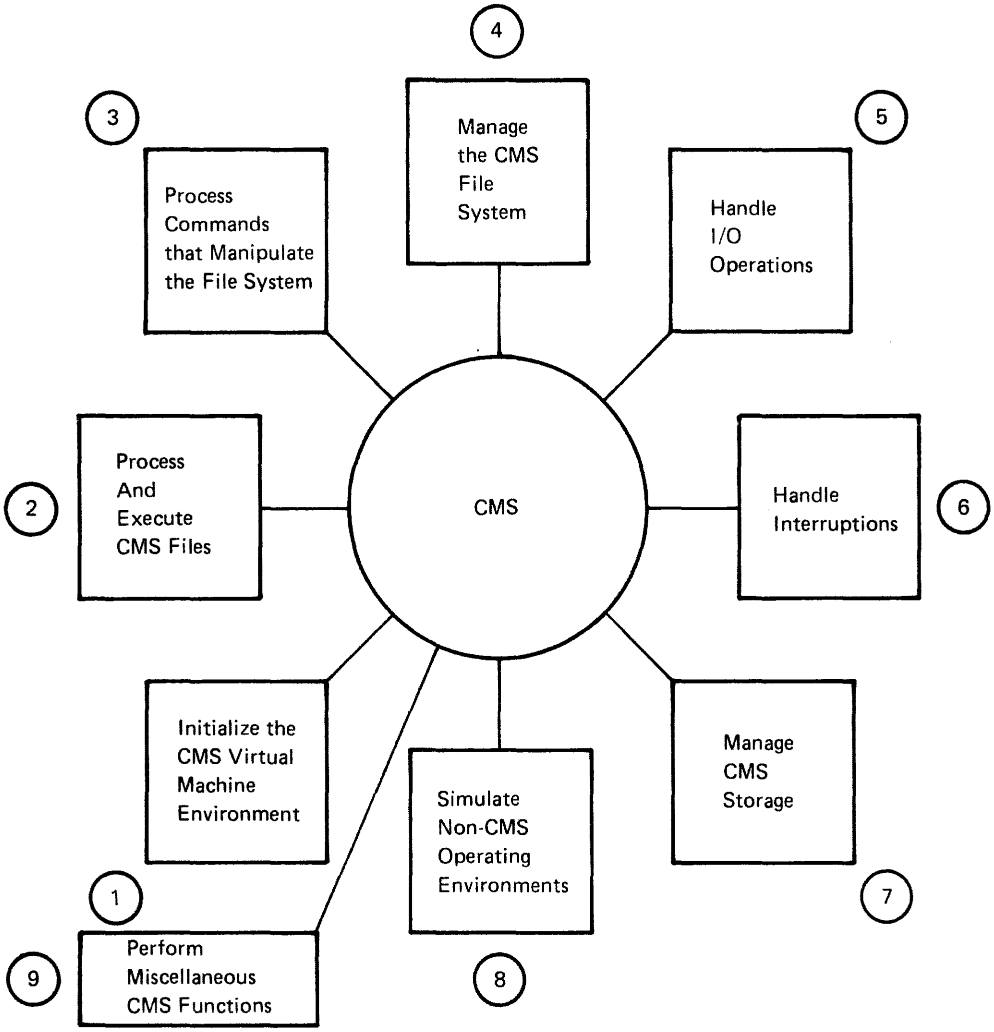


Figure 8. An Overview of the Functional Areas of CMS

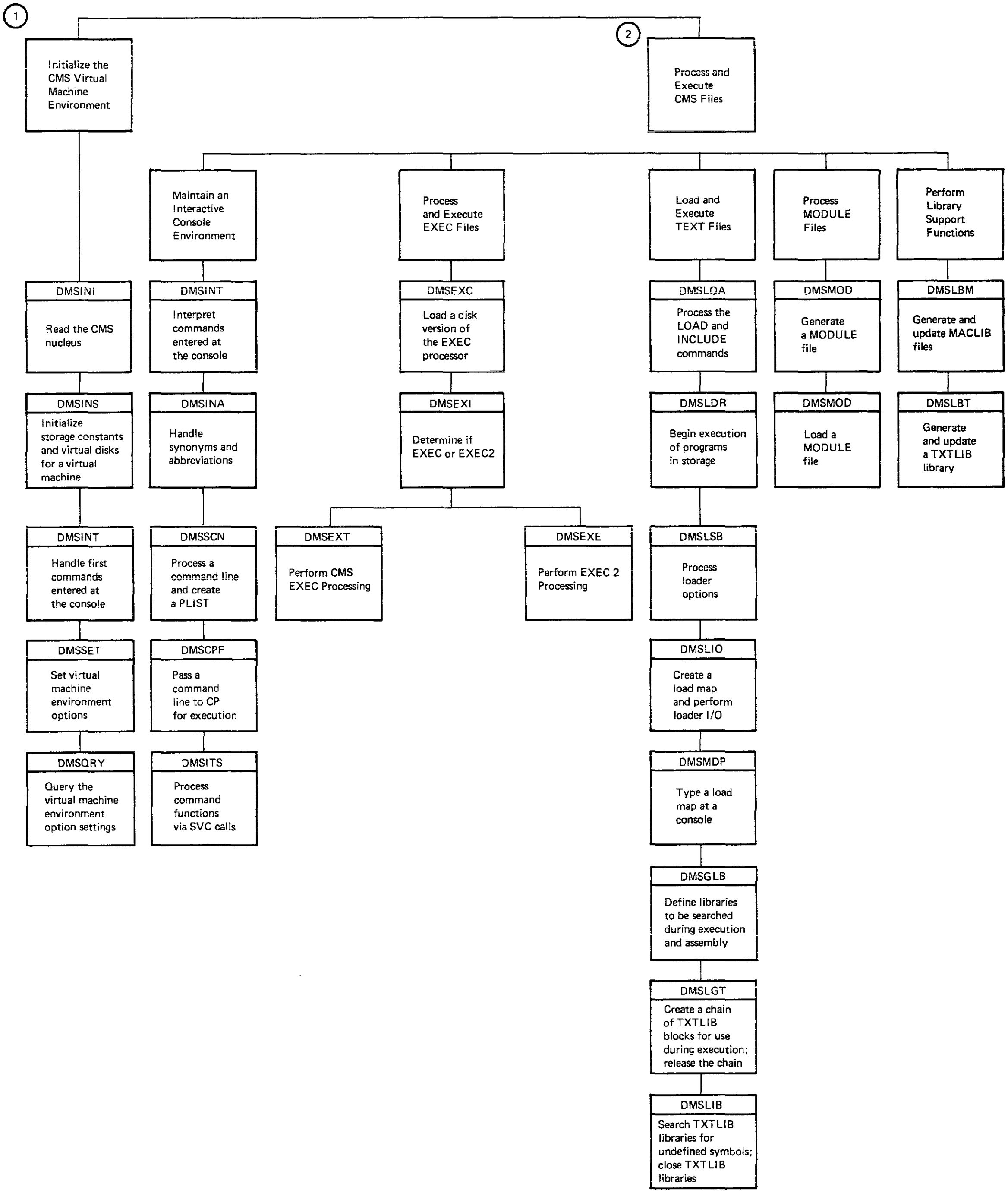


Figure 9. Details of CMS System Functions and the Routines that Perform Them (Part 1 of 4)

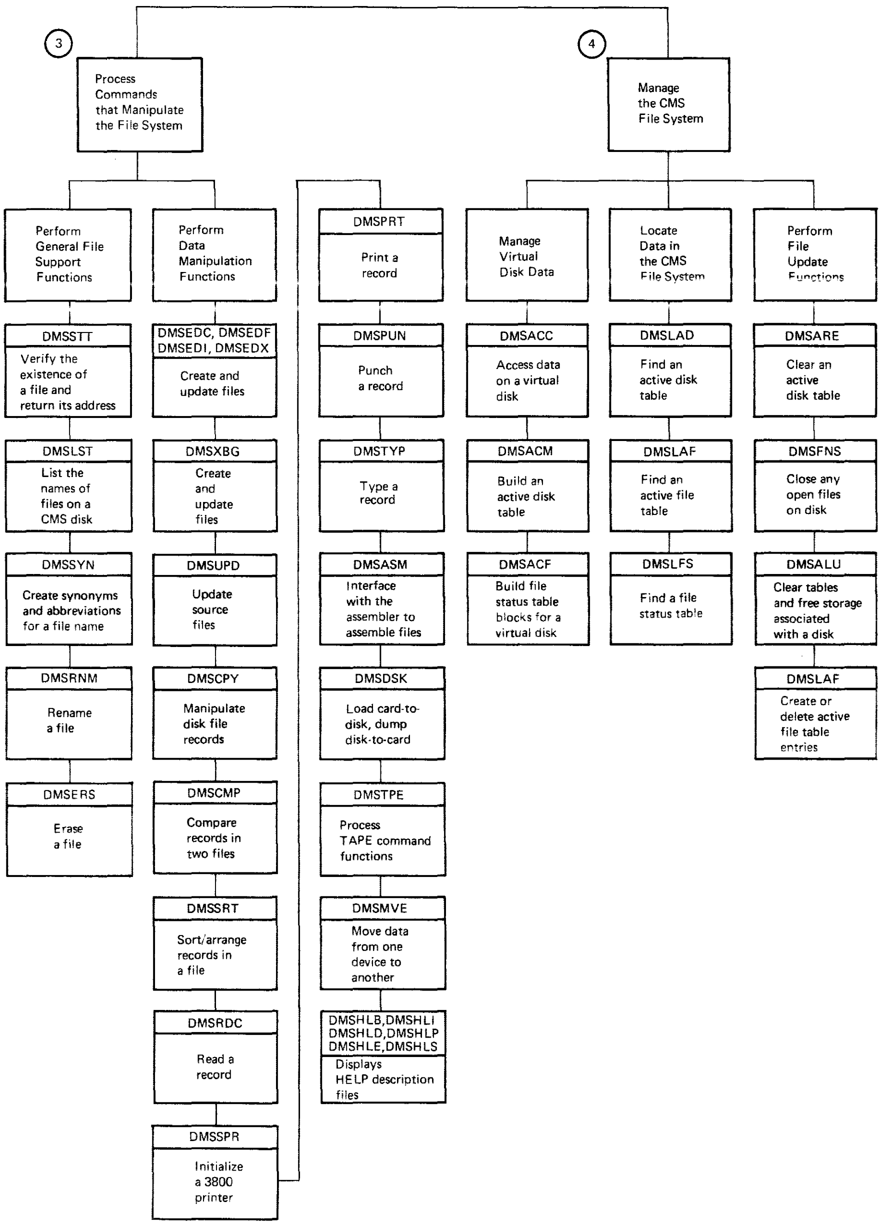


Figure 9. Details of CMS System Functions and the Routines that Perform Them (Part 2 of 4)

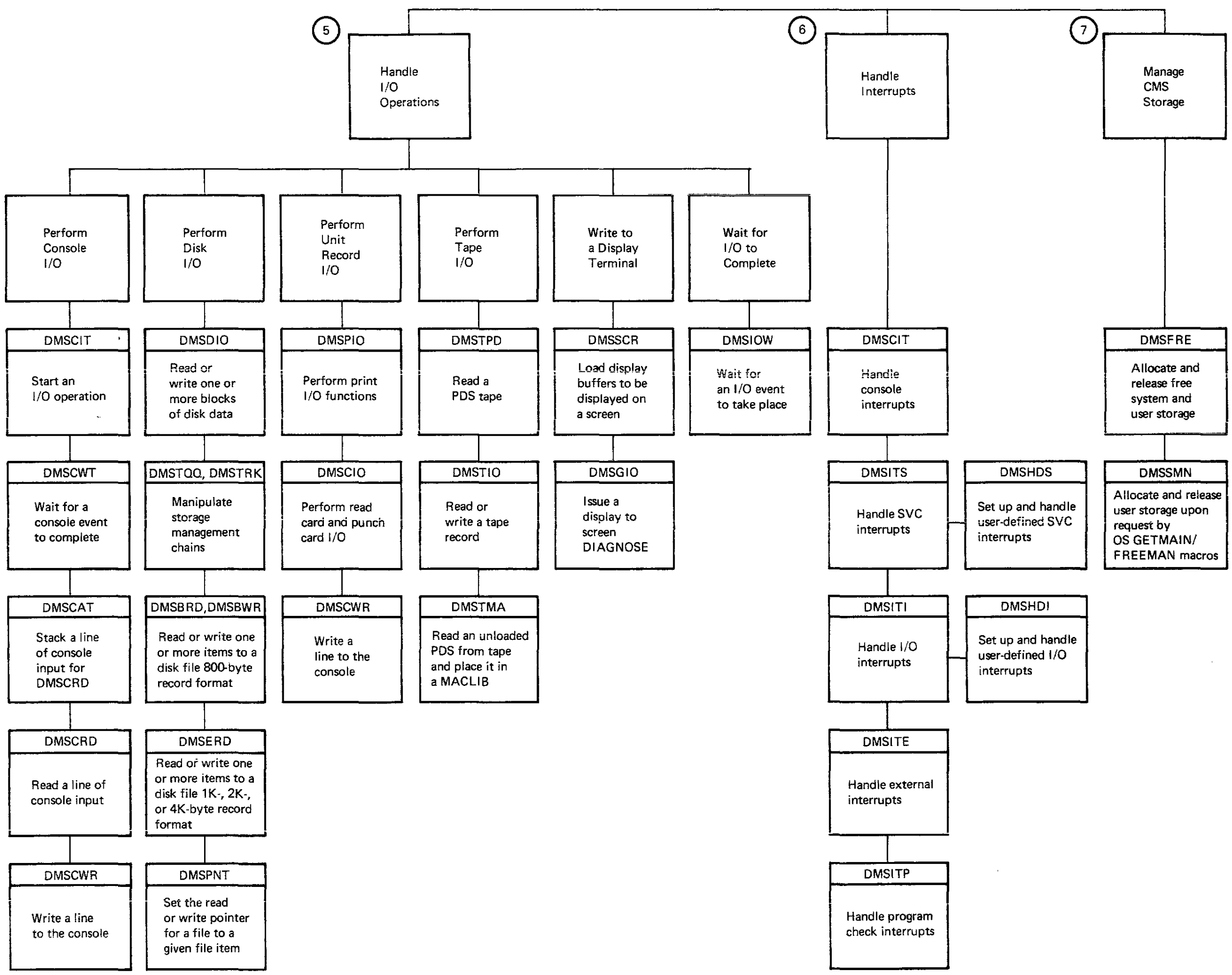


Figure 9. Details of CMS System Functions and the Routines that Perform Them (Part 3 of 4)

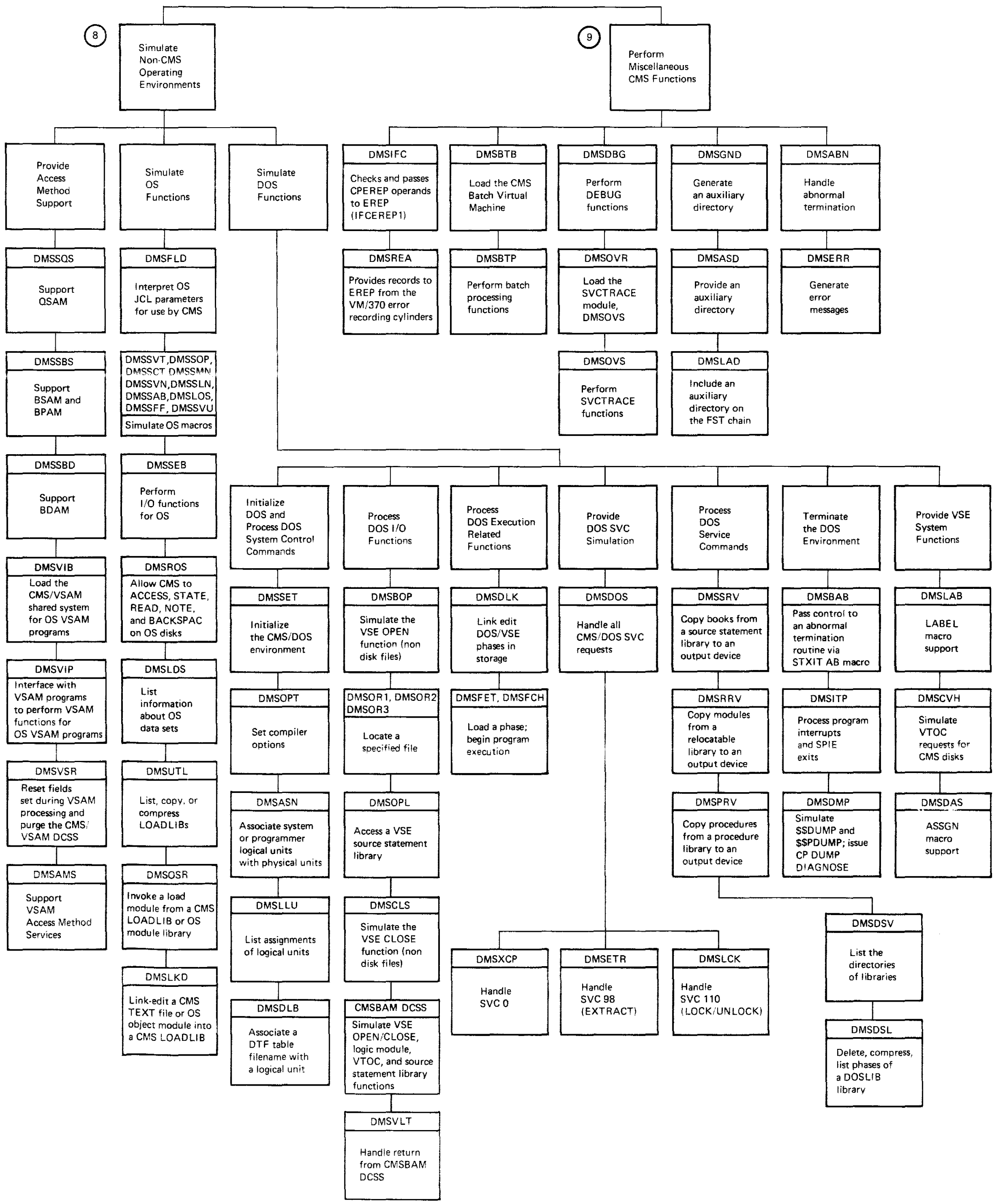


Figure 9. Details of CMS System Functions and the Routines that Perform Them (Part 4 of 4)

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

DMSINS
Saves the address of this virtual machine in NUCON.

DMSLAD
Locates and returns the address of the ADT for this virtual machine.

DMSFRE
Allocates free storage to be used during initialization.

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

DMSACM
Reads the S-disk ADT entry and builds the SSTAT.

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

DMSINS
Stores the address of SSTAT into ASSTAT and ADTFDA in NUCON.

DMSALU
Sorts the entries in the SSTAT.

PROCESSES IPL COMMAND LINE PARAMETERS

DMSINS
Checks for parameters BATCH, SEG=, ZER=, or AUTOCR. If BATCH is specified, DMSINS sets the flag BATFLAGS. If SEG= is specified, DMSINS loops through again to read the segment name. If ZER= is specified, DMSINS locates the CMSZER 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.

DMSINS
Issues DIAGNOSE 24 to obtain the device type of the console.

DMSCWR
Writes the system id message to the console.

DMSCRD
Reads the IPL command line from the console.

DMSSCN
Puts the IPL command line in PLIST format.

DMSINS
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 ccmmand 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.

If ZER= has been specified, the DIAGNOSE 64 FINDSYS function is issued to determine whether the CMSZER segment specified exists, and does not overlap the size of the virtual machine. If it exists and can be used, a DIAGNOSE 64 LOADSYS is issued and the segment is checked for validity, along with the optional saved shared SSTAT and YSTAT. If the segment can be used, the appropriate pointers in DMSNUC are relocated to point to the CMSZER segment. If the segment cannot be found, CMSZER cannot be used.

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.

Modifying a 3800 Named System

The IMAGEMOD command allows an installation to modify an existing 3800 named system without the need for generating from scratch a completely new one. Before, with the IMAGELIB command, a user had to construct a 3800 named system from a control file that listed all the members to be included. The IMAGELIB command contained no means for modifying an existing 3800 named system. Therefore, a system with, for example, 150 members, had to be totally reconstructed each time a member was added, deleted, or replaced. The IMAGEMOD command eliminates this problem by manipulating only the specific members of a 3800 named system that require changing. The format of the command is:

```
| IMAGEMOD | {GEN | ADD | REP | DEL | MAP}
|          |     libname
|          |     modname [modname]...
|          |     [TERM|PRINT|DISK]
```

For further information, refer to the *VM/SP Operator's Guide*.

PROCESSING THE IMAGEMOD COMMAND

Module DMSIMA performs the following steps when processing the IMAGEMOD command:

1. Analyze the input PLIST for syntax. If there is an error, exit with a return code of 2 and issue the appropriate message:

   - DMSIMA001E = NO MODULE NAME SPECIFIED
   - DMSIMA003E = INVALID OPTION 'option'
   - DMSIMA014E = INVALID FUNCTION 'function'
   - DMSIMA046E = NO LIBRARY NAME SPECIFIED
   - DMSIMA047E = NO FUNCTION SPECIFIED

2. Obtain maximum storage area (via GETMAIN macro).

3. Unless the GEN function is specified, read named system into storage just obtained with DIAGNOSE code X'74'. Leave the first 10 pages of storage empty. This permits later expansion by 10 members.

4. Determine the type of function requested:

   - MAP
   - DEL
   - GEN
   - ADD
   - REP

5. If the function requested is MAP, scan the named system directory and format the following information about each member:

   - Name
   - Relative displacement
   - Total size

   Determine the option requested. If the option is TERM, PRINT or DISK, place the formatted information on the user's terminal, virtual printer, or in the CMS file named 'libname MAP A5' respectively.

6. If the function requested is DEL, delete the member from the directory and the data area of the named system. Compress the named system by moving up the remaining members to take up the space vacated by the deletion. If the member is not found, issue message DMSIMA013E.

7. If the function requested is GEN, construct a skeleton named system in virtual storage. This skeleton system has no members initially. Then proceed as if the function were ADD.

8. If the function requested is ADD, load the member into the CMS transient area. If a load error occurs, issue DMSIMA346E and exit with return code of 6. Add the new member entry to the end of the named system directory. If virtual capacity were exceeded by this addition, issue DMSIMA109E and exit with return code of 2. During this process, the directory is moved back in storage one page to prevent new data from overlaying existing data. Move the new member data to the end of the named system residing in user virtual storage. Modify the directory entries after this move takes place. If the member already exists, issue message DMSIMA751E and exit with return code of 4.

9. If the function requested is REP, concatenate the DEL and ADD functions. In other words, perform the DEL function and then the ADD functicn for the specified member.

10. Scan the input command line for more members to be processed. If there are no more members, or if the number of members has reached the maximum (10), write the changed named system back to disk via DIAGNOSE code X'74' (unless this was a MAP function request) and exit. Otherwise, process the next member according to the function requested.

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 ccmmand 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/SP 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 or CMSBAM segments. If the segment is not already loaded, issues DIAGNOSE 64 LOADSYS to load it.

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

DMSSET
Sets up the LOCK/UNLOCK resource table.

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

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 2 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 appropriate user's disk for the 800-byte records on disk, or to update the file directory and the allocation map, or the appropriate user's disk for the 1K-, 2K-, or 4K-byte records on disk. DMSINT also frees the TXTLIB chain and releases pages of storage if required.

After updating the 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)
```

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)".

Whenever SVC 202 is called, a tokenized or untokenized parameter list (PLIST) can be specified. In both cases, register 1 points to an eight-character string defining the symbolic name of the routine or command being called. The SVC handler will examine only the name and the high-order byte of register 1.

Tokenized PLIST: For a tokenized parameter list, the symbolic name of the function being called (8 character string, padded with blank characters on the right if needed) will be followed by extra arguments depending on the actual routine or command being called. These arguments must be "tokenized" (that is, have a maximum length of eight characters, padded on the right with blank characters if shorter than eight characters). Extra information on the origin of the call is provided by the high-order byte of register 1. If the contents of this byte is equal to:

X'0E' - the call is the result of a command invoked from an EXEC file with the "&CONTROL NOMSG".

X'0D' - the call is the result of a command invoked from an EXEC with "&CONTROL MSG" (that is, messages are to be displayed).

X'0C' - the command is called as a result of it's name being typed at the terminal. This flag byte may be used, for example, to recognize the need for human readable messages instead of return codes.

X'00' - the call did not originate from an EXEC file or a command typed at the terminal.

Untokenized PLIST: For an untokenized parameter list, no restriction is put on the structure of the arguments list passed to the called routine or command. The high-order byte of register 1 contains X'01' or X'02'. X'01' means a normal hierarchy search is done in the manner described under the "SEARCH HIERARCHY FOR SVC 202" section of this manual. If it contains X'02', the search for the called routine is limited to the SUBCOM list (see the section entitled "Dynamic Linkage/SUBCOM" in this manual). Register 0 points to the untokenized PLIST which is constituted of four consecutive words:

```
¹DC A("Reserved Word")
²DC A(CMDBEG)
³DC A(CMDEND)
⁴DC A(0)
```

where the last two addresses are defined by:

```
CMDBEG EQU *
        DC C'QUERY INPUT'
CMDEND EQU *
```

CMDBEG EQU * indicates the beginning of the argument list and CMDEND EQU * indicates the end of the argument list.

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

----

[1]The first word is reserved.
[2]The second gives the beginning address of the argument list.
[3]The third gives the address of the byte immediately following the end of the argument list.
[4]The fourth word is optional. Any words following this word are available for passing information between the calling program and the program being called.

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.

USER-HANDLED SVCs: 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.

OS MACRO SIMULATION SVC CALLS: 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.

VSE SVC CALLS: 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 VSE 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.

INVALID SVC CALLS: 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 nct 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

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

RETURNING TO THE CALLER

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

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

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 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.

Loading and Executing 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

Function

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.

Entry

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.

Operation

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

Function

This routine creates loader table and ESID table entries for the card-specified control section.

Entry

This routine has one entry point, location C3AA3. The routine is entered from the ESD card analysis routine.

Operation

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

Function

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

Entry

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.

<u>Note</u>: 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

<u>Function</u>

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

<u>Entry</u>

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.

<u>Operation</u>

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 to 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").

Exits
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

Function
This routine handles the ENTRY and LIBRARY control cards.

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

Operations
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.

ENTRY Card
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.

LIBRARY Card
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 (Pseudo 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)	Absolute address
LD (ENTRY)	Absolute address
CM (COMMON)	Absolute 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.

Load and Execute Member of LOADLIBS

The OS relocating loader support consists of two members of the CMSSEG discontiguous shared segment. The members are the relocating program

(DMSLOS) and the overlay program (DMSSFF). In addition, the OSRUN command (DMSOSR) allows the user to invoke directly from the console a program residing in a CMS LOADLIB or an OS module library. DMSOSR executes in user storage.

When a user program invokes the LINK, LOAD, XCTL, or ATTACH SVC, DMSSLN calls DMSLOS to search the libraries in the LOADLIB global list for the specified member name. If found, DMSLOS loads and relocates the requested program from either an OS module library (for example, SYS1.LINKLIB) or a CMS LOADLIB (created by the LKED command). If the member is not found, return is made to DMSSLN to search for a TEXT file or a member of a TXTLIB by that name.

The program exists in the library as text records, directly followed (when required) by control, relocation, and position records. DMSLOS obtains, via the BLDL macro, the information necessary to start loading the program from the PDS directory entry for the program. Then, text records and control records are read alternately, the proper addresses are modified, and return is made to DMSSLN.

The OSRUN command generates a LINK SVC and therefore follows the same path described in the preceding paragraphs. However, if the requested member is not found in searching the libraries specified in the LOADLIB global list, a search is made for a default library ($SYSLIB LOADLIB); TEXT files and TXTLIB members are not searched.

For detailed information on the library record formats, see the _OS/VS Linkage Editor Logic_, SY26-3815.

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.

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 26 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.

How CMS Files Are Organized in Storage for an 800-byte Record

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.

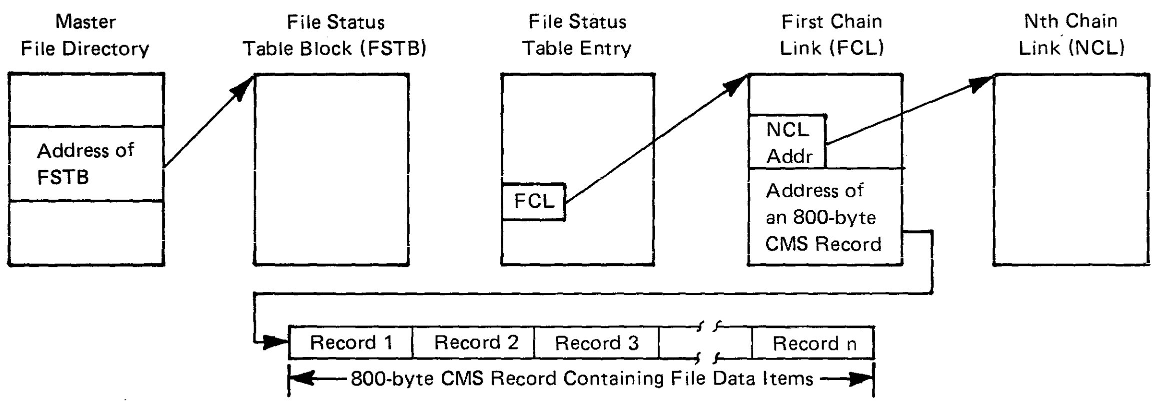


Figure 12. How 800-Byte CMS File Records Are Chained Together

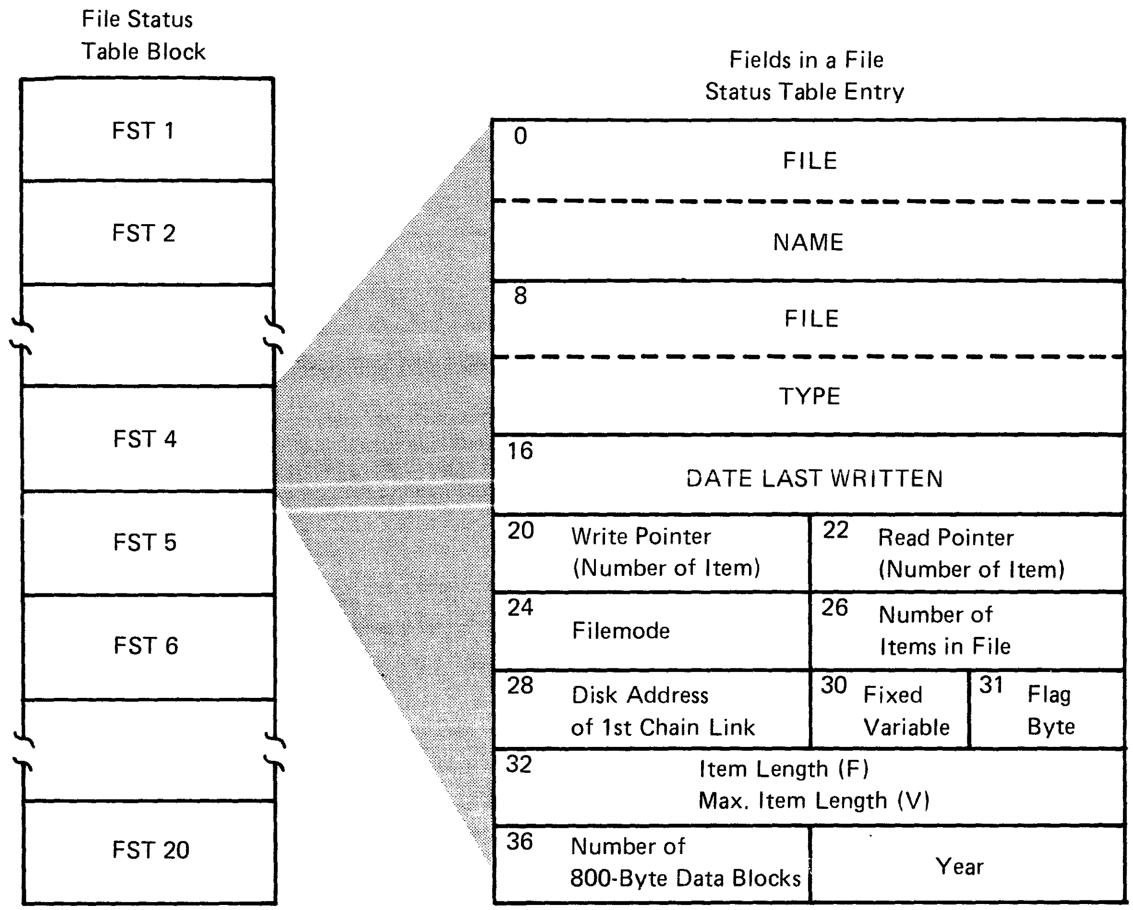


Figure 13. Format of a File Status Block; Format of a File Status Table (for 800-Byte Disk Format)

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:

```
                (Item Number - 1) x Record Length
    locations = --------------------------------
                              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.

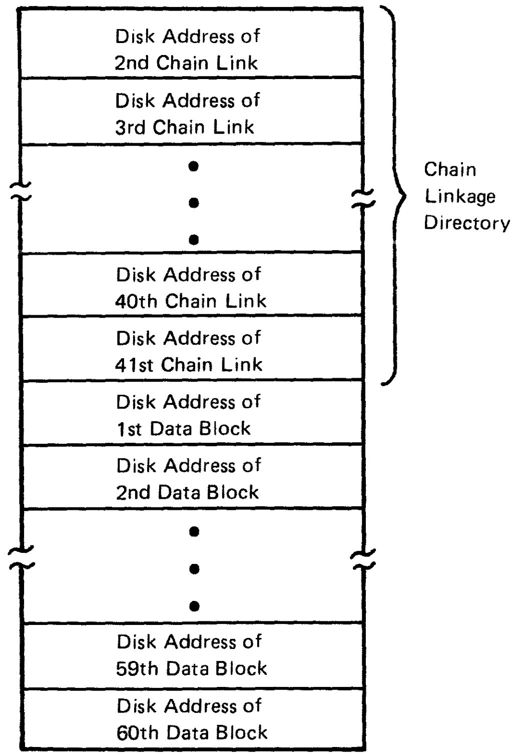
PHYSICAL ORGANIZATION OF VIRTUAL DISKS

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.

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.



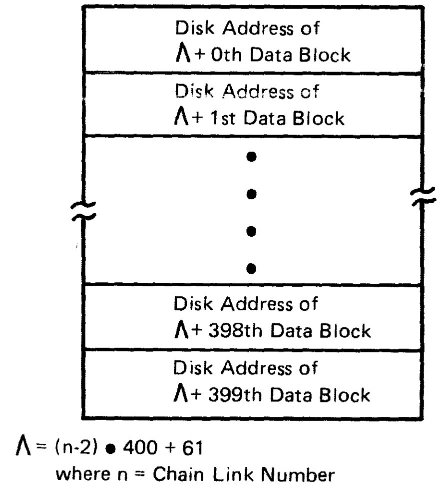


Figure 14. Format of the First Chain Link and Nth Chain Links

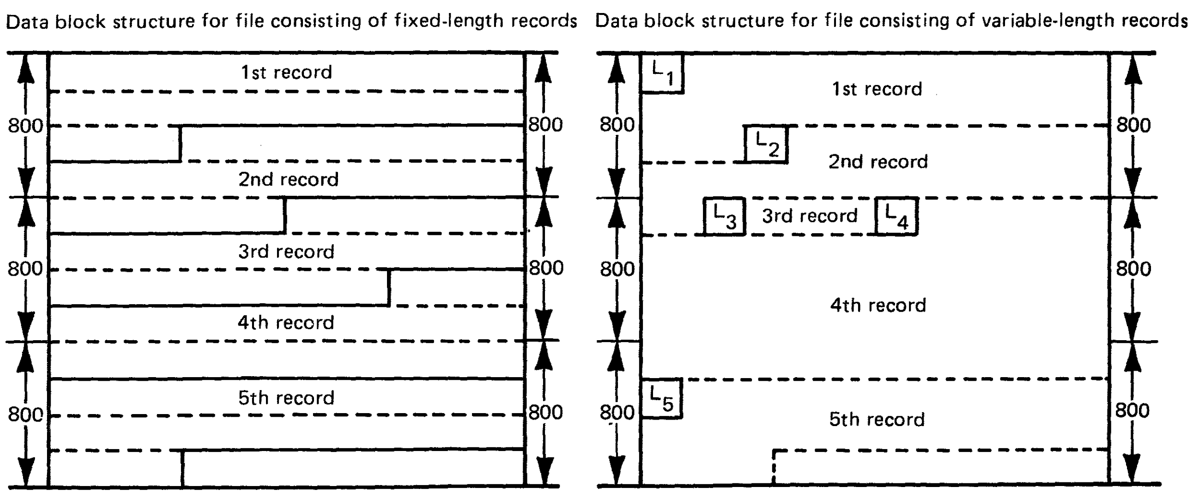


Figure 15. Arrangement of Fixed-Length Records and Variable-Length Records in Files

- 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 main 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

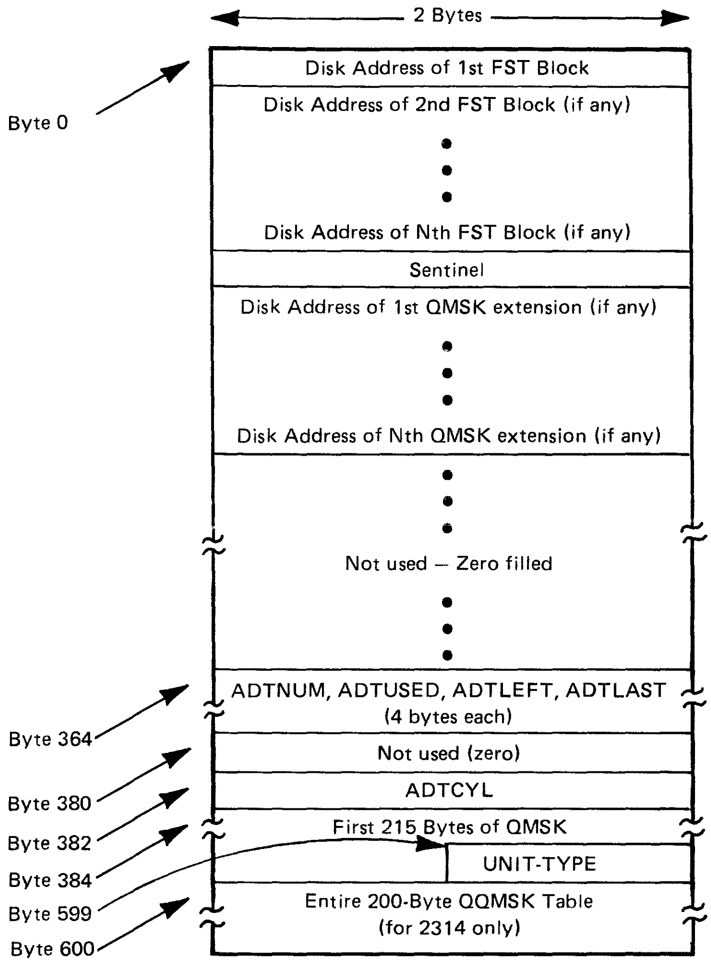
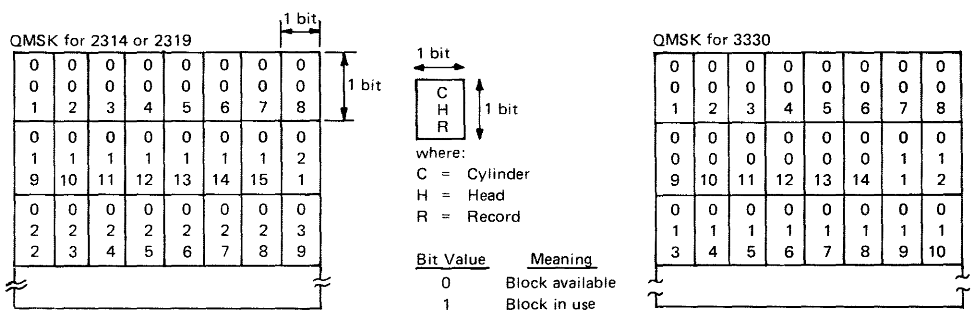


Figure 16. Structure of the Master File Directory



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

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.

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 2 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.

How CMS Files Are Organized in Storage for 1K-, 2K-, or 4K-Byte Records on Disk

CMS files are organized by three types of blocks; the file status table (FST), pointer blocks, and file records. Figure 18 shows how these types of 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 1K-, 2K-, or 4K-byte CMS blocks 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 highest level pointer block or only data block. If it is a pointer block, this block contains addresses of the next lower level pointer blocks or the data blocks that contain the actual data for the file.

The FSTs are grouped into 1K-, 2K-, or 4K-byte CMS blocks called FST blocks (these are sometimes referred to in listings as hyperblocks).

Each FST block contains 16, 32, or 64 FST entries respectively (an FST is 64 bytes long), each describing the attributes of a separate file. Figure 19 shows the structure of an FST block and the fields defined in the FST.

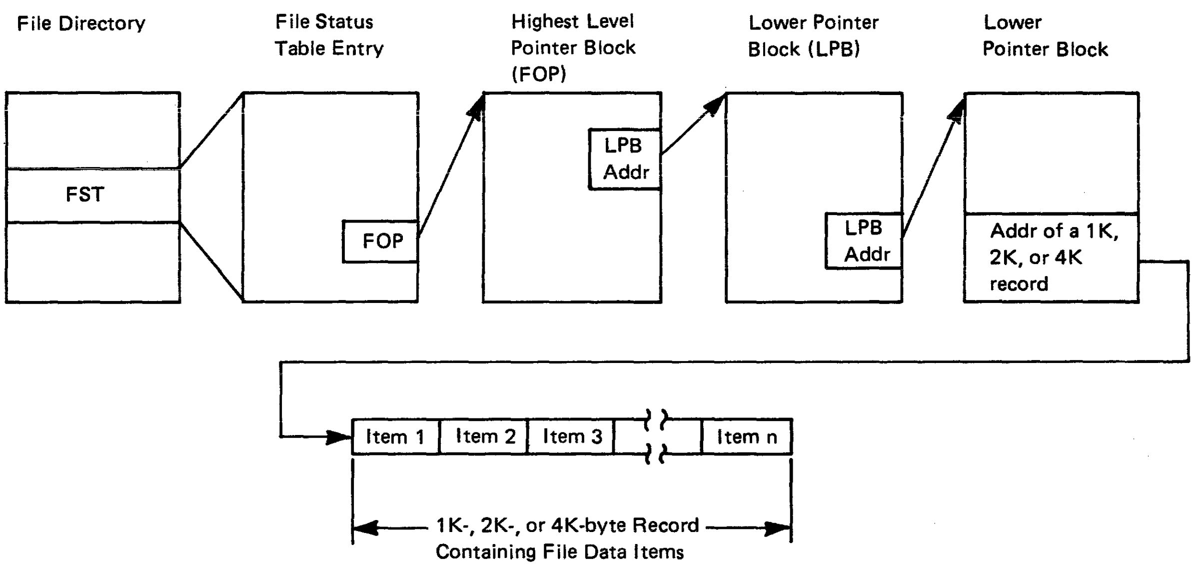


Figure 18. How 1K-, 2K-, or 4K-Byte CMS File Records Are Chained Together

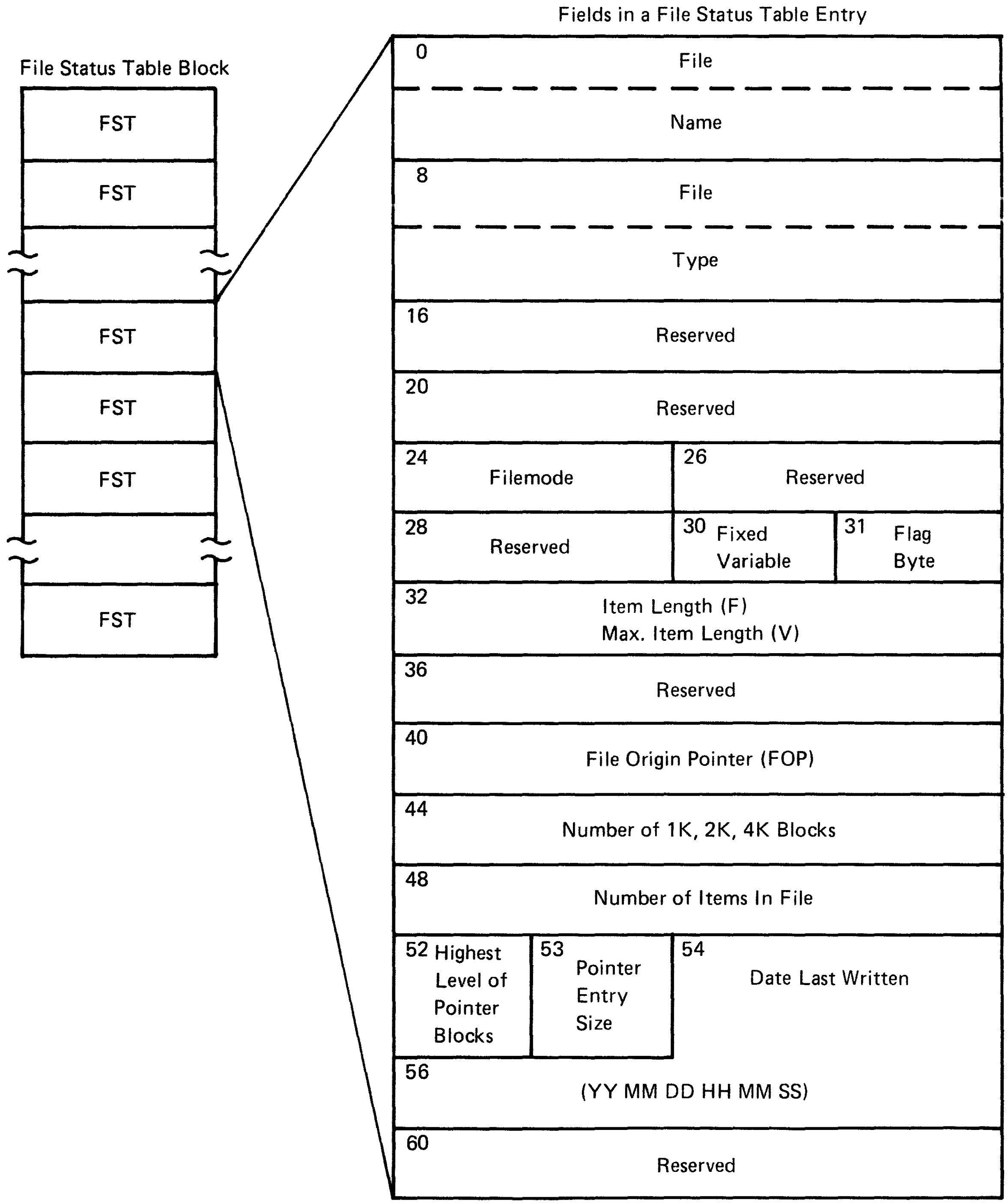


Figure 19. Format of a File Status Table Block and File Status Table (For 1K-, 2K-, and 4K-Byte Disk Format)

POINTER BLOCKS

Pointer blocks are 1K-, 2K-, or 4K-byte blocks of storage that chain the records of a file. There are up to five levels of pointer blocks. All but the first level of pointer blocks contain the fullword disk address of the next lower level pointer block. The level-one pointer blocks contain the fullword disk addresses of the data blocks of the file (see Figures 20 and 21).

There are two types of pointer blocks; pointer blocks for fixed files which are as described above, and pointer blocks for variable files. For the variable files, each pointer block entry is three fullwords long. The first fullword holds the disk address of the next lower level pointer block, the next fullword holds the highest item number contained in this lower corresponding pointer block, and the last fullword holds the displacement, at the data level, to the first identified item contained in a lower corresponding pointer block. CMS blocks are not shared by files.

Each entry of a level-one pointer block is composed of one fullword containing the disk address of the corresponding data block, one fullword containing the highest item number contained in this data block, and one fullword containing the displacement, in bytes, of the first identified item (if any) contained in this data block. This last fullword of the entry may hold the hexadecimal value X'FF...F', indicating that the item is spanned.

The last fullword of a pointer block holds the displacement, in bytes, of the last used entry, if one exists, in the block. This structure permits the creation of very large files. This file management system limits the maximum size for any CMS file to approximately $2^{31}-1$ times 1K-, 2K-, or 4K-byte records. The maximum size for an item is $2^{31}-1$ bytes for a fixed file, and 64K for a variable file.

Each pointer block or data block is prefixed in virtual storage with a header. This header holds an entry called DCHTRUNK that points to the upper level pointer block. Associated with the DCHTRUNK value is a displacement that indicates the corresponding entry in this upper level pointer block.

In virtual storage, each level of pointer block and the data block have an anchor in the corresponding Active File Table (AFT) and are forward and backward chained by the prefix.

CMS BLOCK FORMATS

CMS blocks are 1K-, 2K-, or 4K-byte disk records 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 20 shows how pointer blocks are chained together.

CMS file items 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 1K-, 2K-, or 4K-byte records as are required to accommodate them. Each CMS block (except the last) is completely filled and items that begin in one CMS block can end in the next CMS block. Figure 20 shows the arrangement of items in files containing fixed-length items and files containing variable-length items.

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

$$\text{location} = \frac{(\text{Item Number} - 1) \times \text{Record Length}}{1K,\ 2K,\ \text{or}\ 4K}$$

where the quotient is the sequential number of the data block and the remainder is the displacement of the item into the data block.

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

PHYSICAL ORGANIZATION OF VIRTUAL DISKS

Virtual disks are physically organized in 1K-, 2K-, or 4K-byte disk records. Records 1 and 2 of each user disk are reserved for IPL. Record 3 contains the disk label. The first block of the file directory is alternately exchanged between record 4 and record 5 when the directory is rewritten to disk. The remaining records on the disk contain information such as allocation map blocks, FSTBs, pointer blocks, and the individual file records as discussed above.

CMS disk structures that reside on FB-512 devices are 1024-, 2048-, or 4096-byte CMS block format. The required number of 512-byte physical FB-512 disk records are logically concatenated together to form each CMS block. For example; on a 1024-byte format disk, FB-512 physical record numbers 0 and 1 (origin 0) are used together to form CMS block 1 (origin 1). The FB-512 label occupies FB-512 block 1 (origin 0) leaving CMS blocks 2 and 3 available for general use.

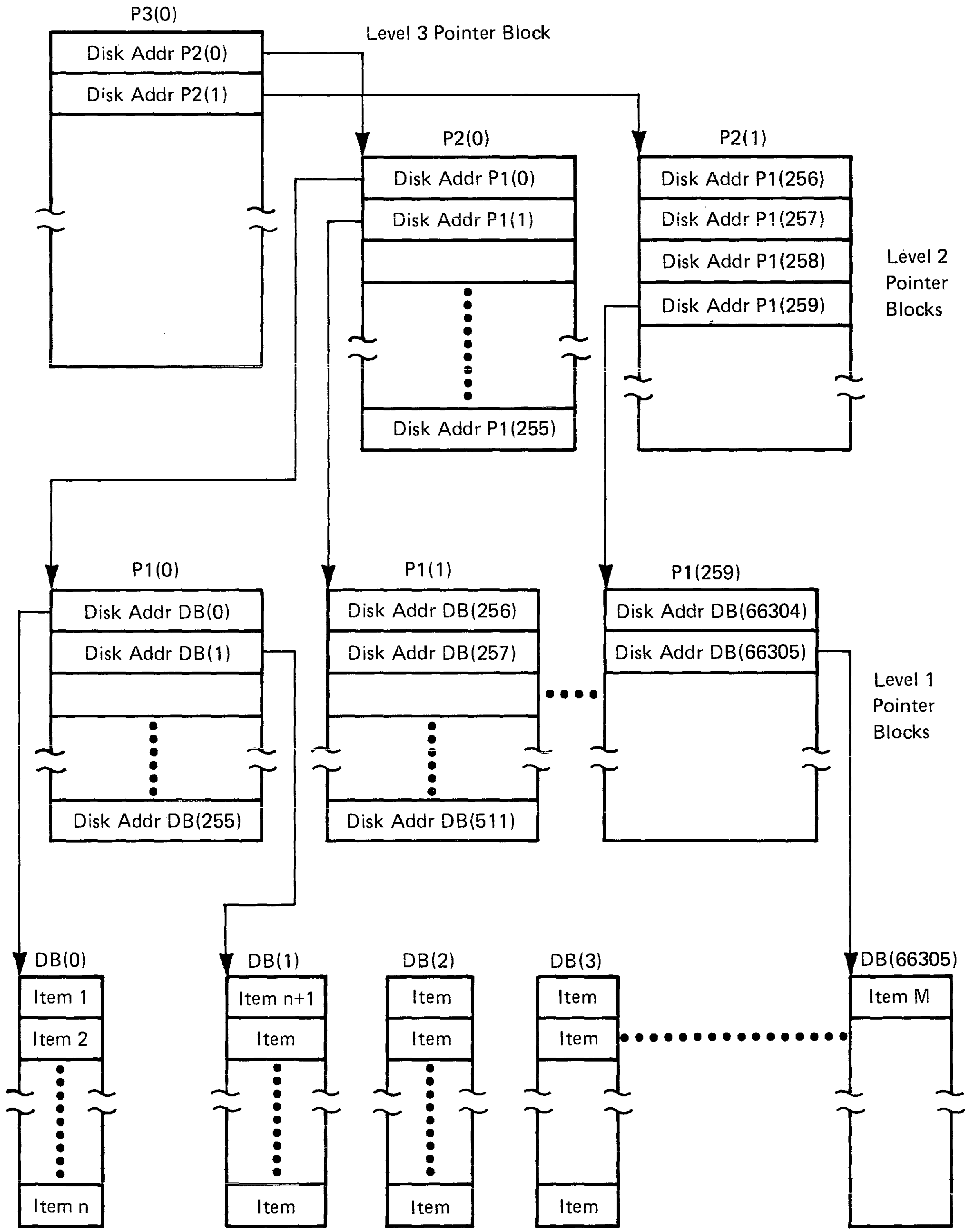


Figure 20. Format of Level 3 Pointer Block Fixed-Length Record File

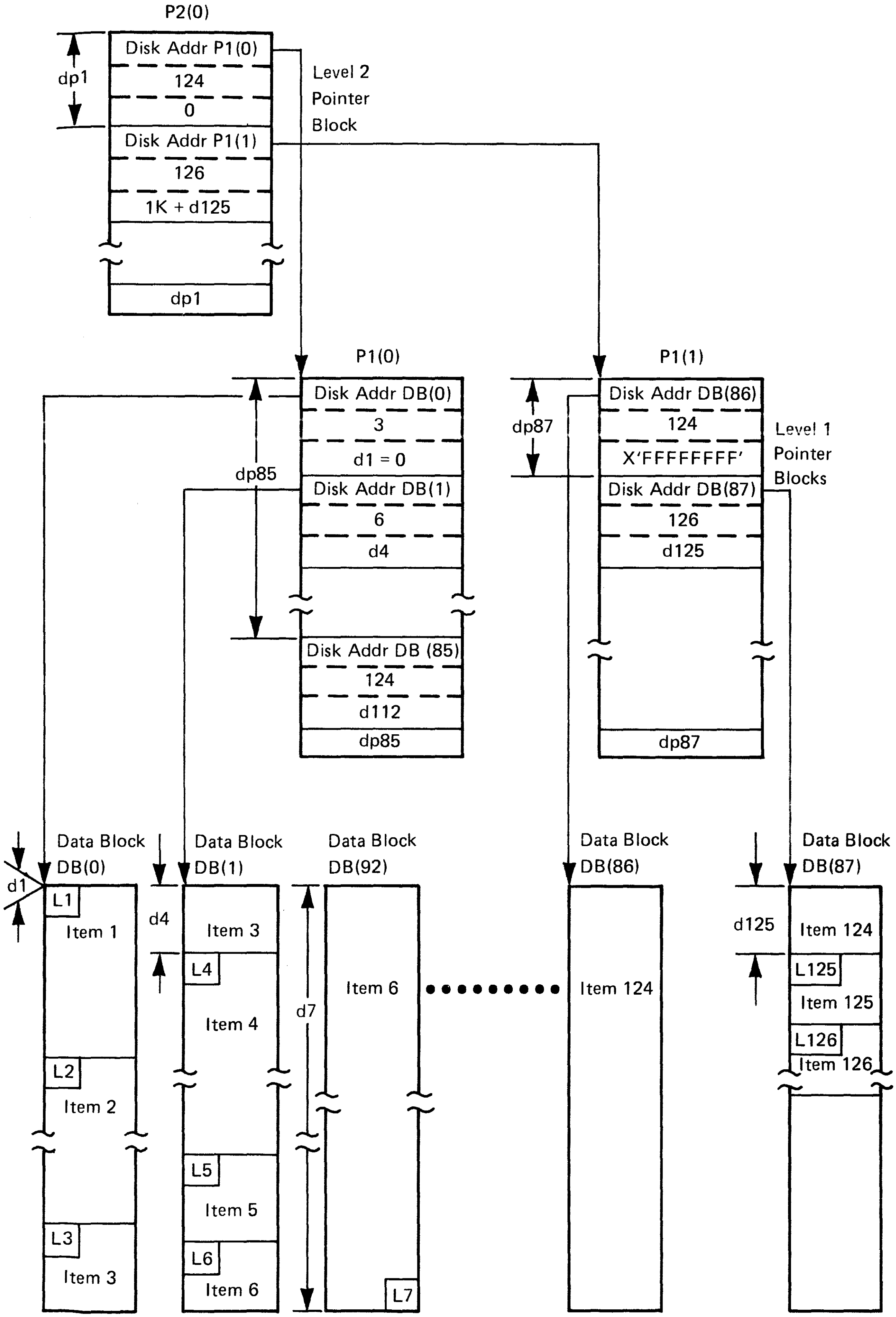


Figure 21. Format of Level Two Pointer Block Variable-Length Record File

THE FILE DIRECTORY, THE ALLOCATION MAP, AND THE DISK LABEL

The file directory and the allocation map have the same organization as files. The directory contains FSTs and the first block resides on cylinder 0, track 0, record 4 or record 5 of each virtual disk. The record number (4 or 5) is maintained in the field disk origin pointer of the disk label.

The directory itself is described by an FST that is the first FST in the first block; the filename for the directory is binary zero (except for byte 4 which is binary 1) and a filetype of "DIRECTOR".

The allocation map is described by an FST that is the second FST in the first block of the directory; the filename is binary zero (except for byte 4 which is binary 2) and a filetype of "ALLOCMAP".

The disk label resides on cylinder 0, track 0, record 3; it is 80-bytes long and contains the following information:

ADTIDENT CMS1 is the label identifier.

ADTID Six characters given by the user are the volume identifier.

ADTDBSIZ One fullword; contains the disk block size that the user chooses at format disk time (1K, 2K, or 4K).

ADTDOP One fullword; contains records 4 or 5 depending upon the actual directory first data block address.

ADTCYL One fullword; contains the number of formatted cylinders available for CMS files.

ADTMCYL One fullword; contains the maximum number of formatted cylinders, that is, the size of the disk.

ADTNUM One fullword; the total number of 1K-, 2K-, or 4K-byte blocks on the user's disk.

ADTUSED One fullword; the number of blocks currently in use on the disk.

ADTFSTSZ One fullword; the size of the FST (64 bytes).

ADTNFST One fullword; the number of FSTs per block.

ADTCRED Six characters; the disk creation date (YYMMDDHHMMSS).

KEEPING TRACK OF READ/WRITE DISK STORAGE: ALLOCATION MAP

In CMS, disk space is composed of 1K-, 2K-, or 4K-byte blocks chained together. Because disk space management only determines the availability of blocks, not extents, it need not allocate disk space contiguously. 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 the allocation map. The allocation map contains bits, each of which is associated with a particular CMS block. The first corresponds to the first CMS block, the second bit corresponds to the second CMS block, and so forth.

When a bit in the allocation map 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 through disk space management, and the disk I/O routine, DMSDIO, finally converts this number to a CCHHR disk address or FB-512 block number.

When the system is not in use, a user's allocation may resides on the corresponding disk. During a session, it is maintained on disk but also resides in real storage. The allocation map is variable in length, depending on how many cylinders exist on the disk. The CMS disk may reside on the entire physical disk pack and is limited only by the physical limit of the disk pack.

A deallocation map exists in real storage when CMS disk blocks are deallocated. During a terminal session a block is recorded as deallocated by turning on its corresponding bit in the deallocation map.

When the disk is updated by rewriting the file directory and the allocation map, the current allocation map is formed by combining the allocation map and the deallocation map. In fact, a deallocation map block is created only for those allocation map blocks in which a CMS block is deallocated.

The allocation maps for read-only disks are not brought into storage because no space allocation is performed for a disk while it is in read-only status. They remain, as is, on the disk until the disk is accessed as a read-write disk.

Selective Directory Update

The file directory and the allocation map are built with CMS blocks (1K-, 2K-, or 4K-bytes). The selective directory update function takes place when the file directory and the allocation map must be updated on the corresponding disk. It writes on disk only the modified blocks of the directory (including required pointer blocks) and the entire allocation map.

DYNAMIC STORAGE MANAGEMENT: ACTIVE DISKS AND FILES

CMS disks are physically mapped in CMS blocks containing the file directory and the allocation map. CMS files on disk are mapped using FST blocks, pointer blocks, and 1K-, 2K-, or 4K-byte file data blocks.

In real storage all of this data is accessed by means of two DSECTs whose addresses are defined in DMSNUC, ADTSECT, and AFTSECT. 10 ADTSECTs reside in DMSNUC and the others (11 through 26) reside in free storage when the are used. Five AFTs reside in DMSNUC and the others reside in free storage. (See Figure 22).

Managing Active Disks: The Active Disk Table

The ADTSECT DSECT maps information in the active disk table (ADT). An ADT contains significant information about the CMS disk such as the anchors for pointer block levels and the data block for the file directory, the anchors for pointer block levels and the data block for the allocation map (if the disk is a read-write disk). The ADTSECT also contains disk label information.

Managing Active Files: The Active File Table

Each open file is represented in storage by an active file table (AFT). The AFT (defined by AFTSECT DSECT) contains data found on disk in FSTs, the anchors for pointer block levels and the data block for the file. The AFT also contains such information as the read pointer and write pointer of the file, the number of entries in a pointer block, the number of pointer block levels, and the length of a pointer block entry. Figure 22 shows the relationship between the AFT and other CMS 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 file directory, and the allocation map for the specified disk; also, DMSACM updates the ADT for the specified disk using information from the file directory and disk label.

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.

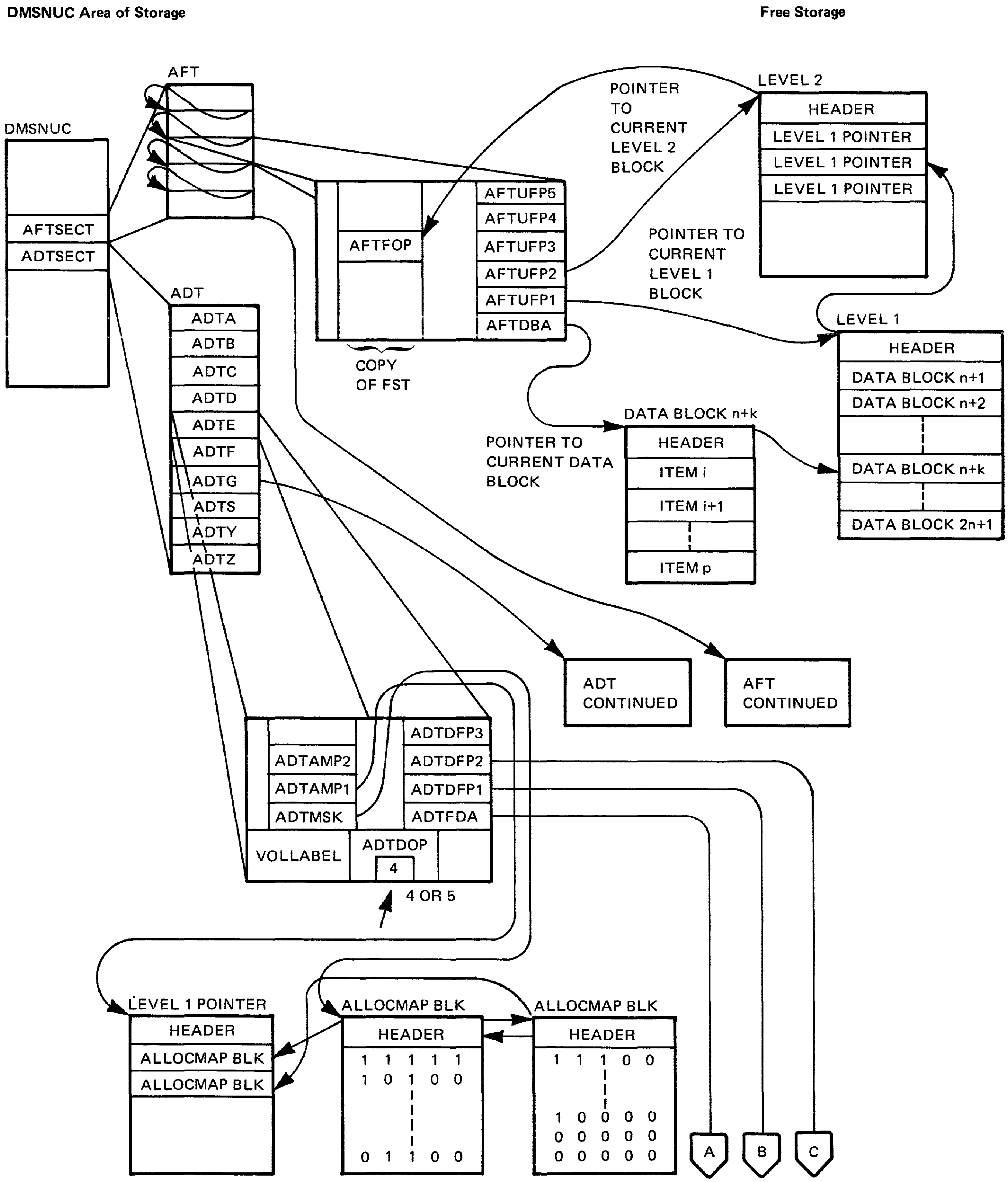


Figure 22. File System for a 1K-, 2K-, or 4K-Byte Record on Disk (Part 1 of 3)

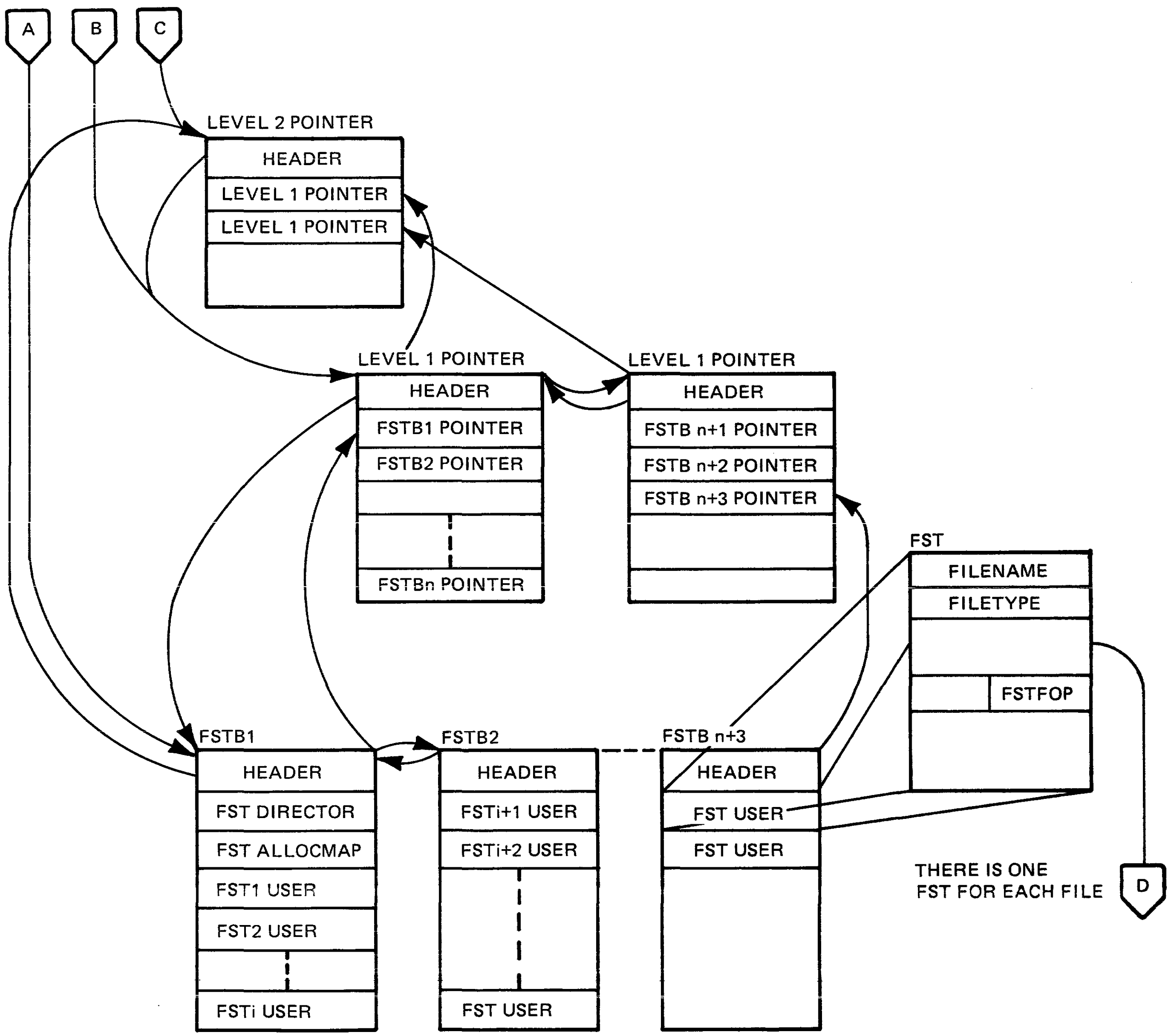


Figure 22. File System for a 1K-, 2K-, or 4K-Byte Record on Disk (Part 2 of 3)

Disk Storage CKD – DEVICE

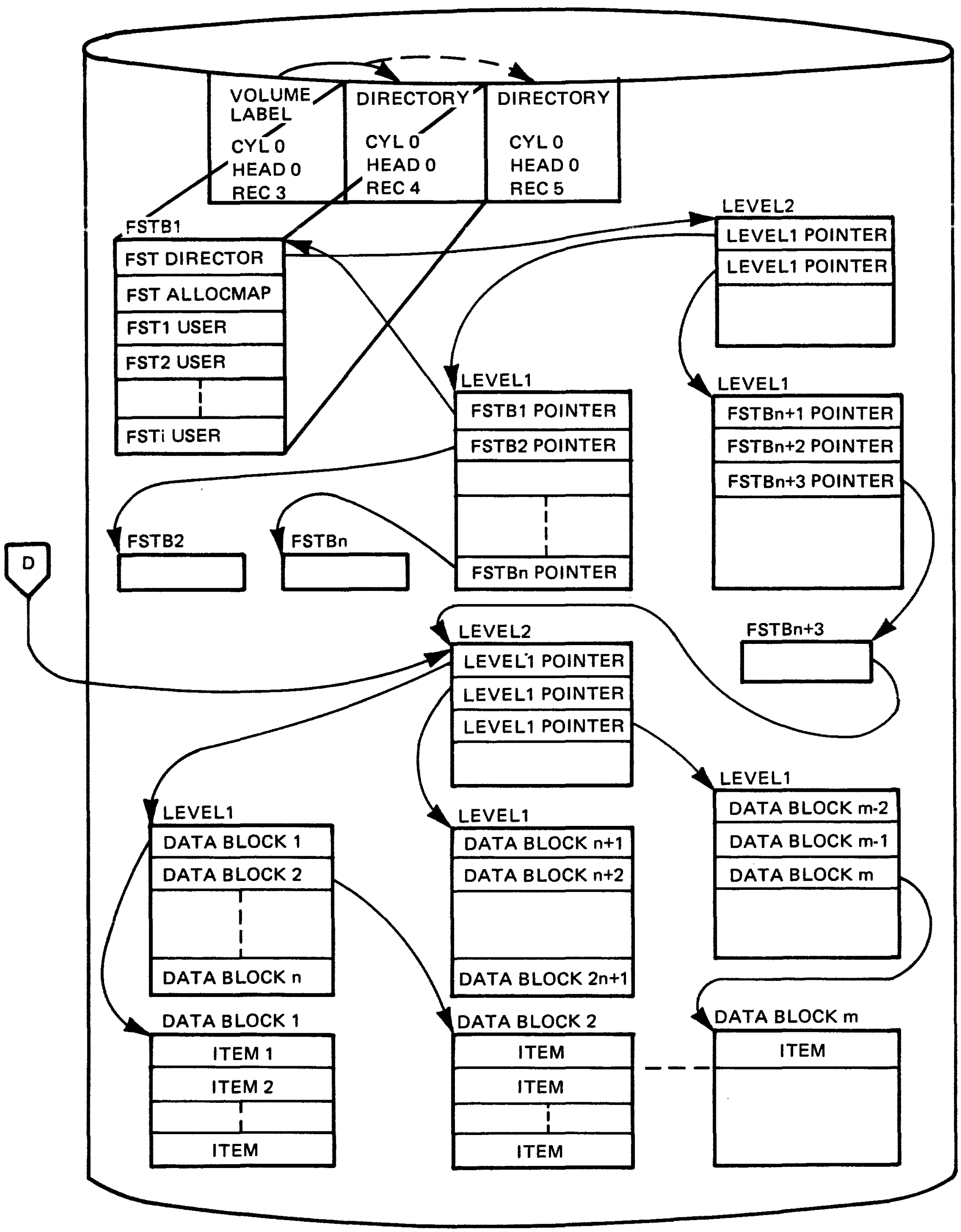


Figure 22. File System for a 1K-, 2K-, or 4K-Byte Record on Disk (Part 3 of 3)

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 23 shows the overall flow of control for I/O operations.

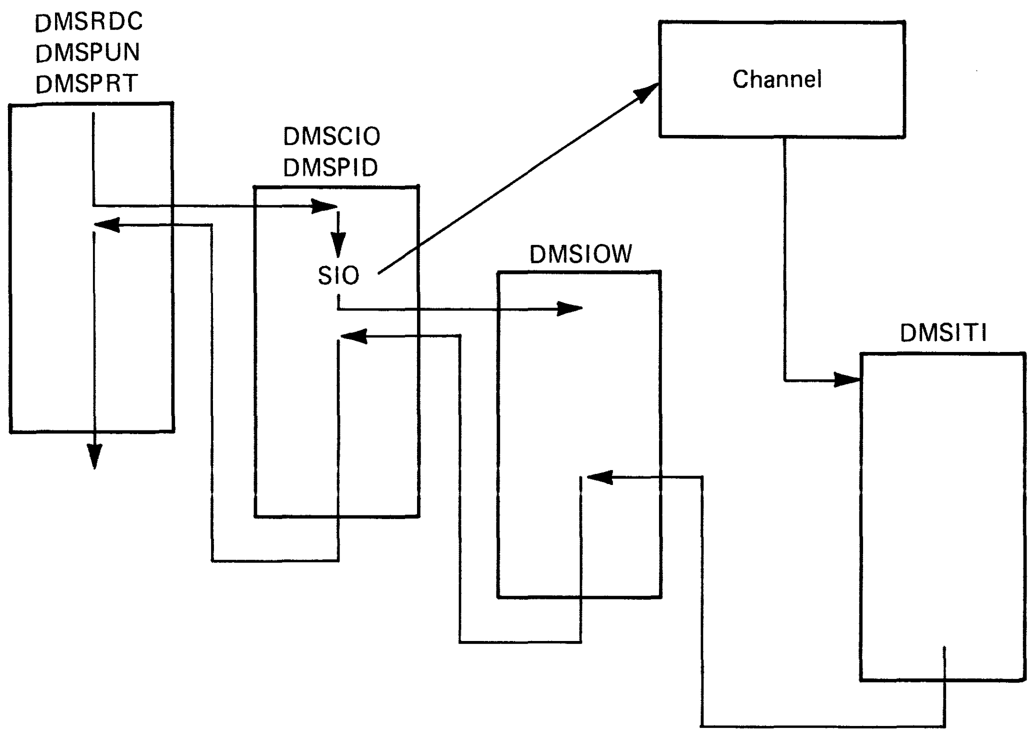


Figure 23. Flow of Control for Unit Record I/O Processing

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.

DMSBWR: Writes a block of data on disk.

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, and calls DMSPIO to print the designated file.

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: Builds appropriate printer CCW chain. 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.

THE SETPRT COMMAND

The CMS SETPRT command allows a CMS user to control the facilities of a virtual 3800 device defined for his virtual machine. The SETPRT command is similar in function to the OS SETPRT macro, allowing the user to request multiple character arrangement tables, loading of copy modifications, etc. The command uses the current CMS search order for locating disk files. Therefore, users can create their own character arrangement tables, copy modifications, etc. and print files with user defined characteristics. The SETPRT command writes 3800 CCWs and data to a virtual 3800 spool file to set up the real 3800 for the data to follow. If a file is created on a virtual 3800 and printed on a real printer of a different type, the 3800 load CCWs imbedded within the file are ignored and printing takes place as normal. However, this may create output that does not appear as originally intended. The format of the command is:

SETPRT	[CHARS [(cccc ....])]]
	[COPIES [(] nnn [)]]
	[COPYNR [(] nnn [)]]
	[FCB [(] ffff [)]]
	[FLASH [(] id nnn [)]]
	[INIT]
	[MODIFY [(] mmmm [)]]

DMSSPR process the SETPRT command in the following manner:

1. Accept input PLIST and analyze. If there are errors, issue a message to the user and exit.
2. Select the correct character set modules and load these modules into free storage.

3. Assign writeable character generation modules (WCGMs) and change the translate tables if necessary.

4. Issue SIOs to the virtual 3800 printer. In the case of an error, terminate processing and issue a message and appropriate return code.

5. Exit with a zero return code if the operation completes successfully.

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

DMSDIO: Initializes the CCW to perform read operations.

DMSLAD: Obtains the address of the disk from which to read or write.

DMSDIO: Determines the size of the record to be read or written.

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

DMSDIO: 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.

CMS Tape Label Processing

DMSLBD: Allows the user to specify tape label information that will be used by a program at execution time.

DMSTLB: Processes IBM standard tape labels for OS simulation, CMS/DOS, CMS commands, and the TAPESL macro. It also provides linkage to nonstandard user label routines for OS simulation and CMS commands. There are common tape label checking routines for input header and trailer labels and common tape label writing routines for output header and trailer labels. These common routines are used for all IBM standard label processing regardless of what operating system is being simulated.

DMSTIO: Reads or writes a tape record. Also performs tape control operations. Functions by issuing diagnose code X'20'.

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'04000') - This is the nucleus constant area. It contains pointers, flags, and other data maintained by the various system routines.

- CMS Nucleus First Part (X'04000' to approximately X'9000') - This area contains the following CMS Nucleus routines: DMSALU, DMSCIO, DMSVIB, DMSVSR, DMSDBD, DMSDBG, DMSFET, DMSTIO, DMSTLA, DMSTQQ, DMSITP, DMSABN, DMSITE, DMSPNT, DMSPIO, DMSLIO and DMSCPF.

- Low-core DMSFREE free storage area (approximately X'09000' 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 nc 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. In addition, a DIAGNOSE code X'10' instruction is sent to CP to release these pages between MAINHIGH and FREELOWE. 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 returned, this pointer is adjusted downward and the freed pages are released by issuing a DIAGNOSE code X'10' instruction to CP.

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 returned, this pointer is adjusted upward when whole 4K pages are completely free and the freed pages are released by issuing a DIAGNOSE code X'10' instruction to CP.

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. When FREELOWE is adjusted upward, the corresponding pages are released by issuing a DIAGNOSE code X'10' instruction to CP.

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. All the corresponding full pages contained in the freed block are released by issuing a DIAGNOSE code X'10' instruction to CP.

- The STRINIT macro instruction releases all storage allocated by any previous GETMAIN requests. All the corresponding full pages between MAINHIGH and FREELOWE are released by issuing a DIAGNOSE code X'10' instruction to CP.

- 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 first part has a nucleus storage key of X'F'.

* 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.

EFFECT ON CMS: 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.

RESTRICTIONS ON CMS: 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	cp-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 VSE 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, 1024, 2048, or 4096 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 VSE/VSAM. Therefore, the OS user is restricted to those services available under VSE/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 four discontiguous shared segments (DCSSs).

  -- The CMSAMS DCSS, which contains the VSE/VSAM code to support AMSERV processing.

  -- The CMSVSAM DCSS, which contains actual VSE/VSAM code, and the CMS/VSAM OS interface program for processing OS VSAM requests.

  -- The CMSDOS DCSS, which contains the code that supports VSE requests under CMS.

  -- The CMSBAM DCSS, which contains the SAM modules required in order for AMS to access SAM files.

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 area 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 VSE/VSAM access method services (AMS) program. Module DMSAMS loads VSE/VSAM AMS code contained in the CMSAMS DCSS by means of the LOADSYS DIAGNOSE 64. The AMS code requires the services of VSE/VSAM 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 24 shows the relationship in storage between the interface module DMSAMS and the CMSAMS and CMSVSAM DCSSs.

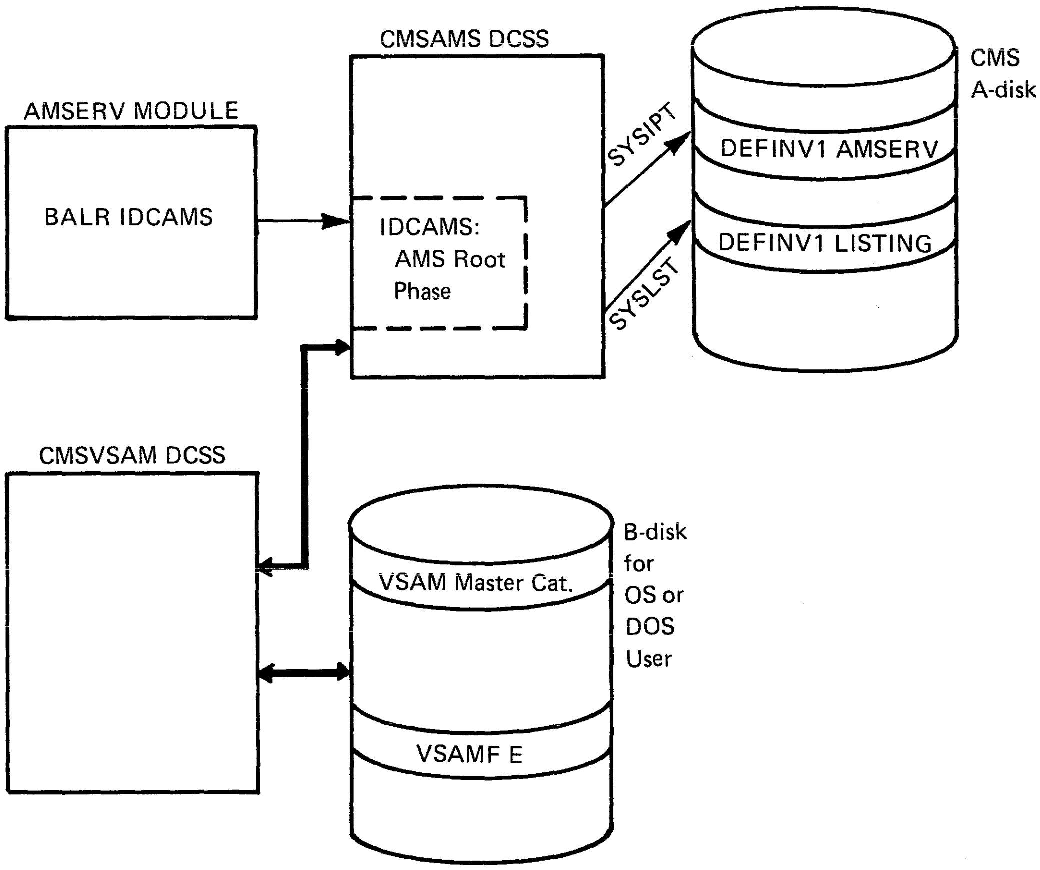


Figure 24. Relationship in Storage between the CMS Interface Module DMSAMS and the CMSAMS and CMSVSAM DCSSs

The following is a general description of the DMSAMS method of operation.

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 VSE/VSAM code. A VSE SVC 65 is issued to find the address of the VSE/VSAM root phase, IDCAMS. When the SVC returns with the address of IDCAMS, a branch is made to IDCAMS, giving control to "live" VSE/VSAM 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 VSE/VSAM phases located in the CMSVSAM DCSS. The CMSVSAM DCSS is loaded when AMS opens the VSAM catalog for processing.

On return from VSE/VSAM 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 VSE User

When a VSAM function, such as an OPEN or CLOSE macro, is requested from a VSE program, CMS routes control through the CMSDOS DCSS to the CMSVSAM DCSS, thus giving control to VSE/VSAM phases. Figure 25 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 VSE 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 IJBLKMD is being requested. IJBLKMD is the VSE lookaside function that VSE/VSAM uses to gain information from the partition anchor tables. If this is the case, DMSDOS returns the address of the IJBLKMD that resides in the CMSBAM DCSS.

If VSAM has not been loaded, a DIAGNOSE 64 (LOADSYS) is issued to load the CMSVSAM DCSS.

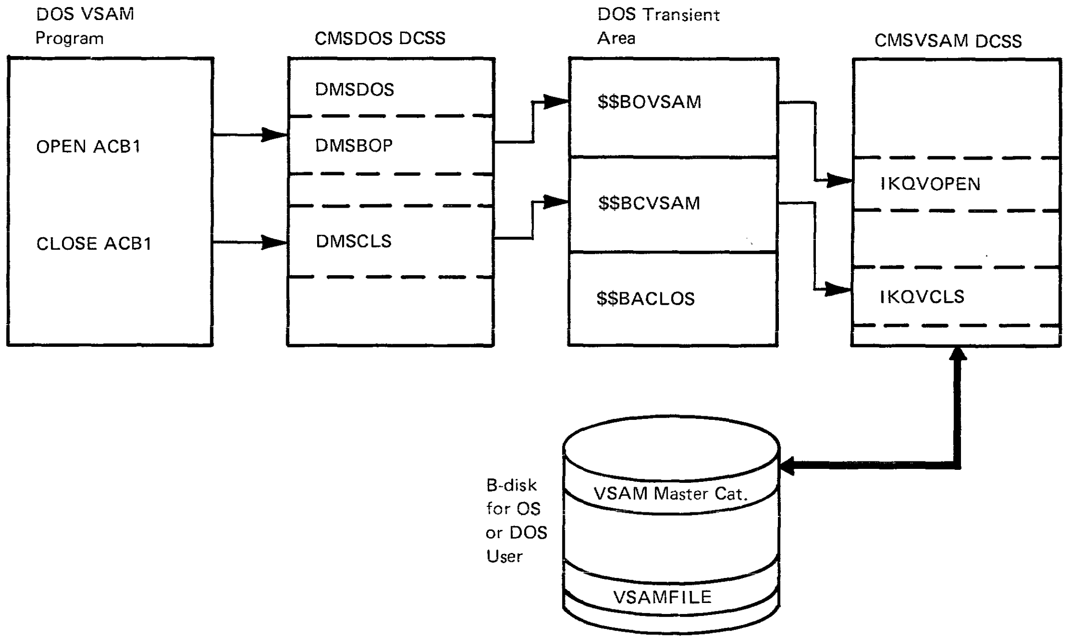


Figure 25. The Relationships in Storage between the User Program and the CMSDOS and CMSVSAM DCSSs

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 CMSVSAM DCSS. DMSBOP then initializes the transient work area and issues a VSE OPEN via SVC 2 to bring the VSAM OPEN $$BOVSAM transient into the VSE 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 VSE transient area. When the VSAM CLOSE routines complete processing, control returns to the user program, as in the case of OPEN.

Note: Since VSE does not support the 3380, CMS/DOS and CMS/VSAM cannot access a 3380 when minidisks are formatted as OS/DOS disks.

Executing a VSAM Function for an OS User

OS user requests for VSAM services are handled by VSE/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 VSE/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 26 shows the relationships in storage between the user program, the OS simulation and interface routines, and the CMSDOS and CMSVSAM DCSSs.

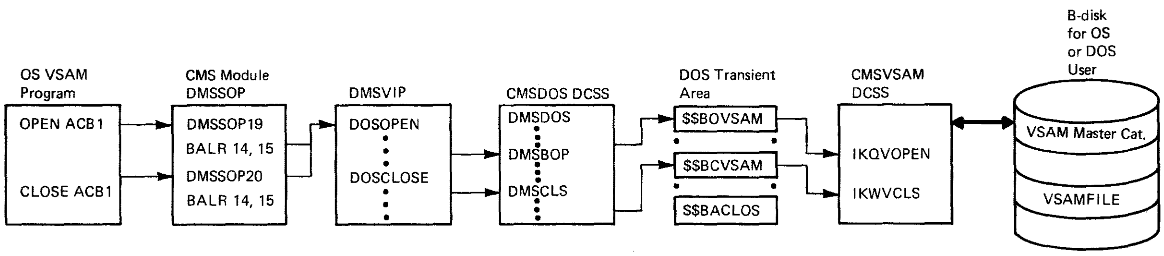


Figure 26. Relationship in Storage between the User Program, the OS Simulation and Interface Routines, and the CMSDOS and CMSVSAM DCSSs

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 CMSDOS 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 VSE/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 VSE/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 CMS/DOS 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 VSE/VSAM 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 VSE request code. The RPL or chain of RPLs is rearranged to VSE format (unless that has already been done).

If there is an ECB address in the OS RPL, a flag is set in the new VSE 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 VSE macro is then issued via a BALR to IKQVSM.

VSE 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 VSE/VSAM PROGRAMS

When an OS or VSE/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 VSE/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 27.

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).

4. Issues the STACK macro to define the terminal as the primary source of input.

	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
4	04	GETMAIN	DMSSMN	Conditionally acquires user free storage
5	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
A	10	GETMAIN/ FREEMAIN	DMSSMN	Manipulates free user storage
		GETPOOL	DMSSMN	Simulates an SVC10
B	11	TIME	DMSSVT	Gets the time of day
D	13	ABEND	DMSSAB	Terminates processing
E	14	SPIE	DMSSVT	Processes program interruptions
11	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

Figure 27. OS Functions that CMS Simulates (Part 1 of 2)

SVC Number	OS Macro Function	Simulation Routine	Comments
-	GET/PUT	DMSSQA	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
112	PGRLSE	DMSSVT	Release storage contents

Figure 27. OS Functions that CMS Simulates (Part 2 of 2)

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. 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.

GETPOOL: 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.

TIME-SVC 11: 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'0FF'. 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.

ABEND-SVC 13: 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.

SPIE-SVC 14: 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.

RESTORE-SVC 17: RESTORE is a NOP located in DMSSVT.

BLDL/FIND(Type D)-SVC 18: 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.

STOW-SVC 21: See STOW under description of BPAM routines in DMSSVT.

OPEN/OPENJ-SVC 19/22: 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.

CLOSE/TCLOSE-SVC 20/23: 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. DMSSOP simulates the CLOSE macro. CMS ignores the DISP option; all other options of CLOSE and TCLOSE are supported.

DEVTYPE-SVC 24: 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 cptions 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 pcinter 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.

PGRLSE-SVC 112: Located in DMSSVT, PGRLSE receives control when a PGRLSE macro instruction is issued. The routine checks the validity of the beginning and end addresses of the area to be freed, or forces the right values (AUSRAREA to the beginning, or FREELOWE to the end). Then the routine checks the length of the area to find out if at least 1 page (4K bytes) has to be released and issues a DIAGNOSE code X'10' instruction to CP. The return code will set to zero in register 15 if the PGRLSE operation is successful, or to four if only a portion of the area is released.

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 MOD 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 of 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 OS 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.

OSRUN COMMAND FLOW: The module DMSOSR gets control first when you invoke the OSRUN command. DMSOSR checks the command syntax. The PARM=parameter, if specified, is set up according to OS convention and a LINK (SVC 6) is issued for the member specified in the OSRUN command. DMSITS (the SVC FLIH) passes control to DMSSVT which in turn goes to DMSSLN for processing of the LINK SVC. DMSSLN passes control to DMSLOS. DMSLOS loads, relocates, and executes the member specified. When the member completes execution and returns control to DMSLOS, DMSLOS returns to DMSSLN for some cleanup; DMSSLN goes through the normal SVC return to DMSOSR. DMSOSR goes through its termination and returns to CMS.

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.

LKED COMMAND FLOW: The module DMSLKD gets control first when you invoke a CMS LKED command. DMSLKD generates the necessary FILEDEFs for execution of the OS linkage editor and calls the linkage editor (HEWLFROU). When the link-edit is complete, DMSLKD receives control to do some clean up prior to returning to CMS.

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 release 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 qualifiers 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, DSORG, 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 OS 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 cther errors, such as a return code 80, cause DMSLFS to exit immediately. A return code of 0 from DMSROS 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 cn 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 OS disk), and the number of cylinders in the disk. If the disk is a DOS disk, ROSACC sets a flag in the ADT. Informaticn 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 cut 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 OS 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 rcsstret.

- 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.

<u>DMSSEB</u> <u>MODULE</u>:

- 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.

<u>DMSSOP</u> <u>MODULE</u> -- 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.

<u>DMSSVT</u> <u>MODULE</u>:

- 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/C - 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 VSE Environment under CMS

CMS/DOS is a functional enhancement to CMS that provides VSE installations with the interactive capabilities of a VM/SP virtual machine. CMS/DOS operates as the background VSE partition; other VSE partitions are unnecessary, since the CMS/DOS virtual machine is a one-user machine.

CMS/DOS provides read access to real VSE data sets, but not write or update access. Real VSE 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, DOS/VS COBOL, and the DOS/VS RPG II compilers, the

FETCH routine, and the RSERV, SSERV, and ESERV commands. No read or write capability exists for the VSE 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 area.

INITIALIZING VSE AND PROCESSING VSE 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).
- Locates and loads the CMSBAM shared segment if available. This segment contains the following:
  - Simulated VSE OPEN/CLOSE and logic module routines for the VSE sequential access method
  - DTFSL support for the DOS PL/I and DOS/VS COBOL compilers
  - LBROPEN, LBRFIND, and LBRGET macro simulation as required by the VSE ESERV program
  - VSE lookaside function support as required by VSE/VSAM
- Obtains free storage and initializes the LOCK/UNLOCK resource control table.

- Sets the DOSMODE, DOSSVC and CMSBAM 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 VSE data areas, SYSCOM and BGCOM. NUCON also contains the address of the TCB.

The SYSCOM DSECT is the VSE system communications region. It consists mainly of address constants, including the addresses of the boundary box, 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 VSE data areas such as the LUB, PUB, NICL, FICL, PIB, and PIB2TAB.

The Task Control Block (TCB) contains the addresses of the PC and AB exit routines. The TCB also contains the addresses of the related PC and AB exit save areas.

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.

Three bits are set in DOSFLAGS in NUCON, DOSMODE, DOSSVC and CMSBAM. DOSMODE specifies that this virtual machine is running in the CMS/DOS operating environment. DOSSVC indicates whether OS or VSE SVCs are operative in the operating environment. CMSBAM indicates that various VSE functions are supported and available. If DOSSVC is set, VSE 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.

A check is made to ensure that the logical units SYSRDR or SYSIPT are not being assigned to a DOS formatted FB-512 DASD. This is not supported in the CMS/DOS environment because SVC 103 (SYSFIL support) is not available.

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.

DMSDAS--Dynamically Associated Programmer Logical Units with Physical Units

The function of DMSDAS is to assign a disk device with address X'cuu' to a programmer logical unit (SYS000 - SYS241).

The dynamic assign function supports assigning a DASD unit either permanently or temporarily, changing a DASD unit temporary assign to permanent, or unassigning a DASD. Temporary assigns are cleared either at end-of-job or when the program is canceled.

DMSDAS first searches the Active Disk Table (ADT) chain to ensure that the X'cuu' supplied is accessed. If the X'cuu' exists, DMSDAS ensures the device is a DASD unit. The programmer LUB table is then searched backwards to find the first available entry. A CMS PLIST is built using the found LUB entry to call DMSASN to actually do the assign.

DMSDAS updates the appropriate LUB entry directly when performing the unassign and change functions.

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 VSE 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 VSE SYSRES unit under VSE. 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 VSE data set name.

6. 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 VSE 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 logical 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.

OPEN/CLOSE processing in the CMS/DOS environment depends upon the DTF type:

- For DTFCP (disk), DTFDI (disk), and DTFSD DTF types, actual OPEN/CLOSE processing is performed by the simulated VSE SAM routines in the CMSBAM DCSS.

- For all other supported DTF types, OPEN/CLOSE processing is performed totally within the CMS/DOS modules mentioned above.

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 VSE 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 VSE OPEN function for nondisk DTFs. The basic function of DMSBOP for nondisk DTFs is the initialization of DTF tables (that is, setting fields in specified DTFs for use by the VSE LIOCS routines). For disk DTFs, DMSBOP services as an interface routine and passes control the the CMSBAM DCSS.

When a VSE 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 item 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 for nondisk files, 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.

4. For disk files, DMSBOP simulates the function of the VSE transient $$BOSFBL. DMSBOP sets up in the CMSBAM DCSS the input parameters and data areas required by the simulated VSE SAM routines. Control is then passed to the CMSBAM DCSS by placing the address of $IJJGTOP (the SAM OPEN/CLOSE phase) in the problem program save area PSW and exiting via SVC 11.

5. DTFDI and DTFCP are device-independent DTFs. Processing is as above depending upon the type of physical unit to which the DTFs are assigned.

6. If no disk DTFs are encountered, DMSBOP opens all files in the table and returns control to the problem program via SVC 11. If a disk DTF is encountered, DMSBOP exits as described above in step 4 for disk files.

7. If errors are encountered during DMSBOP processing, an error message is issued and return is made via SVC 6.

Closing Files Associated With DTFs

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.

Opening and Closing Files Associated with Disk DTFs

The OPEN and CLOSE functions for disk DTFs are performed by the simulated VSE SAM routines located in the CMSBAM DCSS.

These routines normally issue the LABEL macro to obtain DLBL/EXTENT information from the VSE label area, and issue the OVTOC, PVTOC, and CVTOC macros to obtain VTOC information. These macros require special handling in CMS/DOS. Processing is as follows:

1. DMSLAB (LABEL macro support) - CMS/DOS does not support the label information area in the same manner as VSE. CMS/DOS keeps similar

information in the DOSCB for the file. CMS/DOS intercepts invocations of the LABEL macro and passes control to DMSLAB. DMSLAB obtains the appropriate information from the DOSCB and builds the DLBL/EXTENT record. The DLBL/EXTENT record is then returned to the SAM routines in CMSBAM. Only the GETLBL and GETNXL functions of the LABEL macro are supported. All other functions result in an error return code to the SAM routines in CMSBAM.

2. DMSCVH (OVTOC, PVTOC, and CVTOC macro support) - In VSE these macros are normally handled by the Common VTOC Handler routines. These routines are simulated in CMSBAM and are used when accessing the VTOC on an OS or DOS formatted disk. However, when these macros are issued for a file on a CMS formatted disk, DMSCVH must simulate the appropriate function because CMS formatted disks do not contain a VTOC. VTOC functions simulated by DMSCVH are as follows:

   OVTOC - open VTOC
   PVTOC - read format 1 label by name
   PVTOC - read format 1 label by address
   PVTOC - write format 1 label in any slot
   PVTOC - write format 1 label by address
   PVTOC - check for file overlap
   PVTOC - scratch file
   CVTOC - close VTOC

   Any other requested VTOC functions is regarded as an error and the program is canceled via SVC 6.

3. When the SAM routines in CMSBAM complete processing, they exit via an SVC 2 to $$BOSVLT. The functions of this transient are simulated within CMS/DOS by the DMSVLT module. Obtained storage areas are returned and other clean-up functions are performed. DMSVLT exits in one of two different ways:

   - If there are no more DTFs to process, control is returned to the problem program via SVC 11.
   - If there are more DTFs to process, an SVC 2 is issued to the appropriate $$B transient. Then, DMSBOP or DMSCLS is eventually invoked to process the remaining DTFs.

CONTENTS OF THE CMSBAM DCSS

Several VSE functions are supported within the CMSBAM DCSS as simulated VSE phases. The simulated VSE phases and their functions are as follows:

$IJJGTOP - performs OPEN and CLOSE functions for all disk DTFs (DTFSD, DTFDI, and DTFCP).

$IJJHCVH - performs VTOC access functions for all disks in DOS format.

$IJBLBSL - performs I/O operations to the VSE source statement library for the VSE compilers and the ESERV utility program. The compilers invoke this phase via the DTFSL macro. ESERV invokes this phase indirectly via the LBRFIND and LBRGET macros.

DMSLBR - simulates the VSE internal macros LBROPEN, LBRFIND, and LBRGET to the extent required by the VSE ESERV utility program. $IJBLBSL is invoked to perform I/O operations to the VSE source statement library when appropriate.

$IJBLKMD - performs the VSE lookaside function as required by VSE/VSAM.

Eight VSE logic modules and two VSE SAM service routines are also simulated as VSE phases. The logic modules handle I/O macros (GET, PUT, POINT, etc.) for SAM files as issued by the user's program. The logic modules and the specific type of SAM file they are associated with are as follows:

$IJGXSDF - DTFSD fixed length record data files on DOS formatted FB-512 devices assigned to nonSYSFIL logical units.

$IJGXSDU - DTFSD undefined record data files on DOS formatted and CMS formatted disks assigned to nonSYSFIL logical units.

$IJGXSDV - DTFSD variable length record data files on DOS formatted FB-512 devices assigned to nonSYSFIL logical units.

$IJGXSDW - DTFSD work files on DOS formatted and CMS formatted disks assigned to nonSYSFIL logical units.

$IJGXSVI - DTFSD variable length record data files on CMS formatted FB-512 and DOS, or CMS formatted CKD devices assigned to nonSYSFIL logical units.

$IJGXSFI - DTFSD fixed length record data files on CMS formatted FB-512 and DOS, or CMS formatted CKD devices.

$IJGXCP - DTFCP files except for files on DOS formatted FB-512 devices assigned to SYSFIL logical units.

$IJGXDI - DTFDI files except for files on DOS formatted FB-512 devices assigned to SYSFIL logical units.

SYSFIL logical units are not supported for use with DOS formatted FB-512 devices in CMS/DOS. SYSFIL logical units refers collectively to logical units SYSRDR, SYSIPT, SYSLST, and SYSPCH.

The SAM service routines issue the actual I/O channel programs for SAM files. The functions they perform are as follows:

$IJGXSSR - issues I/O operations for DOS formatted FB-512 devices.

$IJGXSRI - issues I/O operations for all CMS formatted disks (FB-512 or CKD) and for DOS formatted CKD devices.

PROCESS CMS/DOS EXECUTION-RELATED CONTROL COMMANDS

The CMS/DOS FETCH and DOSLKED commands simulate the operation of the VSE fetch routines and the VSE 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.

DMSFET and DMSFCH--Bring a Phase into Storage for Execution

The VSE 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 VSE 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 CMS/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 VSE Linkage Editor: DMSDLK

CMS simulation of the VSE Linkage Editor function directly parallels the Release 1 implementation of that function. For detailed information on the logic of the function, see the publication DOS/VSE Linkage Editor Logic, Order No. SY33-8556.

The modules that comprise the VSE 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 VSE counterparts and have been named using the same naming conventions as for the VSE CSECTs. For example, the IJBESD CSECT in VSE is paralleled by the CMS DMSDLK subroutine DLKESD.

A brief description 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 VSE code is the use of OS macros, in some instances, rather than VSE 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 VSE counterparts.

SIMULATE VSE 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.

Figure 28 shows the VSE SVCs and their support in CMS/DOS simulation routines, the name of the macro that invokes a given SVC code, and a brief statement describing how the SVC function is performed.

Function/Macro	SVC No. Dec	Hex	Support
EXCP	0	0	Used to read from CMS or DOS/OS formatted disk. The CCW's are converted to appropriate CMS I/O requests (ex., RDBUF/WRBUF, CARDRD/CARDPH, etc.). The CCB or IORB is posted according to the CMS return information. DMSDOS will call CMSXCP routine to perform the I/O operation. If a non-zero return code is returned from DMSXCP, a cancel is done. I/O requests to DOS disks are handled using CP DIAGNOSE instructions.
FETCH	1	1	Used to bring a problem program phase into user storage, and to start execution of the phase if the phase was found. If the user did specify a directory list, a call to DMSFCH is made. Otherwise, DMSDOS will build a directory list using the specified phase name. Once the directory list is prepared, a call to DMSFCH is made. Upon return from DMSFCH, if the phase was found, the entry point address of the phase is saved in the 'SVC' save area oldpsw so that upon return to CMS, DMSITS will then give control to the phase just loaded. If upon return from DMSFCH there were any errors, a cancel is done. If the phase was not found, a message is issued and a cancel is done.
FETCH	2	2	Used to bring a $$B-transient phase into the CMS transient area (or if the phase is in the CMSDOS segment, not to load it), and start execution of the phase if the phase was found. A search is made through the loaded segment(s) in an attempt to locate the specified transient. If the phase is found in one of the segments, a call to DMSFCH is not needed. If the phase was not found, a call to DMSFCH is made in a similar way as in SVC 1 above. Once the transient entry point is obtained (from storage or loaded), the address is saved in the SVC save area (as above SVC 1) so that DMSITS gives immediate control to the phase wanted. Errors or not found conditions are handled as above in SVC 1.
FORCE DEQUEUE	3	3	Not supported, see note 2.

Figure 28. SVC Support Routines and Their Operation (Part 1 of 11)

Function/ Macro	SVC No. Dec	Hex	Support
LOAD	4	4	Used to bring a problem program phase into user storage, and return the caller the entry point address of the phase just loaded. Loading of the requested phase is done exactly as FETCH (SVC 1) calling DMSFCH. Any errors returned from DMSFCH are processed exactly as in fetch. A difference between FETCH (SVC 1) and LOAD (SVC 4), is that upon return from DMSFCH, assuming there are no errors, the user's registers 0 and 1 are updated to contain the address of the directory list (for the user to test if the phase was found), and the entry point address of the phase, respectively. If IJBSIA is being loaded, the address of DMSLAB is returned. If $IJJHCVA (Common VTOC handler) is being loaded, the address of DMSCVH is returned.
MVCOM	5	5	Provides the user with a means of altering positions 12 through 23 of the partition communications region (BGCOM). Before moving the specified information, a test is made to ensure that the range (user's start address, plus length of field to move) will not exceed the allowed range. Once the specified range is found to be within the allowed limits, the user's specified information is moved to the partition communications region.
CANCEL	6	6	Cancels a VSE session either by a VSE program request, or by request from any of the CMS routines handling CMS/DOS. Cancel will issue the message 'JOB CANCELLED DUE TO PROGRAM REQUEST'. A test will be made to see if the value of register 15 upon entry to cancel is below 256. If below, the value in register 15 will be the return code to CMS. If equal or greater, a special return code of 101 will be used to denote that the cancel was issued from a user program (return code of 101 is not used for CMS error messages). Processing then continues using the 'EOJ' code.
WAIT	7	7	Used to wait on a CCB, IORB, ECB, or TECB (note that CMS/DOS does not support ECBs or TECBs). CCBs are always posted by the DMSXCP routine before returning to the caller. The WAIT support under CMS/DOS will effectively be a branch to the CMS/DOS POST routine.

Figure 28. SVC Support Routines and Their Operation (Part 2 of 11)

Function/Macro	SVC No. Dec	Hex	Support
CONTROL	8	8	Temporarily return control from a $$B-transient to the problem program. If a $$B-transient has to temporarily give control to the problem program, the $$B-transient will issue an SVC 8 passing in register 0 the address of the problem program gaining control. SVC 8 routine will store this address in the SVC work area oldpsw, and return back to CMS SVC handler (DMSITS).
LBRET	9	9	Return to a $$B-transient after an SVC 8 was issued to give control to the problem program. The address saved before (SVC 8 above) is stored in the SVC work area oldpsw, so that when DMSDOS returns to the CMS SVC handler, control is given to the $$B-transient that issued the SVC 8.
SET TIMER	10	A	No operation, successful return code of 0 is given in register 15. See note 1.
TRANS. RETURN	11	B	Return from a $$B-transient to the calling problem program. The address saved when the initial SVC 2 (fetch a $$B-transient) was issued, is stored in the CMS's SVC work area oldpsw. Now, when DMSDOS returns to the CMS's SVC handler, control will return to the problem program that issued the SVC 2 calling the $$B-transient.
JOB CTL. 'AND'	12	C	Resets flags to 0 in the linkage control byte in BGCOM (communication region). If register 1 equals 0, SVC 12 has another meaning. Bit 5 of JCSW4 (COMREG byte 59) is turned off. If register 1 contains a nonzero value, the function depends on bit 8 of this register. If bit 8 is 0, this SVC supplies supervisory support to reset flags in the linkage control byte (displacement 57 in BGCOM - communication region). The user has provided the address of a mask (1 byte) in register 1. An 'AND' operation of the mask with the linkage control byte is performed. If bit 8 of register 1 is one, this SVC supplies the supervisory support to reset flags in a specified byte of BGCOM (communication region). The user has provided a displacement in byte 2 and a mask in byte 3 of register 1. An 'AND' operation of the mask byte with the specified displacement in the partition communication region is performed.
JC FLAGS	13	D	Not supported. See note 2.

Figure 28. SVC Support Routines and Their Operation (Part 3 of 11)

Function/ Macro	SVC No. Dec	Hex	Support
EOJ	14	E	Normally terminates execution of a problem program. The last SVC save work area is unstacked. Cleanup is done by: 1. Clearing the CMS DOSLIB CMSCB 2. Resetting the JOBNAME in BGCOM 3. Unassigning all temporary device assignments The latest return code is loaded into register 15, and control returns to DMSITS (CMSRET).
SYSIO	15	F	Not supported. See note 2.
PC STXIT	16	10	Establish or terminate linkage to a user's program check routine. Locate the appropriate PC option table entry. If the contents of register 0 is zero (terminate linkage), determine if PC routine is active. If the PC routine address in PC option table is negative, terminate linkage by storing zero in routine address field of PC option table. If the routine is not active presently, store zeros in PC routine address field and savearea address field in PC option table. If register 0 is not zero, the address of the PC routine and the savearea address is passed to the STXIT macro. If a STXIT PC routine is active, the complement of the new routine address is placed in the PC option table. If no STXIT PC routine is active, the new PC routine address and savearea address are stored in the PC option table.
PC EXIT	17	11	Used to provide supervisory support for the EXIT macro. SVC 17 provides a return from the user's PC routine to the next sequential instruction in the program that was interrupted due to a program check. Locates the appropriate PC option table entry and restores user's registers and PSW. Stores the address of the PC routine in the PC option table returns to the next sequential instruction in the program that was interrupted.
IT STXIT	18	12	No operation, successful return code of 0 is given in register 15. See note 1.
IT EXIT	19	13	Not supported. See note 2.
OC STXIT	20	14	No operation, successful return code of 0 is given in register 15. See note 1.
OC EXIT	21	15	Not supported. See note 2.
SEIZE	22	16	No operation, successful return code of 0 is given in register 15. See note 1.

Figure 28. SVC Support Routines and Their Operation (Part 4 of 11)

Function/ Macro	SVC No. Dec	Hex	Support
LOAD HEADER	23	17	Not supported. See note 2.
SETIME	24	18	No operation, successful return code of 0 is given in R15. See note 1.
HALT I/O	25	19	Not supported. See note 2.
	26	1A	Validate address limits. The upper address must be specified in general register 2 and the lower address must be specified in general register 1. First the lower address must nct be negative. An error message DMSDOS005E is issued if it is. Second, the high address cannot be negative. If it is, the same error messages is issued. If the low or high address is greater than the end of partition address in BGCOM, the same error message is issued. Otherwise, contrcl returns to the caller.
TP HALT I/O	27	1B	Not supported. See note 2.
MR EXIT	28	1C	Not supported. See note 2.
WAITM	29	1D	Not supported. See note 2.
QWAIT	30	1E	Not supported. See note 2.
QPOST	31	1F	Not supported. See note 2.
	32	20	Reserved.
COMRG	33	21	Used to provide the caller with the address of the partition communications region. DMSDOS will provide the caller with the address of the partition communications region, in the user's register 1.
GETIME	34	22	Provides support for the GETIME macro. SVC 34 updates the date field in the communications region. Upon return, general register 1 contains the time of day in timer units (1/300 sec). The GMT operand is nct supported.
HOLD	35	23	No operation, successful return code of 0 is given in register 15. See note 1.
FREE	36	24	No operation. Successful return code of 0 is given in register 15. See note 1.

Figure 28. SVC Support Routines and Their Operation (Part 5 of 11)

Function/ Macro	SVC No. Dec	Hex	Support
AB STXIT	37	25	Establish or terminate linkage to a user's abnormal termination routine. Locate the appropriate AB option table entry. If register 0 is zero, terminate linkage; if AB routine is active (ab routine address in AB option table is negative), terminate linkage by storing zero in routine address field of AB option table. If routine is not active presently, store zeros in AB routine address field and savearea address field in AB option table. If register 0 is not zero, pass the routine and savearea addresses to the STXIT AB macro. The limits of the savearea are validated and cannot be less than 20000 or greater than partition end. If a STXIT AB routine is active, the complement of the new routine address is placed in the AB option table. If no STXIT AB routine is active, the new AB routine address and savearea address are stored in the AB option table.
ATTACH	38	26	Note supported. See note 2.
DETACH	39	27	Not supported. See note 2.
POST	40	28	Used to post an ECB, IORB, TECB or CCB. Byte 2, bit 0 of the specified control block will be turned 'on' by DMSDOS.
DEQ	41	29	No operation, successful return code of 0 is given in register 15. See note 1.
ENQ	42	2A	No operation, successful return code of 0 is given in register 15. See note 1.
	43	2B	Reserved.
UNIT CHECKS	44	2C	Not supported. See note 2.
EMULATOR INTERF.	45	2D	Not supported. See note 2.
OLTEP	46	2E	Not supported. See note 2.
WAITF	47	2F	Not supported. See note 2.
CRT TRANS	48	30	Not supported. See note 2.
CHANNEL PROG.	49	31	Not supported. See note 2.
LIOCS DIAG.	50	32	Issued by a logical IOCS routine when the LIOCS is called to perform an operation the LIOCS was not generated to perform. The error message 'unsupported function in a LIOCS routine' will be issued, and the session will then be terminated.

Figure 28. SVC Support Routines and Their Operation (Part 6 of 11)

Function/ Macro	SVC No. Dec	Hex	Support
RETURN HEADER	51	33	Not supported. See note 2.
TTIMER	52	34	No operation. Successful return code of 0 is given n register 15. See note 1. Register 0 is also cleared.
VTAM EXIT	53	35	Not supported. See note 2.
FREEREAL	54	36	Not supported. See note 2.
GETREAL	55	37	Not supported. See note 2.
POWER	56	38	Not supported. See note 2.
POWER	57	39	Not supported. See note 2.
SUPVR. INTERF.	58	3A	Not supported. See note 2.
EOJ INTERF.	59	3B	Not supported. See note 2.
GETADR	60	3C	Not supported. See note 2.
GETVIS	61	3D	Used by VSAM to obtain free storage for scratch use or for obtaining an area into which a relocatable VSAM program may be loaded. A free storage subroutine similar to that in the "DMSSMN" routine is called to obtain the needed space (from the user area). If successful, the address is returned in register 1, and register 15 is cleared. If he request cannot be satisfied, a return code of 12 is passed back in register 15. The 'PAGE', 'POOL', and 'SVA' GETVIS options are ignored.
FREEVIS	62	3E	Used to return the free storage obtained via an earlier GETVIS call. The free storage subroutine similar to that in the "DMSSMN" routine is called to return the area designated by register 1. All complete pages (4K bytes) associated with the returned storage are released by issuing a DIAGNOSE code X'10' instruction to CP.
USE	63	3F	The USE/RELEASE function has been replaced by SVC 110 (LOCK/UNLOCK) for serially controlling system resources. All SVC 63 and 64 requests are mapped into SVC 110 requests respectively. Return code previously associated with USE/RELEASE under CMS/DOS are maintained.
RELEASE	64	40	Reference SVC 63.

Figure 28. SVC Support Routines and Their Operation (Part 7 of 11)

Function/ Macro	SVC No. Dec	SVC No. Hex	Support
CDLOAD	65	41	Used to load a relocatable VSAM phase into storage unless the program has already been loaded. If an anchor table is available, it is searched for the given phase; if found, its load point, entry point, and length are returned in the caller's register 0, 1, and 14 respectively, with register 15 set to 0. If not, DMSFCH is called to find the given phase; if found in a discontinuous shared segment, register 0, 1, and 14 are loaded as above and return made. If the phase was found but is not loaded, storage is obtained (if available) from the GETVIS SVC; DMSFCH is called again to load the program into the storage area just obtained. An anchor table is built in the user area (unless one already exists), the appropriate entries made, and registers 0, 1, and 14 loaded as above, with return to caller. If the program cannot be found, or if storage is unavailable for either loading the program or for building the anchor table, an error code 22 (X'16') is returned to the caller in register 15.
RUNMODE	66	42	Used by a problem program to find out if the program is running in real or virtual mode. The caller's register 0 will be zeroed to indicate that the program is running in virtual mode.
PFIX	67	43	No operation, successful return code of 0 is given in register 15. See note 1.
PFREE	68	44	No operation. Successful return code of 0 is given in register 15. See note 1.
REALAD	69	45	Not supported. See note 2.
VIRTAD	70	46	Not supported. See note 2.
SETPFA	71	47	No operation. Successful return code of 0 is given in register 15. See note 1.
GETCBUF/ FREECBUF	72	48	Not supported. See note 2.
SETAPP	73	49	Not supported. See note 2.
PAGE FIX	74	4A	Not supported. See note 2.

Figure 28. SVC Support Routines and Their Operation (Part 8 of 11)

Function/ Macro	SVC No. Dec	Hex	Support
SECTVAL	75	4B	Used by VSAM I/O routines (ex., IKQIOA) to obtain a sector number for a 3330, 3330-11, 3340, or 3350 device. The appropriate sector value is calculated from the input data supplied by the user's register 0 and 1. If the calculation is successful, the sector number (from 0 to 127) is returned in register 0. If any errors were detected, the no-op set-sector value of 255 (X'FF') is returned.
SYSREC	76	4C	Not supported. See note 2.
TRANSCCW	77	4D	Not supported. See note 2.
CHAP	78	4E	Not supported. See note 2.
SYNCH	79	4F	Not supported. See note 2.
SETT	80	50	Not supported. See note 2.
TESTT	81	51	Not supported. See note 2.
LINKAGE	82	52	Not supported. See note 2.
ALLOCATE	83	53	Not supported. See note 2.
SET LIMIT	84	54	Not supported. See note 2.
RELPAGE	85	55	Provides support for the RELPAG macro. At entry register 1 points to a list of 8-byte storage description areas. Each entry contains the beginning address and the length 1 of an area to be released. A non zero byte following an entry indicates the end of the list. An area is released only if it contains at least a full CP page (4K bytes). Pages are released when the virtual machine calls CP via DIAGNOSE code X'10'. On return register 15 holds the return code as follows: register 15 = 0 all areas have been released register 15 = 2 one or more negative area lengths were specified. register 15 = 4 one or more pages to be released were outside the user storage area. register 15 = 16 at least one entry contains a beginning address outside the user storage area.
FCEPGOUT	86	56	No operation. Successful return code of 0 is given in register 15. See note 1.
PAGEIN	87	57	No operation. Successful return code of 0 is given in register 15. See note 1.
TPIN	88	58	Not supported. See note 2.

Figure 28. SVC Support Routines and Their Operation (Part 9 of 11)

Function/ Macro	SVC No. Dec	Hex	Support
TPOUT	89	59	Not supported. See note 2.
PUTACCT	90	5A	Not supported. See note 2.
POWER	91	5B	Not supported. See note 2.
XECBTAB	92	5C	Not supported. See note 2.
XPOST	93	5D	Not supported. See note 2.
XWAIT	94	5E	Not supported. See note 2.
AB EXIT	95	5F	Exit from abnormal task termination routine and continue the task. The linkage to either the PC or AB routine is reestablished, and the cancel condition is reset by clearing the ABEND indication in the partition PIB extension. Control is returned to the instruction following the exit AB macro.
TT EXIT	96	60	Not supported. See note 2.
TT STXIT	97	61	Not supported. See note 2.
EXTRACT	98	62	Support for EXTRACT macro of VSE. The caller requests PUB information, CPUID or, storage boundary information. Register 1 on entry points to a parm list. Output is placed in an area provided by caller.
GETVCE	99	63	Caller requests device information about a specific DASD. Information is returned in an output area pointed to from the parmlist. Register 1 contains a pointer to the parmlist on entry.
	100	64	Reserved.
MODVCE	101	65	No operation. Successful return code of 0 is given in register 15. See note 1.
	102	66	Reserved.
SYSFIL	103	67	Not supported. See note 2.
EXTENT	104	68	No operation. Successful return code of 0 is given in register 15. See note 1.
SUBSID	105	69	SUBSID.. the 'INQUIRY' function is supported for the supervisor subsystem. Information returned is described by the SUPSSID control block. The SUBSID 'NOTIFY' and 'REMOVE' functions are not supported.
LINKAGE	106	6A	Not supported. See note 2.

Figure 28. SVC Support Routines and Their Operation (Part 10 of 11)

Function/ Macro	SVC No. Dec	Hex	Support
TASK INTERF.	107	6B	Provides macro interface support for system information retrieval. The parameters supported are: GETFLD: field=ppsavar - returns problem program save area address. =savar - returns current save area address. =aclose - return in register 1, 0 if in process, 1 if not. MODFLD: =vsamopen - set bix X'08' in tcb tcbflags byte. =aclose - set bit X'10' in tcb tcbflags byte. All other GETFLD/MODFLD requests are treated as a NOP and a return code of 0 is placed in register 15. All other SVC107 macro calls are unsupported. The error message DMSDOS121S will be issued and the program is canceled. See note 2.
DATA SECURE	108	6C	Not supported. See note 2.
PAGESTAT	109	6D	Not supported. See note 2.
LOCK/ UNLOCK	110	6E	Used by VSAM to control access to resources. Access is maintained in either a 'shared' or 'exclusive' control environment. When DOS is SET ON, counters are maintained as well as the type of control for each resource in a table (LOCKTAB) built in free storage. All entries not unlocked by the program are cleared at both normal and abnormal end-of-job. All requests for resource control are passed to SVC 110 through the DTL macro (Define the Lock). SVC 63 requests are mapped into a dummy DTL and processed by SVC 110.

Notes:

1. No operation:

   In each case, register 15 is cleared to simulate successful operation, and all other registers are returned unchanged, unless otherwise noted.

2. Not supported:

   For unsupported SVCs, an error message will be given, and the SVC will be treated as a "cancel."

Figure 28. SVC Support Routines and Their Operation (Part 11 of 11)

PROCESS CMS/DOS SERVICE COMMANDS

DMSSRV--Copies books from a system or private source statement library to a specified output device.

DMSPRV--Copies VSE procedures from a VSE 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 VSE 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.

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 encountered 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/SP error recording records and records passed via the SVC 76 by virtual machines are accumulated in chronological order on the VM/SP error recording cylinders. The following modules are used by CMS CPEREP to edit and print error records compiled by VM/SP as well as SYS1.IOGREC 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/SP OLTSEP and Error Recording Guide_ and the _VM/SP Service Routines Program Logic_ with appropriate referrals to _OS/VS Environmental Recording, Editing, and Printing (EREP) Program_.

EXEC 2 PROCESSING

Two modules process EXEC 2 statements: DMSEXI and DMSEXE.

DMSEXI is an interface routine between CMS and either the CMS EXEC interpreter or the EXEC 2 interpreter.

DMSEXE is the EXEC 2 interpreter.

A description of each module's method of operation follows.

MODULE NAME: DMSEXI

CALLED BY: DMSEXC for all EXEC functions

CALLS TO OTHER ROUTINES:

DMSBRD - 'RDBUF' file system function
DMSEXT - CMS EXEC processor
DMSEXE - EXEC 2 processor
DMSFRE - Get and return free storage

EXTERNAL REFERENCES:

NUCON, IO

METHOD OF OPERATION:

DMSEXI is an interface routine between CMS and the two EXEC interpreters.

DMSEXI allows coexistence with the CMS EXEC interpreter, by routing calls to either the EXEC 2 interpreter, or the CMS EXEC interpreter, according to the following rules:

1. The caller provides an extended-form PLIST, including a file block. DMSEXI directs the call to the EXEC 2 interpreter.

2. The EXEC file specified exists, has a valid format, and contains the word '&TRACE' within the first 255 bytes of line 1. DMSEXI directs the calls to the EXEC 2 interpreter, after generating a file block and copying or building an extended PLIST.

3. DMSEXI directs all other cases to the CMS EXEC interpreter, with the original PLIST pointer.

There are two cases where CMSEXI must build an untokenized command string to pass to DMSEXE:

1. If CMS command mode originates the call, DMSEXI copies the command string to a buffer, and delimits the first word according to CMS rules.

2. If only a tokenized PLIST is available, DMSEXI builds a command string by concatenating the CMS tokens, separating each by one blank, with no leading or trailing blanks.

DMSEXI releases any storage obtained before it called CMSEXE, then returns to the main caller with the return code from DMSEXE in register 15.

The format of the extended-form PLIST is:

```
PLIST      DS    0F                              (alignment)
           DC    A(command-verb)
           DC    A(parm-string)
           DC    A(byte-following-parm-string)
           DC    A(0) or A(file-block)           (the file to be executed)
```

The command-verb and the parm-string form a contiguous area:

```
          COMMAND   DC   C'command-verb'
                    DC   C'parm-string'
```

Trailing blanks are allowed after the command-verb.

The format of the file block is:

```
FILE       DS    0F                              (alignment)
           DC    CL8'filename'                   (or blank)
           DC    CL8'filetype'                   (or blank)
           DC    CL2'filemode'                   (ex. A1, or blank)
```

If the filename contains blanks, CMSEXI will use the first word in the argument list (&0) as the filename.

If the filetype contains blanks, DMSEXI will use a filetype of EXEC.

If the filemode is blank, DMSEXI will use the first file with the specified filename and filetype, found according to the file system search order.

The format of the file block extension is:

```
           DC    XL2(0000) or XL2(0002)    (number of words in extension)
           DC    AL4(PGMFILE)              (address of the in-storage
                                            EXEC 2 descriptor)
           DC    AL4(PGMEND-PGMFILE)       (number of bytes in descriptor)
```

MODULE NAME: DMSEXE

CALLED BY: DMSEXI to interpret EXEC 2 statements.

CALLS TO OTHER ROUTINES:

```
DMSERR - Write all error messages
DMSSCN - Tokenize strings
DMSPNT - 'POINT' file system function
DMSSTT - 'STATE' file system function
DMSBRD - 'RDBUF' file system function
DMSFNS - 'FINIS' file system function
DMSFRE - Get and return free storage
WAITRD - Read from the terminal
TYPLIN - Type on the terminal
ATTN   - Stack lines in console stack
```

EXTERNAL REFERENCES:

NUCON, FST, FVS, ADT

METHOD OF OPERATION:

Overview

DMSEXE reads lines from disk files, or accepts lines previously prepared by the caller and stored in main memory.

If the lines are EXEC 2 statements, DMSEXE interprets the statements. If the lines contain commands, DMSEXE passes the commands to CMS command mode or a subcommand environment.

Execution continues until a statement or command explicitly terminates it, or DMSEXE finds a statement error.

DMSEXE LOGIC DESCRIPTION:

Pseudo Code

Pseudo code is used to describe the logic of portions of DMSEXE. This Pseudo code has the following general statements:

1. DO

```
     statement
     statement
      ...
     statement
   END
```

"Statement" is either:

1. A description of an action to be done, or

2. Another pseudo code statement:

```
DO...END,
IF...THEN...ELSE,
GOTO, or
CALL
```

2. If condition THEN statement ELSE statement.

   "Condition" is a hyphenated sequence of words describing the conditions for which the statement after "THEN" is executed.

   ```
   Example:  IF initial-flag-is-set
             THEN perform initialization
             ELSE indicate error condition
   ```

3. GOTO label

   Transfer control to the label specified. A label is followed by a colon and precedes a statement, or is on a line by itself.

   Example:
```
GOTO George
     ...
George: ...
```

4. CALL name

   CALLs the named subroutine.

DMSEXE General Logic Flow

After initialization, DMSEXE loops continually, reading lines that may contain EXEC 2 statements or commands. The logic follows:

```
Initialization
DO forever
   Loop initialization
   IF executing &loop
      THEN DO
               Test condition
               IF condition
                  THEN set for top of loop
                  ELSE set for exit from &loop
             END
   CALL READSUB   (read next line)
   IF eof
      THEN IF executing-&loop
                THEN error condition
                ELSE exit
   CALL EXECUTE   (execute line)
END
```

READSUB/READLAB

READSUB is the DMSEXE subroutine that reads the next line. READLAB, a secondary entry point to READSUB, reads the next line when scanning forward for labels.

READSUB reads a line from:

1. The console - if the console count is non-zero,
2. The cache - if there is one, and the needed line is there,
3. 'BUF' - if the needed line is there, or
4. The file - if none of the above conditions are true.

If the line is read from the file, and there is a cache, then the line is read into the cache.

READLAB reads a line from;

1. The cache - if there is one, and the line is there,
2. 'BUF' - if the line is there, or

3. The file - if none of the above conditions are true.

If the line is read from the file, and there is a cache, then the line is read into the cache.

In all cases:

1. A blank and a zero byte are placed at the end of a line,

2. The file read may be either an in-storage file, or a file accessed by calls to file system routines.

Line Execution

DMSEXE executes lines according to the following logic:

```
EXECUTE:  IF comment THEN exit

          IF tracing THEN trace the line

          IF blank-line THEN exit

          IF assignment
                THEN DO
                       CALL ASSIGN    (perform assignment)
                       Exit
                     END
          IF command
                THEN DO
                       Pass command to CMS command mode or
                            subcommand environment
                       Exit
                     END
          (Line must be a control-statement:)
          Look up control-statement word
          IF found
                THEN DO
                       GOTO control-statement routine:
                             ex. ARGS
                                 BEGPRINT
                                 BEGSTACK
                                 BUFFER
                                   ...
                       Exit
                     END
                ELSE error (invalid statement)
END EXECUTE
```

Assignment Processing

DMSEXE processes assignment statements according to the following logic:

```
ASSIGN:  CALL SUBS  (Substitute value of EXEC variable into
                     characters 2 through N of target)
         Point to first word after equal sign
         Call GETNEXT
         IF none THEN set null value and exit
         Call GETNEXT
         IF none THEN set value obtained above and exit
```

```
   Top-of-loop:
            IF last-word-is-not-an-operation
                 THEN error
            Call GETNEXT
            IF none THEN error
            Call GETNEXT
            IF none
                 THEN DO
                         Do calculation
                         Set value
                         Exit
                      END
            IF function-reference
                 THEN DO
                         IF not 'of' THEN error
                         IF system-function
                            THEN Call appropriate routine
                                 to evaluate function
                            ELSE invoke user function
                         Exit
                      END
            Do calculation
   GOTO Top-cf-loop
END ASSIGN

GETNEXT:  Get next word
          IF found
               THEN DO
                       Call SUBS
                       IF null THEN GOTO GETNEXT
                    END
END GETNFXT

SUBS:  Set pointer to end of word plus one

SUBSLP:
      Decrement pointer
      IF at-front-of-word THEN exit
      IF not '&' THEN GOTO SUBSLP
      Calculate hash using last character of name and length
      Scan appropriate variable lookaside chain
      IF found
           THEN DO
                   IF not-at-front-of-chain THEN put at front
                   IF predefined-variable
                        THEN DO
                                CALL predefined variable routine and
                                     substitute value
                                IF at-frcnt-of-word THEN exit
                                GOTO SUBSLP
                             END
                   Substitute value
                   IF at-end-of-word THEN exit
                   GOTO SUBSLP
                END
           ELSE DO
                   IF predefined-name
                        THEN DO
                                Build variable blocks
                                Point blcck to processing routine
                                CALL routine and substitute value
                             END
                        ELSE DO
                                Build variable block for null value
                                Substitute null
```

```
                        END
                   IF at-front-of-word THEN exit
                   GOTO SUBSLP
               END
          END SUBS
```

CMS Directories

This section contains the following information:

- Module Entry Point Directory
- Module-to-Label Cross Reference
- Label-to-Module Crcss Reference

Module Entry Point Directory

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 VSE/VSAM 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.

Module Name	Entry Points	Function
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.
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. For nondisk files, the OPEN function is performed in its entirety by DMSBOP. For disk files, the SAM OPEN/CLOSE routines in CMSBAM are invoked. 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. DMSBRD, itself, reads items from 800-byte formatted disks, or calls DMSERD at the DMSERDBF entry point to read items from 1K-, 2K-, or 4K-byte formatted disks.
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 execution 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. DMSBWR, itself, writes to 800-byte formatted disks, or calls DMSERD at the DMSEWRBF entry point to write items to 1K-, 2K-, 4K-byte formatted disks.

Module Name	Entry Points	Function
DMSCAT	DMSCAT	Stacks a line of console input that DMSCRD reads later when it is called.
	DMSCATMK	MAKEBUF command.
	DMSCATNB	SENTRIES command.
DMSCIO	DMSCIOR	Reads one card record.
	DMSCIOP	Punches one card record.
	DMSCIOSI	Punch caller's buffer.
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.
	DMSCITDK	DROPBUF command.
DMSCLS	DMSCLS	Closes CMS/DOS files associated with the following DTF (Define The File) tables: DMTCN, DTFCD, DTFPR, DTFMT, DTFDI, DTFCP, and DTFSD. For nondisk files, the CLOSE function is performed in its entirety by CMSCLS. For disk files, the VSE OPEN/CLOSE routines in CMSBAM are invoked.
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.
DMSCVH	DMSCVH	Simulates VTOC functions for CMS formatted disks in the CMS/DOS environment.
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.
DMSDAS	DMSDAS	Simulates the VSE ASSIGN macro.
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 CMS/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 previously defined disk file descriptions.

Module Name	Entry Points	Function
DMSDLK	DMSDLK	Interface for the CMS/DOS DOSLKED 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.
DMSDMP	DMSDMP	Simulates the VSE $$BDUMP and $$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).
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 for the CMS Editor.
DMSERD	DMSERDBF	Reads one or more items from a specified 1K-, 2K-, or 4K-byte formatted disk.
	DMSEWRBF	Writes one or more items from a specified 1K-, 2K-, or 4K-byte formatted disk.
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.
DMSETR	DMSETR	Provides SVC 98 EXTRACT macro support. Called by DMSDOS.
DMSEXC	DMSEXC	Bootstrap loader for disk version of EXEC.
DMSEXE	DMSEXE	Processes an EXEC 2 file.
DMSEXI	DMSEXI	Determine whether to call CMS EXEC or EXEC 2 processor (DMSEXT or DMSEXE).
DMSEXT	DMSEXT	Processes a CMS EXEC file.

Module Name	Entry Points	Function
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. Reads and writes one track at a time.
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.

Module Name	Entry Points	Function
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.
DMSHLB	DMSHLB	Processes and builds output for .BX HELP format word.
DMSHLD	DMSHLD	HELP facility communication module, loaded into free storage by DMSHLI.
DMSHLE	DMSHLE	Builds messages to be written at virtual control by DMSHLS.
DMSHLI	DMSHLI	Contains HELP facility initialization routines.
DMSHLP	DMSHLP	HELP facility module for processing HELP description file input.
DMSHLS	DMSHLS	Contains HELP facility I/O routines.
	SWRTPREP	Determines virtual terminal characteristics and acquires buffer storage.
	SWRTIO	Performs normal virtual terminal I/O.
	SWRTMSG	Performs I/O to virtual terminal for error messages.
	GOPEN	Performs OPEN function for HELP description file.
	GREAD	Routine to read HELP file input.
	GCLOSE	Routine to perform file closing functions on exit.
DMSIFC	DMSIFC	Scans and passes all non-special parameters to the IFCEREP1 module, initializing values to edit and print records from VM/SP's error recording cylinders.
	DMSIFC76	Immediately reflects SVC76 back to the calling routine.
	DMSIFC18	BLDL handler for IFCEREP1.
	DMSIFC0	EXCP handler for IFCEREP1.
DMSIMA	DMSIMAMD	Implements the IMAGEMOD command. This command is used to modify specific members of a 3800 named system. With this command, you can dynamically delete, add, replace, and generate members for a named system.
DMSINA	DMSINA	Handles either user-defined synonyms or abbreviations or system-defined synonyms for command names.
DMSIND	DMSINDEX	Index of CMS listings in the microfiche deck.
DMSINI	DMSINIR	Reads a nucleus into main storage.
	DMSINIW	Writes a nucleus onto a DASD unit.
DMSINM	DMSINM (GETCLK) (CMSTIMER)	Obtains the time from the CP timer.

Module Name	Entry Points	Function
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, 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.
	CMSITSSB	SUBCOM handling.
DMSLAB	DMSLAB	Simulates the VSE LABEL macro.
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, filetype, 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.
DMSLBD	DMSLBD	Allows the user to specify tape label information that will be used by a program at execution time (the parameters are similar to those of the DOS TLBL statement or the tape options of the OS data definition statement). LABELDEF can also be used to modify, delete, and list previously described label descriptions.
DMSLBM	DMSLBM	Generates a macro library, adds macros to an existing library, and lists the dictionary of an existing macro library.
DMSLBR	DMSLBR	Simulates the VSE LBROPEN, LBRFIND, and LBRGET macros as required by the VSE ESERV utility program.
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.
DMSLCK	DMSLCK	SVC 110 LOCK/UNLOCK macro support. Called by DMSDOS.
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.

Module Name	Entry Points	Function
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.
DMSLOA	DMSLOA	Processes the LOAD and INCLUDE commands to invoke the relocating loader.
DMSLOS	DMSLOS	Provides load and relocate support for OS load modules and CMS LOADLIB modules.
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/SP system volume.

Module Name	Entry Points	Function
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.
	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.
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 VSE options in the System Communications Region as specified by the OPTION command.
DMSOR1	DMSOR1	Relocates all DTF (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.)
DMSOSR	DMSOSR	Allows user to invoke a program from a CMS LOADLIB or an OS module library.
DMSOVR	DMSOVR	Analyzes the SVCTRACE command parameter list and loads the DMSOVS tracing routine.
DMSOVS	DMSOVS	Provides trace information requested by the SVCTRACE command.

Module Name	Entry Points	Function
DMSPIO	DMSPIO	Prints one line.
	DMSPIOCC	Puts CCWs to select translate table (for virtual 3800) and to print the data, plus the data itself, in 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.
DMSPRE	DMSPREEP	Combine, link, and relocate multiple text (object) files into a single text file.
DMSPRT	DMSPRT	Prints CMS files.
DMSPRV	DMSPRV	Copies procedures from the VSE 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.
DMSRNM	DMSRNM	Processes the RENAME command. Changes the fileid of the specified file.
DMSROS	DMSROS ROSACC	Accesses OS disks.
	DMSPOS+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.

Module Name	Entry Points	Function
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.
DMSSFF	DMSSFF	Provides overlay support for OS load modules.
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.
DMSSPR	DMSSPR	Processes the SETPRT command. This command sets up a virtual 3800 printer spool file for a CMS user. With the SETPRT command, a user can select the character arrangement tables, copy modification modules, FCB, and forms overlay frame for printing files with a virtual 3800.
DMSSQS	DMSSQS	Analyzes record formats and sets up the buffers for GET, PUT, and PUTX requests.
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.

Module Name	Entry Points	Function
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.
DMSSVU	DMSSVU	Builds a keys file when a data file using keys is opened and saves the keys in the data file when it is closed.
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.
DMSTLA	DMSTLA	Invokes the tape label processor, DMSTLB.
DMSTLB	DMSTLB	Processes IBM standard tape labels for OS simulation, CMS/DOS, CMS commands, and TAPESL macro. Also provides linkage to nonstandard user label routines for OS simulation and CMS commands.
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, display or write VOL1 labels, scan, skip, rewind, run, FSF, FSR, BSF, BSR, ERG, and WTM.
DMSTPF	DMSTPF	Tapeload function.
DMSTPG	DMSTPG	Dump functions of TAPE command.
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.
	DMSSTRKX	Makes an 800-byte disk area that is no longer needed by one program available for allocation to another.
DMSTYP	TYPE	Processes the TYPE command. Types all or a specified part of a given file on the user's console.

Module Name	Entry Points	Function
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.
DMSUTL	DMSUTL	List, copy, or compress LOADLIBs.
DMSVAN	DMSVAN	First table of Access Method Services nonshared (nonreentrant) modules.
DMSVAS	DMSVAS	Contains a table of access method services shared (reentrant) modules.
DMSVAX	DMSVAX	Second table of access method services nonshared (nonreentrant) modules.
DMSVBM	DMSVBM	Contains table of simulated VSE phases located in CMSBAM DCSS.
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 VSE ASSGN statements for OS user; maps all OS VSAM macro requests to VSE specifications; equivalents, where necessary; traps all transfers of control between VSAM and the OS user and sets the appropriate operating environment flags.
DMSVLT	DMSVLT	Simulates VSE $$BOSVLT transient. Provides return linkage from SAM OPEN/CLOSE routines to CMS/DOS routines.
DMSVSR	DMSVSR	Resets any flags or fields set by VSAM processing; purges the VSAM discontiguous shared segment.
DMSVVN	DMSVVN	Contains table of VSE/VSAM nonshared (nonreentrant) modules.
DMSVVS	DMSVVS	Contains table of VSE/VSAM shared (reentrant) modules.
DMSXBG	DMSXBG	Allocates and initializes storage for the XEDIT work area; checks terminal characteristics. Processes the XEDIT command.
DMSXCG	CDELETE CHANGE COMPRESS COPY COUNT COVERLAY DELETE DUPLICAT EXPAND LOWERCAS MOVE OVERLAY UPPERCAS RECOVER SHIFT	Processes the subcommands (entry points) listed.

Module Name	Entry Points	Function
DMSXCM	STACK	Processes the subcommands (entry points) listed.
	CMS	
	CP	
DMSXCN	DMSXCN	Arranges compound characters into canonical form; disregards tab characters as special characters.
DMSXCP	DMSXCP	Simulates the VSE EXCP function (VSE SVC 0) in the CMS/DOS environment. EXCP (Execute Channel Program) requests initiation of an I/O operation to a specific logical unit.
DMSXCT	CMSG	Processes the CMSG subcommand.
	CURSOR	Processes the CURSOR subcommand.
	DMSXCTPN	Processes the SET POINT subcommand.
	DMSXCTRS	Reserves a screen line for the user.
	DMSXCTTE	Processes the SET TERMINAL subcommand.
	DMSXCTSC	Processes the SET SCREEN subcommand.
	EMSG	Processes the EMSG subcommand.
	FILE	Processes the FILE subcommand.
	MSG	Processes the MSG subcommand.
	PRESERVE	Processes the PRESERVE subcommand.
	PURGE	Processes the PURGE subcommand.
	READ	Processes the READ subcommand.
	RENUM	Processes the RENUM subcommand.
	REPEAT	Processes the REPEAT subcommand.
	RESET	Processes the RESET subcommand.
	RESTORE	Processes the RESTORE subcommand.
	SAVE	Processes the SAVE subcommand.
	TYPE	Processes the TYPE subcommand.
DMSXDC	DMSXDCOD	Scans input from the terminal for a subcommand or macro; operands are decoded and placed in buffers. Executes the MACRO and COMMAND subcommands. Performs synonym substitution.
	DMSXDCSY	Processes the SET SYNONYM subcommand.
DMSXDS	DMSXDSRD	Reads a data set (SAM) from an OS formatted disk.
DMSXED	XEDIT	Processes the XEDIT subcommand; brings a file into the ring of files in storage.
	DMSXEDRT	Removes an edited file from storage (QUIT).
DMSXER	DMSXER	Displays an error message in the standard CMS format: DMSxxxnnnc message text....
DMSXFC	DMSXFCNX	Moves the line pointer to the next line.
	DMSXFCUP	Moves the line pointer to the previous line.
	DMSXFCPL	Moves the line pointer to line number "n".
	DMSXFCML	Moves the current line UP1 or NEXT1.
	DMSXFCCD	Moves the column pointer to the right.
	DMSXFCCG	Moves the column pointer to the left.
	DMSXFCLM	Locates a specified string (FIND) in the current line.
	DMSXFCDP	Inserts the column pointer as "_" in a buffer.
	DMSXFCIN	Inserts a new line in the file.
	DMSXFCRL	Replaces a line in the file with a new one.
	DMSXFCSU	Deletes one line from the file.
	DMSXFCRC	Performs string substitution in the current line.
	DMSXFCRM	Performs an overlay function on the current line.

Module Name	Entry Points	Function
CMSXFC (cont.)	DMSXFCSP	Handles special characters, ex., tab, backspace.
	DMSXFCDC	Displays SET VERIFY or SET TABS columns.
	DMSXFCLR	Defines the logical record length.
	DMSXFCTR	Defines the truncation column.
	DMSXFCHT	Defines the top of range.
	DMSXFCBT	Defines the end of range.
	DMSXFCGA	Defines the zone left column.
	DMSXFCDR	Defines the zone right column.
	DMSXFCPC	Sets the column pointer in column "n".
	DMSXFCCC	Sets the cursor to column "c" in the file.
	DMSXFCCL	Sets the cursor to line "1" in the file.
	DMSXFCTB	Sets up the tabulation columns.
DMSXFD	DMSXFDFI	Writes the file on disk.
	DMSXFDSR	Serializes the file in storage.
	DMSXFDTG	Performs target processing.
	DMSXFDLE	Locates an extended string (a string with arbitrary characters).
	DMSXFDLN	Locates a named line.
DMSXGT	GET	Process the GET subcommand.
DMSXHL	HELP	Invokes the CMS HELP facility.
DMSXIN	DMSXINTF	Initializes a file descriptor block.
	DMSXINLA	Aborts the profile macro if an error occurs during LOAD.
	DMSXINLD	Processes the LOAD subcommand.
	DMSXINLX	Processes an explicit LOAD, from the profile macro.
	DMSXINOP	Handles XEDIT command options.
DMSXIO		Performs I/O at the terminal.
	DMSXIORD	Reads at the terminal.
	DMSXIOWR	Writes at the terminal.
DMSXMA	DMSXMAOP	Executes XEDIT macros (EXEC 2 files).
	DMSXMAED	Performs subcommand processing from ECOMMAND (SVC X'ED').
	DMSXMAEX	Executes subcommand from XEDIT macros.
	DMSXMARD	States existence of a macro and reads it.
	DMSXMARS	Releases a macro from free storage.
DMSXMC	CFIRST	Processes the CFIRST subcommand.
	CLAST	Processes the CLAST subcommand.
	CLOCATE	Processes the CLOCATE subcommand.
	LEFT	Processes the LEFT subcommand.
	RIGHT	Processes the RIGHT subcommand.
	DMSXMCVR	Processes the SET VERIFY subcommand.
DMSXMD		Processes the subcommands (entry points) listed.
	INPUT	
	ADD	
	REPLACE	
	CREPLACE	
	CINSERT	

Module Name	Entry Points	Function
DMSXML		Processes the subcommands (entry points) listed.
	BACKWARD	
	BOTTOM	
	DOWN	
	FIND	
	FINDUP	
	FORWARD	
	FUP	
	LOCATE	
	NEXT	
	NFIND	
	NFINDUP	
	NFUP	
	TOP	
	UP	
DMSXMS	DMSXMS	Arranges records within a file in a descending or ascending sequential order (SORT macro).
DMSXPT	PUT	Processes the PUT subcommand.
	PUTD	Processes the PUTD subcommand.
	DMSXPTER	Erases the temporary file used by GET/PUT(D).
DMSXPX	DMSXPXDC	Decodes prefix subcommands.
	DMSXPXEX	Executes prefix subcommands.
DMSXRE	DMSXRE	Processes the RENUM subcommand.
DMSXSC	DMSXSCDP	Dispatches a logical screen.
	DMSXSCIM	Builds and displays all the logical screens.
	DMSXSCPR	Checks if the cursor is in a protected area.
	DMSXSCCN	Scans the buffer read from the screen.
	DMSXSCCP	Prints an image of the physical screen (COPYKEY function).
	DMSXSCIO	Handles I/O on a 3270.
	DMSXSCRV	Displays a line in the 3270 command line.
DMSXSD	DMSXSDLS	Builds a logical screen block.
	DMSXSDSC	Builds a logical screen.
	DMSXSDPH	Builds the physical screen.
	DMSXSDLN	Builds a line to be displayed.
	DMSXSDML	Moves a line from screen buffer in the file.
	DMSXSDMS	Builds the scale line.
DMSXSE		Processes the subcommands (entry points) listed.
	QUERY	
	SET	
	TRANSFER	
DMSXSG	DMSXSG	Bootstrap to load the XEDIT shared segment; performs initialization for the editor.
DMSXSS	SOS	Processes the SOS subcommand.
	DMSXSSEX	Handles EDGAR /SOS compatibility.

Module Name	Entry Points	Function
DMSXST	DMSXSTLG	Gets a free line in storage.
	DMSXSTNB	Computes the number of free lines available.
	DMSXSTCP	Combines free lines in one free block.
	DMSXSTEX	Dynamically extends the storage for the files.
DMSXSU	DMSXSUVR	Editing supervisor.
	DMSXSUPE	Executes the profile macro.
	QUIT	Executes the QUIT subcommand.
	DMSXSUFL	Flushes subcommand execution if no more savearea.
	DMSXSUNP	No operation (used when a macro ends).
	DMSXSU	Redisplays the last input in the input area.
	DMSXSUIG	Maintains file integrity on multiple windows.
	DMSXSUTY	Types the current line.
	DMSXSUEF	Types "EOF".
	DMSXSUTF	Types "TOF".
	DMSXSUTE	Types "TOF" or "EOF".
	DMSXSUTP	Checks displacement to a taget line.
	DMSXSUNC	Types "NO CHANGE".
	DMSXSUNF	Types "NOT FOUND".
	DMSXSUPR	Checks for prefix subcommand waiting.
	DMSXSULG	Computes line length.
	DMSXSUEX	Executes a subcommand.
	DMSXSUCK	Checks if fname ftype fmode are valid.
	DMSXSUTS	Computes autosave identification.
	DMSXSUCN	Performs EBCDIC-binary conversion.
	DMSXSUCC	Performs binary-EBCDIC conversion.
	DMSXSUCH	Performs EBCDIC-hexadecimal conversion.
	DMSXSUHC	Performs hexadecimal-EBCDIC conversion.
DMSXTB	DMSXTBHC	Address of hash code table.
	DMSXTBRQ	Address of subcommand table.
DMSXTF	DMSXTF	Filetype descriptor table.
DMSXUP	DMSXUPCK	Checks for proper serialization.
	DMSXUPAT	Applies one update file to the source file.
	DMSXUPCT	Handles CNTRL and AUX files for multi-level update.
	DMSXUPBL	Builds the update file (subcommands SAVE or FILE).
	DMSXUPDL	Handles deleted lines in the source file.
	DMSXUPR2	Builds error messages.
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.
DMSZER	DMSZER	Perform AUTO call setup for CMS loader.
DMSZES	DMSZES	Builds a copy of the SSTAT in the CMSZER segment.
DMSZEX	DMSZEX	Marks the end of the segment zero text files.
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	ABNCMSG	ABNERLST	ABNPAS13	ABNPSW	ABNREGS	ABNRR	ABWSECT	ACITDB	ADMSABW	ADMSCAT	ADMSCRD
	ADMSCWT	ADMSERR	ADMSFREB	ADMSFRES	ADMSITSR	ADTADDED	ADTARES	ADTEDF	ADTFDA	ADTFFSTV	ADTFLG1	ADTFLG2	ADTFLG4
	ADTFQQF	ADTFRO	ADTFROS	ADTHBCT	ADTLD	ADTMFDA	ADTMFDN	ADTPQM3	AEXCAB	AFINIS	AFVS	AINTAB	AINTRTBL
	AIOSECT	ALADAD	ALOKTABA	ALOKTABE	ALOKTABS	ALOKTB	AOPSECT	AOUTRTBL	ASTGSB	ASUBFST	ASUBSECT	ASUBSTAT	ATTN
	AUSABRV	AUSRAREA	AUSRILST	AUSRITBL	AVSAMSYS	BALR	BATFLAGS	BATFLAG2	BATLOAD	BATRUN	BATSYSAB	CMNDLINE	CODE203
	CONRDCNT	CONRDCOD	CONREAD	CONSTACK	CONWAIT	CURRSAVE	DBGABN	DBGEXEC	DBGFLAGS	DBGNSHR	DBGSHR	DCHDWSIZ	DCHFLG2
	DCHFWPTR	DCHSECT	DCHSHR	DCSSFLAG	DCSSVTLD	DMSDBG	DOSFIRST	DOSFLAGS	DOSMODE	DOSNUM	DOSSVC	DOSTRANS	EGPRS
	FCBFIRST	FCBNUM	FCHLENG	FREELOWE	FVSECT	IONTABL	IOSECT	IPLPSW	KXFLAG	KXWANT	LABFIRST	LABNUM	LASTUSED
	LDMSROS	LOC	LOCKTAB	MACDIRC	MDPCALL	MISFLAGS	MODFLGS	NOPAGREL	NRMRET	NUCAFCHS	NUCCBLKS	NUCON	NUCOSFLG
	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	SETUP	SSAVE	SUBFLAG	SUBSECT	UFDBUSY	USERKEY	VCADTNXT	VSAMFLG1	VSAMRUN	VSAMSOS		
DMSACC	ADMSALU	ADMSFREB	ADMSLADN	ADMSTRKA	ADMSTRKM	ADTADDED	ADTAMHO	ADTAMP1	ADTAMP2	ADTAMP3	ADTARES	ADTBWPTR	ADTDBSIZ
	ADTDOP	ADTDTA	ADTEDF	ADTEDFAE	ADTFALUF	ADTFDA	ADTFDOS	ADTFFSTF	ADTFLG1	ADTFLG2	ADTFLG3	ADTFLG4	ADTFMIN
	ADTFORCE	ADTFRO	ADTFROS	ADTFRW	ADTFSTC	ADTFSTSZ	ADTHBCT	ADTLAST	ADTLD	ADTLEFT	ADTLFST	ADTLHBA	ADTM
	ADTMFDA	ADTMFDN	ADTMSK	ADTMX	ADTNUM	ADTPQM2	ADTPQM3	ADTPTR	ADTQQM	ADTRES	ADTSECT	ADTUSED	ADT1ST
	AFINIS	AFVS	AKILLEX	AWRTK	BALR	CODE203	CURRSAVE	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFLG2	DCHFULL	DCHFWPTR
	DCHPFIXL	DCHSECT	DCHTDISP	DCHTRUNK	DSKADR	DSKLOC	DSKLST	DTAD	DTADC	D1	D2	D200	D3
	EIGHT	ERRCODE	FVSECT	FWADDR	IADT	KXFLAG	KXWANT	LOC	MISFLAGS	MODESET	MSG	NODISK	NUCON
	NUM	OPTBYTE	RESET	RWCNT	R0	R1	R10	R11	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	TYPE	UFDBUSY	VCADTLKP	VCADTNXT	VCFSTLKP	WRBIT
DMSACF	ADMSALU	ADMSFREB	ADTADD	ADTARES	ADTCFST	ADTCHBA	ADTDBSIZ	ADTDFP1	ADTDFP2	ADTDFP3	ADTEDF	ADTFALNM	ADTFALTY
	ADTFALUF	ADTFDA	ADTFFSTF	ADTFLG1	ADTFLG2	ADTFLG3	ADTFLG4	ADTFMDRO	ADTFORCE	ADTFRO	ADTFROS	ADTFRW	ADTFSORT
	ADTFSTC	ADTFSTSZ	ADTFTYP	ADTHBCT	ADTLFST	ADTLHBA	ADTM	ADTMFDA	ADTMFDN	ADTNFST	ADTPQM2	ADTRES	ADTSECT
	AFVS	ARDTK	ASORTFST	ATYPSRCH	BALR	CODE203	DCHBWPTR	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFWPTR	DCHPFIXL	DCHSECT
	DCHSEQBD	DCHTDISP	DCHTRUNK	DSKADR	DSKLOC	DSKLST	D1	D3	ERBIT	ERRCOD1	FVSECT	FWADDR	F65535
	LOC	NUCON	REGSAV0	REGSAV1	RWCNT	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	TYPE	UFDBUSY			
DMSACM	ABLKIND	ADEVIND	ADEVSUP	ADEVSUP2	ADIOSECT	ADMSFREB	ADMSROS	ADTADD	ADTAMP1	ADTAMP2	ADTARES	ADTCYL	ADTDBSIZ
	ADTDIOA	ADTDIOB	ADTDOP	ADTDTA	ADTEDF	ADTFBABF	ADTFBALB	ADTFDA	ADTFLG1	ADTFLG2	ADTFLG3	ADTFLG4	ADTFMFD
	ADTFORCE	ADTFQQF	ADTFRO	ADTFRW	ADTFSTSZ	ADTHBCT	ADTIDENT	ADTLAST	ADTLEFT	ADTMFDA	ADTMFDN	ADTMSK	ADTMX
	ADTMXBML	ADTNFST	ADTNUM	ADTPQM1	ADTPQM2	ADTPQM3	ADTQQM	ADTROX	ADTSECT	ADTUSED	AFVS	ALABELRD	ARDTK
	ATBLIND	BALR	CDMSROS	CODE203	DCHBWPTR	DCHDAMAP	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFWPTR	DCHPFIXL	DCHSECT	DCHSEQBD
	DCHTDISP	DCHTRUNK	DIOSECT	DSKADR	DSKLOC	DSKLST	DTAD	DTADC	DTADT	D1	D3	ERRCOD0	ERROR1
	ERROR2	ERROR3	FBACD1	FBACL1	FFD	FFE	FFF	FILE	FVSDSKA	FVSECT	FWADDR	F800	JSR0
	LDMSROS	LOC	LOCCNT	MODFLGS	NUCON	OSADTVTA	QQDSK1	REGSAV0	RWCNT	RWMFD	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SECTNUM	SEEKADR	SENSB	SIGNAL	SWTCH	SYSLOAD	TBENT	TYPE	UFDBUSY	UPBIT	VCADTLKP		
DMSALU	ABGCOM	ADMSFREB	ADMSROS	ADTADDED	ADTAMP1	ADTAMP2	ADTAMP3	ADTDAMAP	ADTDBSIZ	ADTDFP1	ADTDFP2	ADTDFP3	ADTEDF
	ADTFDA	ADTFLG1	ADTFLG2	ADTFLG3	ADTFLG4	ADTFMIN	ADTFQQF	ADTFRO	ADTFROS	ADTFRW	ADTFSTC	ADTFSTSZ	ADTFTYP
	ADTIDENT	ADTLABSZ	ADTM	ADTMFDA	ADTMFDN	ADTMSK	ADTMX	ADTNFST	ADTPQM1	ADTPQM3	ADTQQM	ADTRES	ADTROX

Module-to-Label Cross Reference

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	AFVS	BALR	CDMSROS	CODE203	DCHCHGD	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFLG1	DCHFLG2	DCHFWPTR	DCHPFIXL	DCHSECT
	DCHSHR	DOSFLAGS	DOSMODE	D1	D3	FCBDSMD	FCBFIRST	FCBNEXT	FCBOSFST	FCBSECT	FF	FLGSAVE	FVSECT
	FVSL1	LDMSROS	LOC	NUCON	OSADTFST	OSFST	OSFSTLTH	OSFSTNXT	REGSAVO	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SDISK
	STATEFST	STATFST2	VCADTLKP	VCADTNXT									
DMSAMS	AAMSSYS	ABGCOM	ADEVTAB	ADMSERL	ADMSFREB	ADTM	ADTSECT	AERASE	AESTATE	AESTATEW	ALTASAVE	APPSAVE	ASCANN
	ASYSNAMS	ATABEND	ATCBPTR	BALR	BGCOM	BLANK	CLEAR	CMSAMS	CODE203	COMNAME	DOSDD	DOSDEV	DOSDSMD
	DOSDUM	DOSEXTNO	DOSEXTTB	DOSFIRST	DOSFLAGS	DOSMODE	DOSNEXT	DOSRC	DOSSECT	DOSSVC	DOSVOLNO	DOSVOLTB	DOSYSXXX
	DTAD	DTAS	EPOINT	ERROR104	FF	F4096	LOADIT	LOC	LTK	LUBPT	MISFLAGS	NUCON	NUM
	OUTPRINT	PIBPT	PUBPT	RELPAGES	RESET	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	SYSNAMES	SYSNEND	TCBADR	TCBSAVE	VCADTLKW
	VMSIZE	VSAMFLG1	VSAMSERV	VSAMSOS									
DMSARE	ABATPROC	ADMSALU	ADMSFREB	ADTADDED	ADTBWPTR	ADTDTA	ADTEDF	ADTEDFAE	ADTFLG1	ADTFLG2	ADTFLG3	ADTFLG4	ADTFNOAB
	ADTFRO	ADTFROS	ADTFRW	ADTFSTC	ADTLD	ADTM	ADTPTR	ADTSECT	AFINIS	ASORTFST	AUPDISK	BALR	BATCPEX
	BATFLAGS	BATLOAD	BATRUN	BATUSEX	CHKMODE	CODE203	DTAD	ERROR1	ERROR2	FF	LOC	NUCON	NUM
	R0	R1	R10	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	VCADTLKP	VCADTNXT									
DMSARN	ADTFLG1	ADTFRW	ADTM	ADTMX	AFVS	AOPSECT	ASTRINIT	BATFLAGS	BATRUN	BLANK	COMPSWT	DUMMY	FCBBUFF
	FCBBYTE	FCBCATML	FCBCLOSE	FCBDD	FCBDEV	FCBFORM	FCBINIT	FCBIOSW	FCBITEM	FCBPROC	FCBPROCC	FCBPROCO	FCBREAD
	FCBSECT	FINIS	FVSECT	FVSFSTAD	INPUT	IOBCSW	IOBIN	IOBIOFLG	LINECNT	MISFLAGS	NOERASE	NOPRINT	NUCON
	NUM	OSSFLAGS	RELPAGES	RESET	RETURN	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	SAVEAREA	VCADTLKW			
DMSARX	AADTLKW	ADTFLG1	ADTFRW	ADTM	ADTMX	AFVS	BLANK	CMNDLINE	COMPSWT	CONWR	DEVICE	DMSARD	DUMMY
	ERR1	FCBBUFF	FCBBYTE	FCBCATML	FCBCLOSE	FCBDD	FCBDEV	FCBDSK	FCBDSNAM	FCBFORM	FCBINIT	FCBIOSW	FCBITEM
	FCBPROCC	FCBRDR	FCBREAD	FCBSECT	FCBTAP	FILE	FLAG1	FLAG2	FREELOWE	FVSECT	FVSFSTAD	IOBCSW	IOBIN
	IOBIOFLG	MAINHIGH	MISFLAGS	NOERASE	NOPRINT	NOTERM	NOTFOUND	NUCON	NUM	OPSECT	OSIOTYPE	OSSFLAGS	RELPAGES
	RESET	RETURN	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SAVEAREA	STATCOPY	SYSUT1					
DMSASM	AADTLKW	ADTFLG1	ADTFRW	ADTM	ADTMX	AFVS	BLANK	CMNDLINE	COMPSWT	CONWR	DEVICE	DEVTYPE	DMSASD
	DOSFLAGS	DOSSVC	DUMMY	ERR1	FCBBUFF	FCBBYTE	FCBCATML	FCBCLOSE	FCBDD	FCBDEV	FCBDSK	FCBDSNAM	FCBFORM
	FCBINIT	FCBIOSW	FCBITEM	FCBPROCC	FCBRDR	FCBREAD	FCBSECT	FCBTAP	FILE	FLAG1	FLAG2	FREELOWE	FVSECT
	FVSFSTAD	IOBCSW	IOBIN	IOBIOFLG	MAINHIGH	MAX	MISFLAGS	NOERASE	NOPRINT	NOTERM	NOTFOUND	NUCON	NUM
	OPSECT	OSIOTYPE	OSSFLAGS	PRFUSYS	PROTFLAG	RELPAGES	RESET	RETURN	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVEAREA	STATCOPY
	SYSUT1												
DMSASN	ABATABND	ABGCOM	ADEVTAB	ADTDTA	ADTFDOS	ADTFLG1	ADTFLG2	ADTFRO	ADTFROS	ADTFRW	ADTSECT	ASYSREF	BATDCMS
	BATFLAGS	BATFLAG2	BATRUN	BGCOM	BLANK	CHKMODE	CLASTAPE	DENSITY	DEVTAB	DEVTYPE	DOSFLAGS	DOSMODE	DOSVSAM
	DTAD	DTADC	DTADT	ERR70E	FF	FLAG2	FLAG3	MATCH	NUCON	NUM	PACK	PUBPT	R0
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	SCAN	SCANNING	TAPE1	TAPE4	TRACK7	TYP2401	TYP2415	TYP2420	TYP3420	TYP8809	VCADTLKP

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSAUD	ADMSFREB	ADMSTRKA	ADMSTRKD	ADTADD	ADTAMHD	ADTAMP1	ADTAMP2	ADTARES	ADTCHMAP	ADTCYL	ADTDAMAP	ADTDBSIZ	ADTDFP1
	ADTDFP2	ADTDFP3	ADTDOP	ADTDTA	ADTEDF	ADTFDA	ADTFLG3	ADTFLG4	ADTFNOAB	ADTFSTC	ADTFSTSZ	ADTFUPD1	ADTHBCT
	ADTLAST	ADTLEFT	ADTLHBA	ADTMFDA	ADTMFDN	ADTMSK	ADTNUM	ADTPQM1	ADTPQM2	ADTQQM	ADTSECT	ADTUSED	ADT1ST
	AFVS	AKILLEX	ALABELWR	ATRKLKP	ATRKLKPX	AWRTK	BALR	BALRSAVE	CODE203	DCHBWPTR	DCHCHGD	DCHCHMAP	DCHCHOP
	DCHDALLO	DCHDAMAP	DCHDATA	DCHDWSIZ	DCHFLG1	DCHFLG2	DCHFULL	DCHFWPTR	DCHNEW	DCHPFIXL	DCHSECT	DCHTDISP	DCHTRUNK
	DSKADR	DSKLOC	DSKLST	DTADT	D1	EDF010	EDF020	FFD	FFE	FFF	FINISLST	FVSDIOPL	FVSDSKA
	FVSECT	FVSELMNL	FVSELMNT	FWADDR	F800	KXFLAG	KXWANT	LOC	NUCON	REGSAV0	RWCNT	RWFSTRG	RWMFD
	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6
	R7	R8	R9	TYPE	UFDBUSY	UPBIT	XFF						
DMSBAB	ATCBPTR	DOSRC	NUCON	TCBABPTR	TCBADR	TCBPCPTR	TCBSAVE	VSAMFLG1	VSAMSERV				
DMSBOP	ABAMSYS	ABGCOM	ACBCAT	ACBDDNM	ACBERFLG	ACBIN	ACBINFLG	ACBMACR1	ACBOFLGS	ACBOLIGN	ACBOUT	ACBSTSKP	ADMSERL
	ADMSFREB	ADTFDOS	ADTFLG1	ADTFLG2	ADTFLG3	ADTFMFD	ADTFRO	ADTFROS	ADTFRW	AERASE	AESTATE	ASYSCOM	ASYSNAMS
	ASYSREF	AVSAMSYS	BALR	BAMFLAGS	BGCOM	BLANK	BSR	BUFF2	CHKMODE	CMSVSAM	CODE203	CONTROL	CONVERT2
	DEC	DLBLAREA	DOSBAM	DOSBLKSZ	DOSBUFF	DOSDD	DOSDEV	DOSDSMD	DOSDTF	DOSDUM	DOSEPL	DOSEXT	DOSFIRST
	DOSFLAGS	DOSFORM	DOSINIT	DOSNEXT	DOSNUM	DOSOP	DOSOSFST	DOSRC	DOSSECT	DOSSYS	DOSTRANS	DOSUCAT	DOSUCNAM
	DOSVSAM	DOSXXX	DTFBLHLD	DTFBLKSZ	DTFCCW	DTFCPDTL	DTFCTRLF	DTFDEVTP	DTFFLG1	DTFFLG2	DTFINPUT	DTFNAME	DTFOPEN
	DTFSD	DTFTPDI	DTFTPSD	DTFTYPE	DTFWFUNB	DTFWKRLT	DTFWRKFL	DTFX	DTFXIDEN	DTFXORSP	EIGHT	ERR1	FF
	FILE	FILETYPE	FREELN	FREESTOR	FSF	FSR	IJBFLG04	IKQACB	INPUT	LIOCSCOM	LOAD	LOADIT	LOC
	LOOP	LUBPT	MODEL3	MODEL5	MODEL7	MODEL8	MONREGSV	NICLPT	NUCON	NUM	OCTCPDI	OCTDITYP	OCTDXBUF
	OCTGVSIZ	OCTMONAD	OCTMONSV	OCTS	OCTSDWDS	OCTSLEN	OCTSPPSV	OCTSTADR	OCT1FLAG	OPENSW	OSFST	OSFSTFM	OSFSTRFM
	OUTPUT	PACK	PIBPT	PLIST	PUBADR	PUBCUU	PUBDEVT	PUBPT	PUBTAPM1	PUBTAPM2	PUBTAP7	READ	RESET
	REW	RMSROPEN	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SAVEFN	SAVE1	SAVE2	SENSE	SVEAIA	SVEARA	SVEA0908	SWITCH
	SYSCOM	SYSNAMES	SYSNEND	TLBBLOK	TLBCALL	TLBDOS	TLBDTFPT	TLBDWSZ	TLBLABT	TLBMODE	TLBNAME	TLBOPIN	TLBOPOUT
	TLBSL	TLBTAPID	TLBTYPE	TRACK7	TYPE	USRREGSV	USRRGLEN	VCADTLKP	VIPSOP	VMSIZE	VSAMFLG1	VSAMOPEN	VSAMRUN
	VSAMSERV	WRITE	WTM										
DMSBRD	AACTFREE	AACTLKP	ADMSFREB	ADTEDF	ADTFLG4	ADTSECT	AFTADT	AFTCLA	AFTCLB	AFTCLD	AFTCLN	AFTDBA	AFTDBD
	AFTDBN	AFTFBA	AFTFCLA	AFTFLG	AFTFST	AFTID	AFTIN	AFTPFST	AFTRD	AFTWRT	AFVS	ARDTK	AUSRAREA
	BALR	CODE203	DISK$SEG	DMSERDBF	DMSLFS	D1	ERROR2	FILNAM	FSCBD	FSCBFLG	FSCBFV	FVSECT	FVSFLG0
	FVSUFSTC	ITEM	NUCON	PLIST	READ	READCNT	REGSAV3	RETURN	RWFSTRG	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	STATEFST
	STATER0	STATER1	STATFST2	SVCOPSW	TYPE	VMSIZE							
DMSBTB	ABATABND	ABATLIMT	ABATPROC	AFVS	ALDRTBLS	AUSRAREA	BATDCMS	BATFLAGS	BATFLAG2	BATLOAD	BATNOEX	BATRUN	BATUSEX
	FVSECT	LOAD	LOCCNT	NUCON	RESET	R0	R1	R12	R14	R15	R2	R3	R4
	R5	R8	TBENT	TYPE									
DMSBTP	ABNBIT	ADMSCRD	AFVS	ASCANN	ASYSNAMS	BATCPEX	BATDCMS	BATFLAGS	BATFLAG2	BATMOVE	BATNOEX	BATRERR	BATSTOP
	BATTERM	BATUSEX	BATXCPU	BATXLIM	BATXPRT	BLANK	BLK	CMSSEG	CONVERT	CONWAIT	FVSECT	IPLADDR	KEYS
	LINE	LINKPARM	MSG	NUCON	NUMFINRD	PACK	RESET	RETURN	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SYSNAME	SYSNAMES
	SYSNEND	UFDBUSY											

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSBWR	AACTFREE	AACTFRET	AACTLKP	ADMSERL	ADMSFREB	ADTDTA	ADTEDF	ADTFLG1	ADTFLG3	ADTFLG4	ADTFRW	ADTFSTC	ADTFXCHN
	ADTM	ADTMX	ADTNACW	ADTRES	AFTADT	AFTCLA	AFTCLB	AFTCLD	AFTCLDX	AFTCLN	AFTCLX	AFTDBA	AFTDBD
	AFTDBF	AFTDBN	AFTFBA	AFTFCLA	AFTFCLX	AFTFLG	AFTFLG2	AFTFST	AFTFULD	AFTID	AFTIN	AFTNEW	AFTOCLDX
	AFTOLDCL	AFTRD	AFTWRT	AFVS	AKILLEX	AQQTRK	AQQTRKX	ARDTK	ATFINIS	ATRKLKP	ATRKLKPX	AUPDISK	AWRTK
	BALR	CODE203	DEBDCBAD	DMSERR	DMSLAD	DMSLFSW	DOSFLAGS	DOSMODE	EIGHT	ERROR3	FCBDD	FCBFIRST	FCBNUM
	FCBSECT	FF	FVSECT	KXFLAG	KXWANT	LASTREC	LOC	NUCON	PLIST	PS	REGSAV3	RESET	RWFSTRG
	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6
	R7	R8	R9	SAMELEN	SVCOPSW	TYPE	UFDBUSY	UND	VMSIZE	WRBIT			
DMSCAT	ADMSFREB	BALR	CMNDLIST	CODE203	D1	D3	MISFLAGS	NEGITS	NUCFSTLN	NUCLSTLN	NUCNBSTK	NUCNLSTK	NUCON
	NUMFINRD	R0	R1	R12	R13	R14	R15	R2	R3	R4	TYPE		
DMSCIO	ABATABND	ABATLIMT	ADMSERL	ADMSIOW	BATFLAGS	BATLSECT	BATNOEX	BATPUNC	BATPUNL	BATRUN	BATXLIM	BATXPUN	CAW
	CSW	ERROR3	NUCON	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3
	R4	R5	R6	R7	R8								
DMSCIT	ADMSFREB	AFVS	AIOSECT	ASVCSECT	ATTN	ATTNHIT	BALR	BATFLAG2	BATSTOP	BLANK	CAW	CE	CMSTAXF
	CODE203	CONCCWS	CONSTACK	CSW	CURRIOOP	DBGEXEC	DBGEXINT	DBGFLAGS	DE	DMSERR	D1	D2	D3
	FSTFINRD	FVSECT	IOOPSW	KXFLAG	KXWANT	LOC	LSTFINRD	MISFLAGS	MSGFLAGS	NOTYPING	NUCFSTLN	NUCNBSTK	NUCNLSTK
	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	TSOATCNL	TSOFLAGS	UE	UFDBUSY	WAIT	WAITSAVE	XFF				
DMSCLS	ABAMSYS	ACBAMO	ADMSERL	ADMSFREB	ASYSREF	AVSAMSYS	AVSRWORK	BALR	BAMFLAGS	BGCOM	BLANK	BLOCKCNT	BSR
	CHKEOF	CODE203	CON	CONVERT2	DOSBAM	DOSSECT	DOSTRANS	DTFCCWA	DTFCSW	DTFFLG1	DTFFLG2	DTFFMT1R	DTFIGNOP
	DTFOPEN	DTFSD	DTFX	DTFXIDEN	DTFXORSP	EIGHT	EOFSW	FILE	FREESTOR	FSF	IKQACB	LIOCSCOM	LOC
	LUBPT	NICLPT	NUCON	OCTMONAD	OCTMONSV	OCTS	OCTSPPSV	PIBPT	PLIST	PUBADR	PUBCUU	PUBDEVT	PUBPT
	PUBTAPM1	READ	RESET	REW	REWIND	RUN	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SENSE	SVEAIA	SVEARA	SVEA0908
	SWITCH	TAPE	TLBBLOK	TLBCALL	TLBCLIN	TLBCLOUT	TLBDOS	TLBDTFPT	TLBDWSZ	TLBEOV	TLBLABT	TLBMODE	TLBNAME
	TLBSL	TLBTAPID	TLBTYPE	TYPE	USRREGSV	USRRGLEN	VIPINIT	VIPSOP	VIPTCLOS	VSAMFLG1	VSAMSERV	WRITE	WTM
DMSCMP	ADMSFREB	ADTM	ADTSECT	AFINIS	AFVS	ARDBUF	AREA	BALR	BLANK	CODE203	FILE	FVSECT	FVSFSTAC
	FVSFSTAD	LOC	NUCON	READ	RETURN	TYPE							
DMSCPF	AABBREV	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	ADTDBSIZ	ADTEDF	ADTFLG1	ADTFLG4	ADTFRW	ADTFSTSZ	ADTM	AFSTLKP	AFSTLKW
	AFTADT	AROUND	BUFAD	CL	DOSFLAGS	DOSSVC	D1	D2	HEX	INPUT	LOOP	MISFLAGS	NUCON
	OPSECT	PACK	RELPAGES	RESET	R0	R1	R10	R11	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SAVEAREA	TYPE	UNPACK	WRFV		
DMSCRD	ABATPROC	ADMSFREB	AFVS	AINTRTBL	AOPSECT	BALR	BATFLAGS	BATLOAD	BATRUN	CODE203	CONINBLK	CONINBUF	CSW

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	DMSCITB	DMSERR	D1	D3	D6	FSTFINRD	FVSECT	KXFLAG	KXWSVC	LOC	LSTFINRD	MISFLAGS	MSGFLAGS
	NOTYPING	NUCFSTLN	NUCLSTLN	NUCNBSTK	NUCNLSTK	NUCON	NUMFINRD	NUMPNDWR	OPSECT	PENDREAD	QSWITCH	R0	R1
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R8	R9	SCAN
	TSOATCNL	TSOFLAGS	TYPE	WAITLST									
DMSCVH	AADTNXT	ABAMSYS	ADMSERL	ADTFDOS	ADTFLG1	ADTFLG2	ADTFRO	ADTFROS	ADTFRW	ADTM	ADTPTR	ADTSECT	ASYSREF
	BGCOM	BLANK	CVHCLOSE	CVHCOV	CVHDLIST	CVHFLAGS	CVHFLG1	CVHFLG2	CVHIOA	CVHNAME	CVHOPEN	CVHRADR	CVHRETA
	CVHRF1	CVHSCR	CVHSYSNO	CVHWADR	CVHWANY	DLCVHADR	DLDLGOT	DLFLAGS	DLLEN	DLOPENED	DLSTDWDS	DLSYSNO	DOSBUFF
	DOSDD	DOSDSMD	DOSDSTYP	DOSFIRST	DOSF1AD	DOSNEXT	DOSOP	DOSSECT	ERROR2	F1DSI	F1DSN	F1DWDS	F1END
	F1EXSEQ	F1EXTYP	F1FTYPE	F1ID	F1LEN	F1LVOL	F1START	IJJHCPL	IJJHDLST	IJJHFMT1	LOC	LUBPT	NICLPT
	NUCON	PIBADR	PIBLUBNO	PIBPT	PUBADR	PUBDSKM	PUBPT	R0	R1	R10	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9				
DMSCWR	ADMSFREB	AFVS	AOPSECT	AOUTRTBL	BALR	BLANK	CODE203	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	WAITLST										
DMSCWT	AFVS	AOPSECT	FVSECT	KXFLAG	KXWSVC	NUCON	NUMPNDWR	OPSECT	PENDREAD	R0	R1	R10	R11
	R12	R14	R15	R9	WAITLST								
DMSDAS	ABGCOM	ADMSERL	ADTDTA	ADTFLG1	ADTFLG2	ADTFRO	ADTFROS	ADTFRW	ADTM	ADTPTR	BGCOM	D1	D2
	FREESTOR	IADT	LOC	LUBPT	MSG	NICLPT	NUCON	RESET	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9		
DMSDBD	ADEVTAB	ARGS	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								
DMSDBG	ABNPSW	ABNREGS	ABWSECT	ADMSCRD	ADMSERL	AIOSECT	AKILLEX	AOPSECT	ARGMAX	ARGS	ARGSAV	ARGSCT	BALRSAVE
	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	F6	GPRLOG	HEX	HEXHEX	IC	INPUT	INPUTSIZ	INPUT1	IOOPSW	IPLPSW	JFLAGS
	LASTDMP	LINE	LOWSAVE	MVCNT	MVCNT1	MVCNT2	NUCON	OPSECT	ORG	OUTPT1	PGMOPSW	PRFPOFF	PROTFLAG
	RETSAV	RETURN	RSTNPSW	R0	R1	R10	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R8	R9	SAVE1	SAVE2	SILI	SSAVE	STOPAT	SYMTABLE	SYMTBG	TBLEND	TPFUSR
	TSYM	TYPFLAG	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	FBACD1	FBACL1	FREERO	FVSECT	IOCOMM	IOOLD	IOOPSW	KXFLAG	KXWANT
	LASTCYL	LASTHED	LASTREC	LOC	LOOP	NUCON	PLIST	QQDSK1	QQDSK2	QQTRK	READ	RETREG	RETURN
	RWCCW	R0	R1	R10	R11	R12	R13	R14	R15	R2	R4	R5	R6
	R7	R8	R9	SAVEADT	SECTNUM	SEEKADR	SENCCW	SENSB	TOOBIG	TYPE	UFDBUSY	VCADTLKP	WRITE

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)											
	WRTKF	XRSAVE										
DMSDLB	ADMSFREB	ADTFDOS	ADTFLG2	ADTFROS	ADTSECT	ASYSREF	BALR	BGCOM	BLANK	CLEAR	CLR	CMSOP
	CODE203	CONREAD	CONVERT	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	DOSVOLNO	DOSVOLTB	DOSXXX	DOSYSXXX	DOUBLE	DUMMY	EGPR0	ERR70E	FILE	LFILEID	LOC
	LUBPT	MATCH	NICLPT	NUCON	PARM	PUBPT	READ	RESET	RETURN	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	SCAN2	SSAVE	STATLST	VCADTLKP	VSAMFLG1	VSAMSERV	VSJOBCAT				
DMSDLK	AADTLKP	AADTLKW	ADTFLG1	ADTFRW	ADTM	ADTSECT	AERASE	AESTATE	AFINIS	ARDBUF	AWRBUF	BALR
	BGCOM	BLANK	BLKSIZE	CHKTYPE	CLEAR	COMNAME	CSW	DEC	DOSDD	DOSDEV	DOSDSK	DOSFIRST
	DOSFLAGS	DOSMODE	DOSOP	DOSOSFST	DOSSECT	DOSSVC	ESD1ST	FREELOWE	FSCBAITN	FSCBBUFF	FSCBD	FSCBFM
	FSCBFN	FSCBFV	FSCBITNO	HEADER	HEX	JOBDATE	LABLEN	LINECNT	LOADBASE	LUB	LUBPR	LUBRES
	LUBRLB	NOAUTO	NODISK	NOMAP	NUCON	OSFST	OSFSTDSK	OSFSTXTN	OUTPUT	PACK	PLIST	PO
	PRTERR	PUBADR	PUBCUU	PUBDEVT	PUBPT	RA	READLST	REGSAVE	RESET	SCAN	SF	SYSLINE
	SYSUT1	TYPE	WRITE									
DMSDMP	ADMSFREB	ASYSREF	BALR	BGCOM	CODE203	CONVERT	EOCADR	LOC	NUCON	PLIST	R0	R1
	R12	R2	R3	R4	R5	R6	R7	TYPE				
DMSDOS	AADTLKP	AAMSSYS	ABAMSYS	ABGCOM	ABNBIT	ACLOSE	ACMSRET	ADIKQLAB	ADMSERL	ADMSFREB	ADTDTA	ADTFLG1
	ADTFRO	ADTFRW	ADTID	ADTPTR	ADTSECT	AFVS	ALOKTB	ALTASAVE	ANCHENDA	ANCHENTP	ANCHINST	ANCHLDPT
	ANCHLENG	ANCHPHLN	ANCHPHNM	ANCHSECT	ANCHSTSW	AOSRET	APPSAVE	ARFLG	ASYSCOM	ASYSNAMS	ASYSREF	ATCBPTR
	AUSRAREA	AVRADR	AVRDEVC	AVRFLAG	AVRLNO	AVRNLNO	AVRPUB	AVRTYPE	AVRVCI	AVRVNUM	AVRVOLID	AVRVTOC
	AVSAMSYS	AVSREOJ	BALR	BAMFLAGS	BGCOM	CALLER	CLKVALMD	CMSVSAM	CODE203	COMNAME	CONVERT	CONVERT2
	CURRDATE	CURRSAVE	DACTIVE	DCTACYL	DCTADR	DCTBTRK	DCTMAXR	DCTPCYL	DCTROH	DCTTCYL	DCTTFIX	DCTUCBC
	DEC	DIRC	DIRLL	DIRN	DIRNAME	DIRTT	DMSCVH	DMSDAS	DMSETR	DMSFCH	DMSLAB	DMSLCK
	DMSXCP	DOSBAM	DOSFLAGS	DOSRC	DOSSECT	DOSTRANS	DOSVSAM	D1	D2	EGPR0	EGPR1	EGPR14
	EGPR15	EGPR9	EIGHT	FCHLENG	FCHTAB	FREELOWE	FVSECT	HEX	IJBCCWT	IJBFTTAB	INTINFO	JCSW2
	JCSW4	JOBDATE	LOAD	LOADIT	LOC	LTK	LUBPT	MAINHIGH	MAINLIST	MAINSTRT	NICLPT	NOPAGREL
	NOTEXT	NUCON	OLDPSW	OPTFLAGS	OSADTDSK	OSTEMP	PIBADR	PIBFLG	PIBLUBNO	PIBPT	PIB2PTR	PIK
	PNOTFND	PPEND	PUBADR	PUBCUU	PUBPT	RESET	RETURN	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVEAREA
	SSAVE	SVCOPSW	SVC12SAV	SVEARA	SVEPSW	SVEPSW2	SVEROF	SVER00	SVER09	SYSCOM	SYSNAMES	SYSNEND
	TCBABPTR	TCBADR	TCBFLAGS	TCBPCPTR	TCBSAVE	TID	TPFSVO	TYPFLAG	UFDBUSY	VIPINIT	VMSIZE	VSAMFLG1
	VSAMOPEN	VSAMRUN	VSAMSERV	XFF								
DMSDSK	AACTLKP	ABATABND	ACFILE10	ACFILE20	ADMSBLKR	ADMSBLKW	ADTCHBA	ADTDBSIZ	ADTEDF	ADTFLG4	ADTFSTSZ	ADTFTYP
	ADTID	ADTM	AEPOINT	AERASE	AESTATE	AFINIS	AFTARP	AFTAWP	AFVS	AKILLEX	ARDBUF	ATYPSRCH
	AUPDISK	AWRBUF	BATDCMS	BATFLAGS	BATFLAG2	BATRUN	BLANK	CARDIN	CARDOUT	COPYEND	COPYEOF	DATAIN
	DATAOUT	DCHCHGD	DCHFLG1	DCHSECT	DEC	DUMP	D1	D6	EDF010	EDF015	EDF018	EDF020
	EDF030	EDF040	EDF200	EIGHT	EOFCHK	ERR70E	FBLOCK	FILE	FNAME	FSPARSE	FSTSAVE	FVSECT
	FVSFLG0	FVSFSTAD	FVSFSTM	FVSL1	FVSUFSTC	F65535	HEADER	IADT	INRPTR	INTYPE	INWPTR	KH12
	KXFLAG	KXWANT	NUCON	OVERLAP	READ	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVE10R	STATEFST	STATER1
	S202	TYPE	UFDBUSY	UPBIT	VBLOCK	VCFSTLKP	WRBIT	WRTYPE	XFF			

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSDSL	ADTFLG1	ADTFRW	ADTM	ADTSECT	AERASE	AESTATE	AFVS	BLANK	BLKSIZE	DA	DIRNAME	DIRR	DIRTT
	DOSFLAGS	DOSSVC	ERR104	FCBIOSW2	FCBITEM	FCBMVPDS	FCBSECT	FILE	FVSECT	FVSFSTAD	FXD	INPUT	NUCON
	OUTPUT	PO	PS	READ	RESET	R0	R1	R10	R12	R14	R15	R2	R3
	R4	R5	R8	SAVE1	VCADTLKP	WRITE							
DMSDSV	BGCOM	BLANK	BLANK2	COMNAME	DEC	DEVTYPE	DOSDD	DOSFIRST	DOSFLAGS	DOSMODE	DOSOP	DOSOSFST	DOSSECT
	EIGHT	FORM	FREELOWE	HEX	INPUT	LUB	LUBCLB	LUBP	LUBPR	LUBRES	LUBRLB	LUBSLB	MAXADDR
	NUCON	OSFST	OSFSTXTN	PLIST	PRTERR	PUBADR	PUBCUU	PUBPT	READ	RESET	SAVER0	SETUP	TYPE
DMSEDC	DUALNOS	EDCB	R0	R1	R10	R13	R14	R15	R2	R3	R4	R5	R6
	R7	R8	R9	SAVEAR									
DMSEDI	ADEVTAB	ADMSERL	AERASE	AESTATE	AEXTEND	AFINIS	AFSTFNRD	AINCORE	ALCHAR1	ALCHAR2	ALTLIST	ARDBUF	AREA
	ATTN	ATTNLEN	AUTOCNT	AUTOCURR	AUTOREG	AWRBUF	BLOC	BYTE	CARDINCR	CARDNO	CASEREAD	CASESW	CHGTRUNC
	CHNGCNT	CHNGFLAG	CHNGMSG	CHNGNUM	CMODE	CONSOLE	CONWAIT	CORITEM	COUNT	CRBIT	DEC	DECIMAL	DEVTAB
	DITCNT	DMSSCR	DOSFLAGS	DOSSVC	EDCB	EDCT	EDLIN	EDRET	ENDBLOC	ENDTABS	FILE	FILEMS	FLAG
	FLAG2	FLAG3	FMODE	FNAME	FPTR	FREELEN	FSIZE	FSTAIC	FSTD	FSTRECFM	FTYPE	FV	GETFLAG
	GETFST	HALF	HEX	INCRNO	INMODE	INPUT	INVLD	IOID	IOLIST	IOMODE	ITEM	JAR	LINE
	LINENO	LMCURR	LMINCR	LMSTART	MACRO	MISFLAGS	MSGFLAGS	NEWMODE	NEWNAME	NEWTYPE	NOTFOUND	NOTYPING	NUCON
	NUMFINRD	PACK	PADBUF	PADCHAR	PLIST	PLSTFV	PLSTITEM	PTR1	PTR2	PTR3	RANGE	REGSAV	REGSAVX
	RELPAGES	REPCNT	REPL	RESET	RETURN	RPLIST	R0	R1	R10	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SAVCNT	SAVCWD	SCRFLGS	SCRFLG2	SEQNAME	SERSAV
	SERTSEQ	SERTSW	SIGNAL	SPARES	STACK	STACKAT	STACKATL	STRTNO	SWITCH	TAB	TABLIN	TABS	TEMPTAB
	TIN	TOUT	TRNCNUM	TRUNCOL	TVERCOL1	TVERCOL2	TWITCH	TYPE	TYPFLG	UTILFLAG	VERCOL1	VERCOL2	VERLEN
	WRTYPE	XAREA	XXXCWD	XYCNT	XYFLAG	YAREA	ZONE1	ZONE2					
DMSEDX	ACMSSEG	ADEVTAB	ADMSFREB	ADTM	AEDLIN	AESTATE	AESTATEW	AEXTEND	AFINIS	AFLAGLOC	AFSTFNRD	ALINELOC	ALTMODE
	ANUMLOC	ARDBUF	ASYSNAMS	BALR	BLANK	BLANK1	BLANK2	BLANK3	BLOC	CARDINCR	CASESW	CHNGMSG	CMDBLOK
	CMSSEG	CODE203	CONSOLE	CONWAIT	CORITEM	DCSSAVAL	DCSSFLAG	DCSSLDED	DEC	DEVTAB	DOSFLAGS	DOSSVC	EDCB
	EDCBEND	EDCBLTH	EDLIN	EDRET	EDWORK	ENDBLOC	ENDTABS	FILE	FLAG	FLAGLOC	FLAG2	FMODE	FNAME
	FREELEN	FSTAIC	FSTD	FSTFINRD	FSTFMODE	FSTRECFM	FTYPE	FV	INVLDHDR	IOAD	IOID	IOLIST	IOMODE
	ITEM	JAR	LINE	LINELOC	LMSTART	LOADIT	LOADMOD	LOC	LOCCNT	MAINAD	NUCON	NUMLOC	PADBUF
	PADCHAR	PLIST	PLSTITEM	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	TIN	TRUNCOL	TWITCH	TYPE	TYPSCR	VCFSTLKP	VERCOL1
	VERCOL2	VERLEN	WRTYPE	ZONE1	ZONE2								
DMSERD	AACTFREE	AACTFRET	AACTLKP	ADMSERL	ADMSFREB	ADMSTRKA	ADMSTRKD	ADTADD	ADTADD2	ADTANACW	ADTARES	ADTCHBA	ADTDBSIZ
	ADTDTA	ADTFLG1	ADTFRW	ADTFSTC	ADTLEFT	ADTM	AFTADT	AFTARP	AFTAWP	AFTBFORM	AFTBLKWD	AFTBPRCT	AFTCLB
	AFTDBA	AFTEBDSP	AFTEBLIN	AFTERR8	AFTFLG	AFTFLG2	AFTLSTRC	AFTMXBLK	AFTNEW	AFTOVLAP	AFTPFST	AFTPHYP	AFTRD
	AFTRDBLK	AFTRDID	AFTREAD	AFTSVBLK	AFTSVFP1	AFTSVFP2	AFTSVFP3	AFTSVFP4	AFTUBFAD	AFTUBFLG	AFTUFP1	AFTUFP2	AFTUFP3
	AFTUFP4	AFTUFP5	AFTVLGTH	AFTVLREC	AFTWRT	AFVS	AKILLEX	ARDTK	ATRUNC	AWRTK	BALR	CODE203	DCHCHGD
	DCHDAMAP	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFLG1	DCHFWPTR	DCHPFIXL	DCHSECT	DCHSEQBD	DCHTDISP	DCHTRUNK	DMSERR	DSKADR
	DSKADR2	DSKCHAIN	DSKLOC	DSKLOC2	DSKLST	DSKLST2	DSKPTRSZ	DSKPTRS2	ERROR1	ERROR3	FBLOCK	FFORMAT	FREELOWE
	FVSECT	FWADDR	KF1	KH2	KXFLAG	KXWANT	LOC	MAINHIGH	NUCON	REGSAV0	REGSAV1	REGSAV3	RETURN

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	RWCNT	RWCNT2	RWFSTRG	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3
	R4	R5	R6	R7	R8	R9	SAMELEN	TYPE	UFDBUSY	VBLOCK	VCADTLKP	VCFSTLKP	VCFSTLKW
	VFORMAT	WRBIT	XFF										
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	NUCAPIO	NUCON	OLDPSW	R0	R1	R10	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SSAVE					
DMSERS	AACTFRET	AACTLKP	AACTNXT	ADMSERL	ADMSFREB	ADMSTRKA	ADMSTRKD	ADTADD	ADTANACW	ADTARES	ADTCFST	ADTCHBA	ADTDBSIZ
	ADTEDF	ADTFDA	ADTFLG1	ADTFLG4	ADTFRO	ADTFRW	ADTFSTC	ADTFSTSZ	ADTHBCT	ADTLFST	ADTLHBA	ADTM	ADTSECT
	AFTADT	AFTFLG	AFTFST	AFTPFST	AFTPHYP	AFTRDBLK	AFTUFP1	AFTUFP2	AFTUFP3	AFTUFP4	AFTUFP5	AFTWRT	AFVS
	AKILLEX	AQQTRKX	ARDTK	ASTATEW	ATFINIS	ATRKLKPX	AUPDISK	AWRTK	BALR	CODE203	DCHBWPTR	DCHCHGD	DCHCHOP
	DCHDA	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFLG1	DCHFLG2	DCHFLG4	DCHFWPTR	DCHLHBLK	DCHPFIXL	DCHRSV	DCHSECT	DCHSEQBD
	DCHTDISP	DCHTRUNK	DMSERR	DMSLAD	DMSLADW	DMSLFSW	DSKADR	DSKLOC	DSKLST	D1	D2	D3	ERBIT
	ERRCOD1	ERROR3	ERR1	ERSFLAG	FVSECT	FVSERAS0	FVSERAS1	FVSERAS2	FVSERAS3	FVSERAS4	FVSERAS5	FVSFSTHP	FWADDR
	KXFLAG	KXWANT	LOC	NUCON	REGSAV1	RWCNT	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SIGNAL	STATEFST	STATER1	UFDBUSY
	VCADTLKP	VCFSTLKP											
DMSETR	ASYSCOM	ASYSREF	BGCOM	IJBBOX	LUBPT	NICLPT	NUCON	PUBPT	R0	R1	R10	R11	R14
	R15	R2	R3	R4	R5	R6	R7	SYSCOM					
DMSEXC	ACMSSEG	ADMSFREB	ADTM	ADTSECT	AEXEC	AFINIS	AFSTLKP	AFVS	AOPSECT	ASYSNAMS	BALR	CMSSEG	CODE203
	DCSSAVAL	DCSSFLAG	DCSSLDED	EXADD	EXECFLAG	EXECRUN	EXLEVEL	EXNUM	FFD	FILEBUFF	FILEBYTE	FILEMODE	FSTD
	FSTEPL	FSTLRECL	LOC	MISFLAGS	NEGITS	NUCON	OPSECT	PLIST	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SYSNAMES	SYSNEND
	TYPE												
DMSEXE	ADMSERL	ADMSFREE	AESTATE	AFINIS	ALL	APOINT	ARDBUF	AREA	ARGS	ASCANN	ATTN	BALR	BLANK
	CL	CODE203	COMLINE	DEC	DUMP	DUMPING	ERR$202	ERROR1	ERROR2	ERROR3	EXEC2	FBLOCK	FILE
	FILEMODE	FILENAME	FILETYPE	FLAG	FSCBAITN	FSCBANIT	FSCBBUFF	FSCBD	FSCBEPL	FSCBFLG	FSCBFM	FSCBFN	FSCBFT
	FSCBFV	FSCBITNO	FSCBNOTT	FSCBNORD	FSCBSIZE	LINE	LOC	LOOP	NUCON	PLIST	RANGE	READ	RETCODE
	RETURN	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R8	R9	SAVER14	SKIP	SPARES	STACK	START	SVC$202	TRUNCOL	TYPE	TYPLIN
	VAR	WAITRD	WRITE										
DMSEXI	ADMSFREB	AOPSECT	ARDBUF	BALR	CMNDLINE	CMNDLIST	CODE203	CONRDCNT	DMSEXE	DMSEXT	LOC	NUCON	OPSECT
	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R8
	R9	TYPE											
DMSEXT	ADMSFREB	ADTFDOS	ADTFLG1	ADTFLG2	ADTFRO	ADTFROS	ADTFRW	ADTM	ADTSECT	AEPOINT	AESTATE	AFINIS	AGETCLK
	AOPSECT	ARDBUF	ASCANO	BALR	BLANK	CODE203	CONWAIT	CURRDATE	CURRTIME	DOSDSK	DOSFLAGS	DOSMODE	DOSSVC
	DSKLIN	EIGHT	ENDFREE	ERR$202	EXADD	FF	FLAG	FLAG1	FMODE	FNAME	FSIZE	FSTEPL	KH2

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	LASTCMND	LASTEXEC	LOC	LOOP	MSGFLAGS	NEED	NODISK	NOTYPING	NUCON	NUMFINRD	OPSECT	OSRESET	OSSFLAGS
	PBUFF	PREVCMND	PREVEXEC	READCNT	RETCODE	R0	R1	R10	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SETUP	SKIP	SUBFLAG	SVC$202	TIMBUF	TYPLIN	UNPACK	VCADTLKP
	VCADTLKW	VIPINIT	VSAMFLG1										
DMSFCH	ADMSERL	ADMSFREB	AESTATE	ANCHSIZ	ASYSREF	AUSRAREA	BALR	BGCOM	BLKSIZE	CODE203	COMNAME	CSW	DACTIVE
	DIRAAA	DIRC	DIREEE	DIRLL	DIRN	DIRNAME	DIRPPP	DIRRR	DIRTT	DIRTTR	DOSDD	DOSDEV	DOSFIRST
	DOSFLAGS	DOSKPART	DOSLIBL	DOSOP	DOSOSFST	DOSSECT	DOSSVC	DOSTRANS	DOSVSAM	EIGHT	ERR104	FCBDSK	FCBDSNAM
	FCBINIT	FCBSECT	FREELOWE	FRERESPG	HIPHAS	HIPPOG	IHADEB	INPUT	LOC	LUBPT	MAINHIGH	MAINLIST	MAINSTRT
	NODISK	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	SF	VSAMFLG1	VSAMRUN	VSAMSERV	VSMINSTL			
DMSFET	ABGCOM	ADMSERL	ADMSERR	ADMSFREB	ALDRTBLS	ASYSCOM	AUSRAREA	BALR	BGCOM	BLANK	CODE203	COMNAME	DACTIVE
	DIRN	DIRNAME	DOSCOMP	DOSFLAGS	DOSMODE	DOSRC	DOSSVC	ERR1	FCHAPHNM	FCHLENG	FCHOPT	FCHTAB	HIPHAS
	IJBFTTAB	LASTLOC	LOC	LOCCNT	NOTEXT	NUCON	PNOTFND	RETURN	R0	R1	R12	R14	R15
	R2	R3	R4	R5	R6	R7	START	STRTADDR	SYSCOM	TBENT	VSMINSTL		
DMSFLD	ABATABND	AESTATE	ASTATE	BATDCMS	BATFLAGS	BATFLAG2	BATRUN	BLANK	BLKSIZE	CLEAR	CLPAREN	CLR	CONREAD
	CONVERT	CURRSAVE	DENSITY	DUMMY	D1	D2	EGPR0	ERR70E	FCBBLKSZ	FCBBLP	FCBCASE	FCBCATLD	FCBCATML
	FCBCON	FCBDD	FCBDEV	FCBDOSL	FCBDSK	FCBDSMD	FCBDSNAM	FCBDSORG	FCBDSTYP	FCBDUM	FCBEND	FCBENSIZ	FCBFIRST
	FCBINIT	FCBIOSW	FCBLABPT	FCBLABT	FCBLEAVE	FCBLRECL	FCBMEMBR	FCBMODE	FCBNEXT	FCBNL	FCBNOEOV	FCBNSL	FCBNSLMD
	FCBNSLNM	FCBNUM	FCBOFF	FCBOSDSN	FCBPCH	FCBPOS	FCBPROC	FCBPTR	FCBRDR	FCBRECFM	FCBSECT	FCBSL	FCBTAP
	FCBTAPID	FCBTPSW	FCBXTENT	FILE	FLAG1	FLAG2	FLAG3	JFCBIND2	JFCBUFNO	JFCKEYLE	JFCLIMCT	JFCOPTCD	LABDEXD
	LABDVID	LABFCBPT	LABFDEF	LABFILE	LABFIRST	LABFLAG1	LABFLAG2	LABNEXT	LABNUM	LABSECT	LABSIZE	LABVOLID	LOC
	LOOP	MATCH	NUCON	PACK	RESET	RETURN	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SCAN	SCANNING	SSAVE	STATLST
	TRTVOLID	TYPE	XFF										
DMSFNC	ATTN	CONREAD	CONWAIT	DMSABNSV	DMSBWR	DMSCAT	DMSCATMK	DMSCATNB	DMSCIOSI	DMSCITDB	DMSCITDK	DMSCPF	DMSCRD
	DMSCWR	DMSCWT	DMSDBG	DMSERR	DMSEXC	DMSFET	DMSFREB	DMSFREES	DMSFREEX	DMSFRES	DMSFRETS	DMSFRETX	DMSITET
	DMSITSK	DMSITSSB	DMSITSXS	DMSLADAD	DMSLDRA	DMSLOA	DMSMOD	DMSPIO	DMSPIOCC	DMSPIOSI	DMSPNTE	DMSSTGAT	DMSSTGCL
	DMSSTGSB	DMSSTGSV	DMSSTTN	DMSSTTNW	DMSTLABL	DMSTLB	DMSVSR	EPOINT	FINIS	LOAD	LOADMOD	LOC	RDBUF
	RETURN	START	TYPLIN	WAITRD	WRBUF								
DMSFNS	AACTFRET	AACTLKP	ADIOSECT	ADMSERL	ADMSFREB	ADTADD	ADTANACW	ADTDBSIZ	ADTDTA	ADTEDF	ADTFLG1	ADTFLG3	ADTFLG4
	ADTFRO	ADTFSTC	ADTFTYP	ADTFUPD1	ADTFXCHN	ADTLHBA	ADTNACW	ADTRES	ADTSECT	ADTXNREC	AERASE	AFTADT	AFTARP
	AFTAWP	AFTCLA	AFTCLB	AFTCLD	AFTCLDX	AFTCLN	AFTCLX	AFTDBA	AFTDBD	AFTFBA	AFTFCLA	AFTFCLX	AFTFLG
	AFTFLG2	AFTFST	AFTFULD	AFTNEW	AFTPFST	AFTPHYP	AFTRD	AFTRDBLK	AFTUFP1	AFTUFP2	AFTUFP3	AFTUFP4	AFTUFP5
	AFTUSED	AFTWRT	AFVS	AKILLEX	ALL	AQQTRKX	ARDTK	ATRKLKPX	ATYPSRCH	AUPDISK	AWRTK	BALR	BALRSAVE
	CLKVALMD	CODE203	DATIPCMS	DCHCHGD	DCHDATA	DCHDWSIZ	DCHFLG1	DCHSECT	DCHTDISP	DCHTRUNK	DEVTYP	DIOCSW	DIOSECT
	DISK$SEG	DMSERR	DMSLFSW	DSKADR	DSKLOC	DSKLST	FBACD1	FBACL1	FINISLST	FNBIT	FVSECT	FVSL1	FVSPATCH
	FWADDR	HEX	KXFLAG	KXWANT	LOC	MCKOPSW	NUCON	QQDSK1	REGSAV3	RWCNT	RWFSTRG	R0	R1
	R10	R11	R12	R13	R14	R15	R2	R5	R6	R7	R8	R9	SAVEADT
	SECTNUM	SEEKADR	SENSB	STATEFST	STATFST2	SUBFLAG	SUBINIT	TYPE	UFDBUSY				

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSFOR	ABLKIND	ADEVIND	ADEVTAB	ADMSALU	ADMSFREB	ADMSLADN	ADMSTRKD	ADTAMP1	ADTAMP2	ADTARES	ADTCYL	ADTDBSIZ	ADTDCRED
	ADTDFP1	ADTDIOA	ADTDIOB	ADTDTA	ADTEDF	ADTEDFAE	ADTFALUF	ADTFBABF	ADTFBALB	ADTFDA	ADTFDOS	ADTFFSTF	ADTFLG1
	ADTFLG2	ADTFLG4	ADTFQQF	ADTFRO	ADTFROS	ADTFRW	ADTFSTC	ADTFSTSZ	ADTHBCT	ADTID	ADTIDENT	ADTLABSZ	ADTLAST
	ADTLEFT	ADTLFST	ADTLHBA	ADTM	ADTMCYL	ADTMSK	ADTNFST	ADTNUM	ADTPQM1	ADTPQM2	ADTPQM3	ADTQQM	ADTRES
	ADTSECT	ADTUSED	ADT1ST	AFINIS	AFVS	ALABELRD	ALABELWR	ATBLIND	AUPDISK	BALR	BLKSIZE	CODE203	DCHBWPTR
	DCHDAMAP	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFLG1	DCHFWPTR	DCHPFIXL	DCHSECT	DCHSEQBD	DCHTDISP	DCHTRUNK	DEVADDR	DTAD
	D1	D2	D3	D6	EDF110	EDF120	EDF180	ERR1	FBACD1	FBACL1	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	START	STATLST	TYPE	WAITRD
	XPF												
DMSFRE	AABNGO	ABNPSW	ABNREGS	ABWSECT	ACALL	ADMSERL	AFREETAB	ASVCSECT	AUSRAREA	BALR	BATFLAGS	BATLOAD	BLOCKLEN
	CALLER	CL	CODE203	CURRSAVE	DMSABW	DMSALU	DMSERR	DMSFRT	FINIS	FLAGS	FLCLN	FLHC	FLNU
	FLPA	FRDSECT	FREEFLG1	FREEFLG2	FREEHN	FREEHU	FREELN	FREELOWE	FREELOWR	FREELOW1	FREELU	FREESAVE	FRF1B
	FRF1C	FRF1E	FRF1H	FRF1L	FRF1M	FRF1N	FRF1V	FRF2CKE	FRF2CKT	FRF2CKX	FRF2CL	FRF2NOI	FRF2SVP
	LOC	LOCCNT	MAINHIGH	MAX	MAXCODE	NOPAGREL	NUCCODE	NUCKEY	NUCON	NUM	OPTFLAGS	POINTER	PRFPOFF
	PROTFLAG	RETURN	SEGORELO	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	AESTATE	AFINIS	ARDBUF	DOSLBSV	DOSLIBL	FILE	FSTEPL	LOC	LOOP	MACLBSV	MACLIBL	NUCLDLIB	NUCLODSV
	NUCON	RETURN	R0	R1	R11	R12	R13	R14	R15	R2	R3	R4	R5
	R7	R8	TOTLIBS	TXLIBSV	TXTDIRC	TXTLIBS							
DMSGND	ADTFSTSZ	ADTSECT	AESTATE	AFVS	ALDRTBLS	DIRNAME	D1	FILE	FSTAIC	FSTBLKCT	FSTD	FSTDATEW	FSTFMODE
	FSTFOP	FSTLRECL	FSTNLVL	FSTRECCT	FSTRECFM	FVSECT	FVSFSTAC	FVSFSTAD	NUCON	NUM	R0	R1	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R9	TBENT	
DMSGRN	DUMMY	EXECRUN	FF	FORM	INPUT	OUTPUT	PARM	PARMLIST	RETURN	RUN	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SAVEAREA	START											
DMSHDI	ADMSFREB	AIOSECT	ANUCEND	AUSRILST	AUSRITBL	BALR	CODE203	DOSFLAGS	DOSSVC	ERRCODE	F256	IONTABL	IOSECT
	LOC	NUCON	RETURN	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	RETURN	R0	R1	R10	R12	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R8	R9	SVCSECT	VMSIZE							
DMSHLB	ADMSHLB	ASERR	BLANK80	BUFF1	BUFF1LGZ	BUFF1M1	BUFF2	BUFF2LGZ	BXBUFF	BXBUFFND	B1LG	COFF	ERINDEX
	ERM05	FFIFTEEN	FFCUR	FNINE	HLPSECT	HTEN	MAXADDR	NOCHARS	REGSAVE2	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R5	R6	R7	R8	R9	SWITCH	VBOX
	VRULEREM												

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSHLD	HLPCTA	R13											
DMSHLE	AMAIN	AQUICKEX	ASWRTMSG	ATRUEEND	AWNGRET	CL	ERENTRY	ERINDEX	ERRBF	ERRLG	ERRORNN	FFIFTEEN	HLPSECT
	LINECNT	MSGNTAB	RETCODE	R1	R13	R14	R15	R2	R3	R4	R7	R9	SAVEAREA
	TBUF	TBUFLGZ											
DMSHLI	AABBREV	AGCLOSE	AGOPEN	ALINKINT	ALL	ALTFILN1	ALTFILN2	ALTFILT1	AMAIN	APARMRET	APSTATE	ASERR	ASWRTPRP
	ATRUEEND	ATTRELO	AUXDIR	AWNGRET	BLANK80	BUFF1M1	BUFF2M1	BXBUFFND	CSINCLUD	CS1	CS2	CS3	DBSWS
	DESC	DMSHLD	ENTSIZ	ERINDEX	ERM09	ERM12	ERM13	ERM15	ERRBF	ERRLG	EXBUF	EXC2	EXEC2
	FILNAM	FILTYP	FORM	F4096	HLPSECT	LINKBACK	LINKBEG	LINKDOWN	LINKELEM	LINKFOR	LINKNEXT	LINKSIZE	LINKSTAR
	LOC	MENU	NEWBUF	NUCON	OUTPARM	PARM	PTCHLOC	QUITSWS	REGSAVE	REGSAVE2	RESET	RETCODE	R0
	R1	R10	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	STACK	STATEID	STCKLEN	STORAGE	SWITCH	TAB	TABBGN					
DMSHLL	ACMSSEG	ASYSNAMS	CMSSEG	DCSSAVAL	DCSSFLAG	DCSSLDED	D2	LASTLMOD	LOADIT	LOADMOD	MSGFLAGS	NOTYPING	NUCON
	RESET	R0	R1	R10	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	STACK	STRTADDR	SUBACT	SUBFLAG	SYSNAMES	SYSNEND					
DMSHLP	ADMSHLB	AGREAD	ALINKPUT	AMERGE	ASP	ASWRTIO	ATRUEEND	AUGSW	AVRPUT	BLANK80	BUFF1	BUFF1LGZ	BUFF1M1
	BUFF2	BUFF2LG	BUFF2LGZ	BUFF2M1	B1LG	B2LG	CCS	COF	CON	CSSAVE	CSSWS	CVBSAVE	DECM
	DEF1IN7	DELTA7	EIGHTOF	ERM02	ERM03	ERM04	ERM05	ERM06	ERM07	ERM11	ERRBF	ERRLG	FEIGHT
	FF	FFIFTEEN	FNONE	FTHREE	FTWO	FZERO	HLPSECT	HTEN	HZERO	INDEXS	INDL	LINECNT	LINKBACK
	LINKCHAR	LINKDOWN	LINKELEM	LINKFOR	LINKMULT	LINKPARM	LINKSIZE	LINKSTAR	LLZ	MATCH	MAXADDR	MULTPNTR	NFSWS
	OFFL	OFFLI	OLDCOUNT	PARMLIST	PARMPUT	PBUFF	PBUFFCT	PBUFFSZ	PERSU	PLONE	PRSAVE	PTCHLNG	PTCHLOC
	PTELLNG	RMARGIN	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SAVEAREA	STCKCNT	SUBR	SUOFFSW	SUONSW	SUSAVE	SUSAVE2	SUT1A
	SUT2	SWITCH	TABBGN	TBUF	TRANTAB	TRSWS	TSCB	TSCBLENG	UNDL	VRULE	VRULEREM		
DMSHLS	AADTLKP	ADMSHLS	ADTEDF	ADTFLG1	ADTFLG4	ADTFRW	ADTM	ASERR	ATRUEEND	ATTCHFST	ATTLAD	AUGSW	BUFF1
	BUFF2	BUFF2LG	CSSAVE	DBSWS	DECM	DIRBUF	DIRBYTES	DIRDISK	DIRREC	DIRSIZ	DIRTYP	DSKLST	D1
	EDFCHAR	EDFSW	EOFSW	ERINDEX	ERM01	ERM08	ERM13	ERM14	ERRBF	ERRLG	FILNAM	FILNUM	FONE
	FSTD	FSTFMODE	GOFNAM	GOFTYP	HELPFST	HLPSAVE	HLPSECT	HONE	HTWELVE	HTWENTY	LINSEQNO	LLZ	LOC
	MSG	NUCON	OPENSW	PERMSW	PLONE	QUITSWS	RETCODE	R0	R1	R10	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	STACK	STARS4	STATCOPY	STATEID
	STCKCNT	STCKLEN	SWITCH	TBUF	TBUFLGZ	YEXTS	YSRCH						
DMSIFC	AADTLKW	ADTM	ADTSECT	BSF	CLR	COMPSWT	CURRSAVE	DMSREA	DOSFLAGS	DOSSAVE	DOSSVC	D1	D2
	EGPR15	FILE	FORM	FSCBBUFF	FSCBD	FSCBFM	FSCBFN	FSCBFV	FSF	IOBECB	LOC	MODNAME	NUCON
	OLDPSW	OSSFLAGS	RESET	REW	R0	R1	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SAVEAREA	SAVER0	SAVER1	SAVER14	SAVER15	SAVE2	SSAVE	TAP1
	TXTDIRC	TXTLIBS	TYPE										
DMSIMA	FILE	MAINHIGH	MODNAME	NUCON	RETURN	R0	R1	R10	R12	R14	R15	R2	R3
	R4	R5	R6	R7	R8	R9							

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSINA	AUSABRV	BALRSAVE	ERROR1	ERROR2	ERR1	LOADMOD	NOABBREV	NOSTDSYN	NUCON	OPTFLAGS	R0	R1	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	TYPE			
DMSINI	ABLKIND	ADEVIND	ADEVSUP	ADEVTAB	ADTCYL	ADTDBSIZ	ADTIDENT	ADTMCYL	ATBLIND	CAW	CC	CE	CHANO
	CONSOLE	CSW	DE	DEVTAB	DMSDBGP	DMSINS	DMSINSE	DMSITS1	D1	D2	D3	D6	EXTNPSW
	FBACCWL1	FBADEF	FBADWDT	FBAIPL	FBALOC	FBALWDT	FBARD	FBAWR	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	SKIP	SYSADDR	SYSTEMID	TIC	WAIT	WRDATA	WRITE	WRITE1	YDISK			
DMSINM	ASUBSECT	BALRSAVE	CURRCPUT	CURRDATE	CURRVIRT	NUCON	R0	R1	R10	R14	R15	R2	R3
	R4	R5	R8	SUBSECT	TIMBUF								
DMSINS	AABNGO	AABNSVC	ABGCOM	ACMSCVT	ACMSSEG	ACMSZER	ADMPEXEC	ADMSABN	ADMSALU	ADMSCPF	ADMSFREB	ADMSLIO	ADMSPIOC
	ADMSVIB	ADTFDA	ADTFFSTF	ADTFFSTV	ADTFLG1	ADTFLG3	ADTFORCE	ADTFSORT	ADTFSTC	ADTFSTSZ	ADTIDENT	AEPOINT	AESTATE
	AEXTSECT	AKILLEX	ALDRTBLS	ALL	ANUCEND	AOPSECT	AOSMODL	APOINT	AQQTRK	AQQTRKX	AREA	ASORTFST	ASSTAT
	ASSTATX	ASSTATZ	ASTATEXT	ASYSNAMS	ASYSREF	ATLBMODL	AUSRAREA	AVSREOJ	AYSTATX	AYSTATZ	BALR	BATFLAGS	BATFLAG2
	BATIPLSS	BATLOAD	BATRUN	BGCOM	BLANK	CAW	CC	CHANO	CLKVALMD	CMNDLINE	CMNDLIST	CMSCVT	CMSSEG
	CMSZER	CODE203	CONRDCNT	CONRDCOD	CONREAD	CURRDATE	CVTAVIB	CVTMDL	CVTMZ00	CVTNUCB	CVTOPTA	CVTSECT	DATIPCMS
	DCHBWPTR	DCHDATA	DCHDTSIZ	DCHDUM	DCHDWSIZ	DCHFLG2	DCHFWPTR	DCHPFIXL	DCHSECT	DCSSAVAL	DCSSFLAG	DCSSOVLP	DCSSVTLD
	DMSALU	DMSDBG	DMSFRES	DMSLAD	DMSLOA	DMSSCNN	DMSZER	DMSZEX	DTAD	D1	D2	D3	EXTNPSW
	EXTSECT	FBACD1	FBACL1	FREELOWE	FRERESPG	FVS	F0	GRAFDEV	H4096	IONPSW	IPLADDR	IPLPSW	LOC
	LOCCNT	MAINHIGH	MCKM	MISFLAGS	MODFLGS	MSGFLAGS	NOVMREAD	NUCAPIO	NUCCOPYR	NUCON	OPSECT	OPTFLAGS	OSMODLDW
	PGMNPSW	PRFTSYS	PROTFLAG	REGSAV	R0	R1	R10	R11	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SAVEXT	SEGORELO	SILI	SPECLF	SYSLOAD	SYSNAME
	SYSNAMES	SYSNEND	SYSREF	SYSTEMID	TBENT	TIMCHAR	TIMER	TIMINIT	TYPE	VCADTLKP	VMSIZE	WAIT	YYDDD
DMSINT	AACTLKP	ADMSCPF	ADMSFREB	ADTEDF	ADTFLG4	ADTSECT	AEPOINT	AEXTSECT	AFINIS	AFTADT	AFTARP	AFTAWP	AFVS
	AIOSECT	AOPSECT	ASCBPTR	ASTGSB	ASUBFST	ASUBRET	ASUBSECT	ASUBSTAT	ASVCSECT	ASYSNAMS	AUSRAREA	BALR	CMNDLINE
	CMSSEG	CMSTIM	CODE203	CONRDCNT	CONRDCOD	CONREAD	CONWAIT	CONWRBUF	CONWRCOD	CONWRITE	DCSSFLAG	DCSSJLNS	DCSSLDED
	DMSDBG	DMSLFS	DMSSCNN	DOSFLAGS	DOSMODE	DOSSVC	ERRNUM	EXTPSW	EXTSECT	FILENAME	FILETYPE	FINISLST	FREELOWE
	FSTFINRD	FVSECT	FVSPATCH	IONTABL	IOSECT	JNUMB	LASTCMND	LOC	LOCCNT	MISFLAGS	MSGFLAGS	NEGITS	NOABBREV
	NOIMPCP	NOIMPEX	NOPAGREL	NORDYTIM	NOTYPING	NOVMREAD	NUCON	NUCSCBLK	OPSECT	OPTFLAGS	OSRESET	OSSFLAGS	PLIST
	PREVCMND	QSWITCH	REDERRID	RELPAGES	RMSGBUF	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R9	SCBFWPTR	SCBLOCK	SCBLOCKD	SPECLF	SPIESAV	STAESAV
	STARS	STATEFST	SUBACT	SUBFLAG	SUBREJ	SUBRTN	SUBSECT	SVCSECT	SWTCHSAV	SYSNAMES	SYSNEND	TIMCHAR	TIMER
	TIMINIT	TYPE	VIPINIT	VSAMFLG1									
DMSIOW	AEXTSECT	CSW	DBGEXINT	DBGFLAGS	DEVICE	DMSDBG	EXTFLAG	EXTSECT	IONPSW	IOOPSW	NUCON	REALTIMR	RETURN
	R0	R1	R10	R11	R14	R15	R2	R4	R5	R6	R7	R8	R9
	TIMCHAR	TIMER	TIMINIT	WAITSAVE									
DMSITE	ABATABND	ABATLIMT	ADBGSECT	ADMSCWR	ADMSFREB	ADMSITI	AEXTSECT	AFVS	ARGS	ASVCSECT	BALR	BATCPUC	BATCPUL
	BATFLAGS	BATFLAG2	BATLOAD	BATLSECT	BATRUN	BATUSEX	BATXCPU	BATXLIM	CMSTAXE	CODE203	CONHCT	CSW	DBGEXEC
	DBGEXINT	DBGFLAGS	DBGOUT	DBGSECT	DECDEC	DMSDBG	DOSFLAGS	DOSSVC	EXSAVE	EXSAVE1	EXTFLAG	EXTOPSW	EXTPSW
	EXTRET	EXTSECT	FVSECT	F0	F2	F4	F6	INPUT	IONPSW	IOOPSW	JR1	LINE	LOC

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	LOOP	MVCNT1	NUCON	NUMPNDWR	OSSFLAGS	OSWAIT	OVSTAT	PENDREAD	REALTIMR	RESET	RETURN	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	TSOATCNL	TSOFLAGS
	TYPE	TYPLIST	UFDBUSY	WAIT	XPSW								
DMSITI	AABNGO	ABNPSW	ABNREGS	ABWSECT	ADIOSECT	AFVS	AIOSECT	ATTNHIT	CMSTAXE	CSW	DEVICE	DIOSECT	DMSABW
	FBACD1	FBACL1	FVSECT	HOLD	IONTABL	IOOLD	IOOPSW	IOPSW	IOSAVE	IOSECT	KXFLAG	KXWANT	MISFLAGS
	NEXTO	NUCON	OLDEST	QQDSK1	RETURN	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							
DMSITP	AABBREV	ABNERLST	ABNPSW	ABNREGS	ABWSECT	ADMSABW	ADMSERR	ADMSFREB	AFVS	ALTASAVE	APGMSECT	APPSAVE	ASYSREF
	ATCBPTR	AUPIE	BALR	BGCOM	CALLEE	CODE203	CURRSAVE	DMSABNGO	DOSFLAGS	DOSMODE	DOSSVC	FVSECT	INTINFO
	LOC	LTK	NUCON	OPSW	PGMNPSW	PGMOPSW	PGMSECT	PIBPT	PICADDR	PIE	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	TCBABPTR	TCBADR	TCBPCPTR	TCBSAVE
	TPFUSR	TYPE	TYPFLAG	UFDBUSY	VSAMFLG1	VSAMSERV							
DMSITS	AABNGO	ABNPSW	ABNREGS	ABWSECT	ACMSSEG	ADMSABW	ADMSERL	ADMSFREB	ADOSDCSS	AERR	AFVS	AOSMODL	ARDBUF
	ASVCSECT	ASYSNAMS	AWAIT	AWRBUF	BALR	CALLEE	CALLER	CHKWRD1	CHKWRD2	CMSSEG	CODE	CODE203	CURRSAVE
	DCSSAVAL	DCSSFLAG	DCSSLDED	DCSSVTLD	DMSCWT	DMSERR	DMSFNC	DMSFNC3	DMSMOD	DOSFLAGS	DOSSVC	EFPRS	EGPRS
	EGPR0	EGPR11	EGPR15	EGPR2	ERRET	FLAGS	FVSECT	F0	F6	GPRLOG	ITSBIT	KEYMAX	KEYP
	KEYS	KXFLAG	KXWANT	KXWSVC	LASTTMOD	LENOVS	LOC	MCKM	MISFLAGS	NEGITS	NRMRET	NUCON	NUCSCBLK
	OLDPSW	OVSECT	PRFPOFF	PRFTSYS	PRFUSYS	PROTFLAG	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SCBENTR	SCBFWPTR	SCBLOCK	SCBLOCKD
	SCBNAME	SCBPSW	SCBWKWRD	SEGORELO	SSAVE	SSAVENXT	SSAVEPRV	SSAVESZ	START	STRTADDR	SVCOPSW	SYSNAMES	SYSNEND
	TPFERT	TPFNS	TPFR01	TPFSVO	TPFUSR	TSOATCNL	TSOFLAGS	TYPE	TYPFLAG	UFDBUSY	USAVE	USAVEPTR	USAVESZ
	XFF												
DMSLAB	ADTID	DEVSECT	DOSBUFSP	DOSCMS	DOSDD	DOSDEV	DOSDSMD	DOSDTF	DOSDUM	DOSEXTCX	DOSEXTNO	DOSEXTTB	DOSFIRST
	DOSINIT	DOSNEXT	DOSNUM	DOSOSDSN	DOSSECT	DOSTYPE	DOSUCNAM	DOSVOLNO	DOSVOLTB	DOSYSXXX	DTFFLG2	DTFINPUT	DTFSD
	DUMMY	LABBUFSP	LABCONV	LABDSN	LABED	LABEXN	LABEXT	LABFID	LABFNAME	LABFSER	LABIND	LABLAST	LABLUBA
	LABOMIT	LABOPCOD	LABREC	LABSEQ	LABST	LABSTBK	LABSW	LABTYP	LABUCNAM	LABVOL	LABVSEQ	LAB64K	LHH
	NOTFOUND	NUCON	OSADTDSK	OUTPUT	RESET	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	VCADTLKP				
DMSLAD	ADMSFREB	ADTADDED	ADTBWPTR	ADTDBSIZ	ADTDTA	ADTEDF	ADTFDA	ADTFFSTV	ADTFLG1	ADTFLG2	ADTFLG4	ADTFRO	ADTFROS
	ADTFRW	ADTFSTSZ	ADTFVS	ADTHBCT	ADTLB	ADTLD	ADTLEFT	ADTM	ADTMX	ADTNFST	ADTPSTM	ADTPTR	ADTRES
	AFVS	ASVCSECT	BALR	CODE203	DCHBWPTR	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFWPTR	DCHPFIXL	DCHSECT	ERROR1	FSTAIC
	FSTBLKCT	FSTD	FSTDATEW	FSTFMODE	FSTFNAME	FSTFOP	FSTLRECL	FSTNLVL	FSTRECCT	FSTRECFM	FVSECT	IADT	LOC
	NUCON	REGSAV0	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SVCSECT	SVLAD	SVLADW	TYPE				
DMSLAF	ADMSFREB	ADTFLG1	ADTFRW	ADTM	ADTMX	ADTSECT	AFTADT	AFTFLG	AFTFSF	AFTFST	AFTLD	AFTPFST	AFTPTR
	AFTSTART	AFTUSED	BALR	CODE203	LOC	NUCON	R0	R1	R11	R12	R13	R14	R15
	R2	R3	R4	R5	TYPE								

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSLBD	CLPAREN	CONREAD	ERR70E	FCBDD	FCBDEV	FCBFIRST	FCBLABPT	FCBLABT	FCBNEXT	FCBNUM	FCBSECT	FCBSL	FCBTAP
	LABCRD	LABDEXD	LABEXD	LABFCBPT	LABFDEF	LABFID	LABFILE	LABFIRST	LABFLAG1	LABFLAG2	LABFSEQ	LABGENN	LABGENV
	LABNEXT	LABNUM	LABPERM	LABSEC	LABSECT	LABSIZE	LABVOLID	LABVSEQ	LOC	NUCON	R0	R1	R10
	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	S202
	TRTVOLID	TYPE	XFF										
DMSLBM	AACTLKP	AADTLKP	AADTLKW	ADTEDF	ADTFLG1	ADTFLG4	ADTFRO	ADTFRW	ADTM	ADTSECT	AFTAWP	AFVS	AROUND
	ATFUNC	DIRITEM	DIRSECT	DOUBLE	EIGHT	ERRCODE	FILE	FLAGS	FLAGS2	FREELOWE	FVSECT	FVSFSTAD	INBUFF
	INFV	INITNO	INMODE	INNAME	INNOIT	INSIZE	INTYPE	LIBDIR	LIBDIRSZ	LIBIDENT	LIBSECT	MISFLAGS	NUCON
	OUTBUFF	OUTCOMM	OUTFV	OUTITNO	OUTMODE	OUTNAME	OUTNOIT	OUTSIZE	OUTTYPE	PLIST	PREVIOUS	RELPAGES	RESET
	RETURN	R0	R1	R10	R12	R13	R14	R15	R2	R3	R4	R5	R6
	R7	R8	R9	SCAN	SCAN2	SUBR							
DMSLBR	ADMSERL	NOTFOUND	NUCON	R0	R1	R12	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R9										
DMSLBT	AADTLKP	AADTLKW	ADTARES	ADTDBSIZ	ADTEDF	ADTFLG1	ADTFLG4	ADTFRO	ADTFRW	ADTSECT	ARDBUF	AWRBUF	DIRITEM
	DOUBLE	ENDFREE	EOFCHK	FILE	FINIS	FLAGS	FLAGS2	FSTAIC	FSTD	FSTEPL	FSTFMODE	FSTLRECL	FSTRECFM
	MISFLAGS	MODESET	NODISK	NOLIBE	NOTFOUND	NUCON	RADD	RDBUF	RELPAGES	RESET	RETCODE	RITEM	R0
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	SCAN	TYPLIN									
DMSLCK	ALOKTABA	ALOKTABE	ALOKTB	D1	D3	LASTUSED	LOC	LOCKTAB	NUCON	R0	R1	R11	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	XFF		
DMSLDR	ACMSRET	ADMSFREB	ADMSLIO	AESTATE	AFINIS	ALDRTBLS	AOSMODL	APRILB	APSV	ARDBUF	AROUND	ASCANN	AUSRAREA
	BALR	BATFLAGS	BATLOAD	BRAD	CALLEE	CHKTYPE	CLOSELIB	CMD	CMNDLIST	CODE203	COMMONEX	CONWAIT	CRDPTR
	CURRSAVE	C12	C7	C9	DMSLGTA	DMSLGTB	DMSLIB	DMSLSBA	DMSLSBB	DMSLSBC	DMSLSBD	DMSLSY	DMSSTGSB
	DOSCOMP	DOSFLAGS	DOSMODE	DOSRC	DOSSVC	DYLD	DYNAEND	EGPR1	ENDCDADR	ENTADR	ENTNAME	ESD1ST	ESIDTB
	EXEC2	FINIS	FLAGS	FLAG1	FLAG2	FLAG3	FREELOWE	FRSTSDID	FSTEPL	FSTXTADR	FTYPE	GPRSAV	LDRADDR
	LDRFLAGS	LDRRTCD	LDRST	LOC	LOCCNT	LOCCT	LUNDEF	MAINHIGH	MEMBOUND	MODFLGS	NEED	NOAUTO	NODUP
	NOINV	NOLIBE	NOREP	NOSLCADR	NUCON	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	TMPLOC	TPFUSR	TXTDIRC	TYPFLAG	UNRES	VMSIZE		
DMSLDS	ADMSROS	ADTCYL	ADTFLG1	ADTFLG2	ADTFRO	ADTFROS	ADTFRW	ADTID	ADTIDENT	ADTM	ADTSECT	BLANK	CHKMODE
	CONVERT	CSW	DOSFLAGS	DOSSVC	EIGHT	FCBIOSW2	FCBMEMBR	FCBMVPDS	FCBOSDSN	FCBSECT	FMODE	HALF	LOC
	LOOP	NUCON	OSADTDSK	OSADTVTA	OSADTVTB	PO	POU	RESET	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	VCADTLKP	VCADTNXT
DMSLFS	ADMSFREB	ADMSROS	ADTARES	ADTCFST	ADTCHBA	ADTDBSIZ	ADTDFP1	ADTDFP2	ADTDFP3	ADTEDF	ADTFDA	ADTFFSTV	ADTFLG1
	ADTFLG2	ADTFLG3	ADTFLG4	ADTFRO	ADTFROS	ADTFRW	ADTFSORT	ADTFSTSZ	ADTFTYP	ADTHBCT	ADTLFST	ADTLHBA	ADTM
	ADTMX	ADTPSTM	AFVS	ASVCSECT	BALR	CODE203	DCHBWPTR	DCHDAMAP	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFLG1	DCHFWPTR

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	DCHPFIXL	DCHSECT	DCHSEQBD	DCHTDISP	DCHTRUNK	DISK$SEG	DMSLAD	DMSLADN	DMSSTTR	D3	EDF110	EDF120	EIGHT
	ERROR2	FVSECT	NOTFOUND	NUCON	REGSAVO	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	D3	FILE	FMODE	FNAME	FSCBEPL	FSCBFV	FTYPE
	LDRST	LOC	NUCON	OUTBUF	RADD	READBUF	REPL	RFIX	RITEM	RLENG	RNUM	R0	R1
	R10	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SPEC	TXTDIRC	TXTLIBS	TYPE									
DMSLIB	ADMSFREB	AEPOINT	AESTATE	AFINIS	APSV	BALR	CLOSELIB	CODE203	DEC	DIRITEM	DIRITEMX	DIRSECT	DMSLDRD
	DYMBRNM	FILE	FINIS	FLAGS	FLAG2	FMODE	FNAME	FTYPE	LDRST	LOC	NOAUTO	NOLIBE	NUCON
	NUMBYTE	OSSFLAGS	OUTBUF	RADD	READBUF	RETURN	RITEM	RLENG	RNUM	RRDPT	RWRPT	R0	R1
	R11	R12	R13	R14	R15	R5	R7	SETLIB	SETUP	SPEC	TBLCT	TBLREF	TXTDIRC
	TXTLIBS	TYPE											
DMSLIO	ADMSERR	AERASE	AFINIS	ALIASENT	APSV	AWRBUF	DSKAD	DSKLIN	DYLD	FILE	FLAG1	FLAG2	FNAME
	FSTEPL	LDRADDR	LDRST	LINE1	NOERASE	NOMAP	NUCON	OSSFLAGS	OUTBUF	OUTPUT	PACK	PARMLIST	PLISTSAV
	RETURN	R0	R1	R10	R11	R13	R14	R15	R2	R3	R4	TYPE	TYPEAD
	TYPLIN	UNPACK											
DMSLKD	AADTLKW	ADTM	CLR	FILE	LOOP	MISFLAGS	MODNAME	NOPRINT	NOTERM	NUCON	RELPAGES	RETURN	R0
	R1	R10	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7	R9
	SYSUT1												
DMSLLU	ADTFLG1	ADTFLG3	ADTFRW	ADTFRWOS	ADTSECT	AERASE	AFINIS	ALL	ASYSREF	AWRBUF	BGCOM	BLANK	BUFF1
	BUFF2	DEVTYP	DOSFLAGS	DOSMODE	DSKLST	FINIS	FSTEPL	LUBPT	NICLPT	NOPRINT	NUCON	PUBADR	PUBCUU
	PUBDEVT	PUBDSKM	PUBPT	RETURN	R0	R1	R10	R11	R12	R14	R15	R2	R3
	R4	R5	R6	R7	R8	TAPE	VCADTLKP						
DMSLOA	ALDRTBLS	AUSRAREA	DMSLDRB	FSTXTADR	LDRADDR	LDRFLAGS	LOCCNT	MAINHIGH	NOAUTO	NOERASE	NOINV	NOLIBE	NOMAP
	NOREP	NUCON	PRHOLD	RETURN	R0	R1	R12	R14	R15	R2	R6	STRTADDR	SUBACT
	SUBFLAG	SYSREF	TBENT	TYPE	UNRES								
DMSLOS	ACMSCVT	FCBBUFF	FCBBYTE	FCBDD	FCBDSNAM	FCBFIRST	FCBINIT	FCBITEM	FCBNEXT	FCBOPCB	FCBSECT	FCHLENG	IHADEB
	LOAD	NUCAFCHS	NUCCBLKS	NUCGLOBL	NUCLDLIB	NUCON	NUCOSFLG	NUCOSRLD	NUCOSRUN	NUCSYSDF	NUCTIEIN	PLIST	PO
	PS	RETURN	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R5
	R6	R7	R8	R9	TYPE	UND							
DMSLSB	ADMSFREB	APSV	AUSRAREA	BALR	BATFLAGS	BATLOAD	BRAD	CLEAROP	CODE203	DMSLDRC	DMSLDRD	ENDCDADR	ENTNAME
	FLAGS	FLAG1	FLAG2	FREELOWE	FRSTSDID	FSTXTADR	LASTTMOD	LDRST	LOC	LOCCT	LOOP	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	ADTEDF	ADTFDA	ADTFLG1	ADTFLG2	ADTFLG4	ADTFRO	ADTFROS	ADTFRW	ADTFSTSZ	ADTID	ADTM	AERASE	BRAD
	DCHDATA	DCHDTSIZ	DCHFWPTR	DCHSECT	DEC	ERR1	FLAG	FLAGS	FMODE	FNAME	FORM	FTYPE	HEADER

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	LOOP	MATCH	NUCON	RETREG	R0	R1	R10	R11	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SCAN	TYPE	VCADTLKP	VCADTNXT		
DMSLSY	DSYM	GET1	JSYM	NUCON	NXTSYM	R0	R1	R14	R15				
DMSMDP	ALDRTBLS	FILNAM	LOOP	MDPCALL	MODFLGS	NUCON	PLIST	R0	R1	R14	R15	R2	R3
	R4	TBENT											
DMSMOD	ADMSERL	ADMSFREB	ADTADD	ADTEDF	ADTFLG4	ADTSECT	AERASE	AESTATE	AESTATEW	AFINIS	AFVS	ALDRTBLS	ARDBUF
	ARDTK	AUSRAREA	AWRBUF	BALR	CL	CLEAR	CODE203	DMSERR	DMSSTGSB	DOSFLAGS	DOSMODE	DOSSVC	DSKADR
	DSKLIN	DSKLOC	DSKLST	EDF010	EDF020	ENDLOAD	FILE	FREELOWE	FRSTLOC	FVSECT	FVSFSTAC	FVSFSTAD	FWADDR
	F65535	LASTLMOD	LASTTMOD	LDRFLAGS	LOC	LOCCNT	MDPCALL	MODFLGS	MODGNALL	MODGNDOS	MSG	NOERASE	NOMAPFLG
	NUCON	PRFTSYS	PRFUSYS	PROTFLAG	REGSAV3	RWCNT	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	STORAGE	STRTADDR	SUBFLAG	TBENT
DMSMVE	AADTLKP	ADTDTA	ADTFDOS	ADTFLG1	ADTFLG2	ADTFRO	ADTFRW	ADTSECT	BATFLAGS	BATMOVE	DA	DDNAM	DEVFLAG
	DEVSECT	DMSMVG	DOSFLAGS	DOSSVC	FCBBLKSZ	FCBDD	FCBDEV	FCBDSK	FCBDSMD	FCBDSNAM	FCBINIT	FCBIOSW2	FCBITEM
	FCBLRECL	FCBMMV	FCBMVFIL	FCBMVPDS	FCBOP	FCBOPCB	FCBOSFST	FCBRECFM	FCBSECT	FCBTAP	FCBTAPID	FLAG	IHADEB
	INPUT	LOOP	NUCON	OSFST	OSFSTBLK	OSFSTLRL	OSFSTRFM	OUTPUT	PLIST	PS	RESET	R0	R1
	R10	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SAVEFN												
DMSMVG	BLKSIZE	DA	DOSFLAGS	DTFBLKSZ	DTFIEND	DTFLOGRS	DTFNAME	EIGHT	FCBBLKSZ	FCBLRECL	FCBOSFST	FCBRECFM	FCBSECT
	MACRO	NUCON	OSFST	OSFSTDSN	OSFSTFM	PS	RECS	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9			
DMSNCP	BLKSIZE	BYTE	CODE	CONTROL	DA	ERR1	FILE	FILEMODE	FILENAME	FORM	FREELOWE	FSTD	FSTFMODE
	INPUT	NUCON	PO	QS	READBUF	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R8	R9	SF					
DMSOLD	ADMSFREB	ADMSLIO	AERASE	AESTATE	AFINIS	ALDRTBLS	APRILB	APSV	ARDBUF	AROUND	ASCANN	AUSRAREA	AWRBUF
	BALR	BATFLAGS	BATLOAD	BRAD	CHKTYPE	CLOSELIB	CMD	CMNDLIST	CODE203	COMMONEX	CRDPTR	DMSLGTA	DMSLGTB
	DMSLIB	DMSLSBA	DMSLSBB	DMSLSBC	DMSLSBD	DMSLSY	DYLD	DYNAEND	ENDCDADR	ENTADR	ENTNAME	ESD1ST	ESIDTB
	FINIS	FLAGS	FLAG1	FLAG2	FLAG3	FREELOWE	FSTEPL	FSTXTADR	FTYPE	GPRSAV	LDRADDR	LDRFLAGS	LDRRTCD
	LDRST	LOC	LOCCNT	LOCCT	LUNDEF	MEMBOUND	MODFLGS	NEED	NOAUTO	NODUP	NOINV	NOLIBE	NOREP
	NOSLCADR	NUCON	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	ADMSFREB	AESTATE	ASYSREF	BALR	BGCOM	CODE203	DOSDD	DOSFIRST	DOSNEXT	DOSOP	DOSOSFST	DOSSECT	DOSSYS
	FREESTOR	HEX	LOC	LUBPT	NUCON	OSFST	OSFSTDSK	OSFSTXTN	PUBPT	R0	R1	R12	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	TYPE			
DMSOPT	ABGCOM	BGCOM	DOSFLAGS	DOSMODE	DUMP	JCSW3	JCSW4	NUCON	RESET	R0	R1	R10	R11
	R12	R14	R15	R2	SOB1	TEMOPT							

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSOR1	ADMSFREE	BALR	CODE203	CON	CONTROL	DTFSD	FREESTOR	INPUT	LOC	NOTFOUND	NUCON	OUTPUT	RETURN
	R0	R1	R12	R15	R2	R5	R6	SWITCH	TYPE	VAR			
DMSOR2	FF	RETURN	R1	R12									
DMSOR3	CCW2	NOTFOUND	RETURN	R1	R12	R14							
DMSOSR	CMNDLINE	CMODE	EXEC2	NUCGLOBL	NUCON	NUCOSFLG	NUCOSRLD	NUCOSRUN	NUCTIEIN	PARM	PLIST	PS	R0
	R1	R12	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	TBENT
DMSOVR	ADMSOVS	ASVCSECT	DEC	DMSOVS	FORM	LENOVS	LOC	LOOP	NUCON	OVAPF	OVBPF	OVF1F	OVF1FS
	OVF1GA	OVF1GB	OVF1GS	OVF1ON	OVF1PA	OVF2CM	OVF2NR	OVF2OS	OVF2WA	OVSECT	OVSHO	OVSON	OVSSO
	OVSTAT	RETURN	R0	R1	R12	R14	R15	R3	R4	R5	R6	R7	R8
	SVCSECT	TYPE											
DMSOVS	ASVCSECT	BUFFA	CALLEE	CALLER	CURRSAVE	DEPTH	EFPRS	EGPRS	EGPR0	EGPR15	FLAGS	NOWORK	NUCON
	OLDPSW	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
	TPFSVO	TYPE	TYPFLAG	VMSIZE	XCOUNT	XGPR0	XGPR1	XGPR15					
DMSPIO	ABATABND	ABATLIMT	ADMSERL	ADMSIOW	BATFLAGS	BATLSECT	BATNOEX	BATPRTC	BATPRTL	BATRUN	BATXLIM	BATXPRT	BLANK
	CAW	CSW	DOSFLAGS	EIGHT	ERROR1	ERROR2	ERROR3	NUCON	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9			
DMSPNT	AACTFREE	AACTLKP	ADTCHBA	ADTEDF	ADTFLG4	ADTSECT	AFTADT	AFTARP	AFTAWP	AFTPHYP	AFVS	FVSECT	F65535
	NUCON	REGSAV3	RETURN	R0	R1	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	VCFSTLKP										
DMSPRE	AADTLKP	ADTDTA	ADTID	ADTM	BLANK	DEVADDR	DEVSECT	ERRCODE	LOC	NUCON	OUTBUF	OUTNAME	OUTPUT
	R0	R1	R10	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	VMSIZE										
DMSPRT	ADMSERL	ADMSPIOC	AESTATE	AFINIS	ARDBUF	AREA	BITS	CLOSIO	DIRITEM	DIRITEMX	DIRMEMB	DIRSECT	D1
	D2	FILE	FILEBUFF	FILEMODE	FILENAME	FILETYPE	HEX	INSTALID	LIBDIR	LIBDIRSZ	LIBDIRX	LIBIDENT	LIBSECT
	LOC	NUCON	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	SCAN2							
DMSPRV	AERASE	AFINIS	ASYSREF	AWRBUF	BGCOM	BLANK	DEVTYPE	DOSFLAGS	DOSMODE	DSKLST	FNAME	FSTEPL	FTYPE
	INPUT	LUBPT	NUCON	PUBADR	PUBCUU	PUBDEVT	PUBPT	RDCOUNT	RESET	R0	R1	R10	R12
	R14	R15	R2	R3	SENSE								
DMSPUN	ADMSERL	ADTID	AESTATE	AFINIS	AFVS	ARDBUF	BITS	CLOSIO	D1	FILE	FILEBUFF	FILEMODE	FILENAME
	FILETYPE	FSTEPL	FVSECT	FVSFSTAD	LOC	NUCON	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SCAN2			

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSQRY	ABGCOM	ADTCYL	ADTDBSIZ	ADTDTA	ADTEDF	ADTFDOS	ADTFLG1	ADTFLG2	ADTFLG3	ADTFLG4	ADTFRO	ADTFROS	ADTFRW
	ADTFRWOS	ADTFSTC	ADTID	ADTLEFT	ADTM	ADTMX	ADTNUM	ADTSECT	ADTUSED	AEXTSECT	AFVS	AINTRTBL	ALDRTBLS
	AOUTRTBL	ASYSNAMS	ASYSREF	AUSABRV	BGCOM	BLANK	CMSSEG	CONVERT1	CONVERT2	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	DTADC	DTADT	DUMMY	DUMP	ERROR014	ERROR047	ERROR070	EXTM	EXTSECT	FCBDD	FCBDEV	FCBDSNAM
	FCBDSTYP	FCBFIRST	FCBLABPT	FCBLABT	FCBNL	FCBNSL	FCBNSLNM	FCBNUM	FCBOFF	FCBPOS	FCBSECT	FCBSL	FCBTAPID
	FVSECT	INPUT	LABCRD	LABDVID	LABEXD	LABFDEF	LABFID	LABFILE	LABFIRST	LABFLAG1	LABFLAG2	LABFSEQ	LABGENN
	LABGENV	LABNEXT	LABNUM	LABSEC	LABSECT	LABVOLID	LABVSEQ	LFILEID	LINECT	LMSG	LOC	MACLIBL	MISFLAGS
	MSGFLAGS	NEGITS	NOABBREV	NOIMPCP	NOIMPEX	NOPAGREL	NORDYTIM	NOSTDSYN	NOTFOUND	NUCLDLIB	NUCON	OPTFLAGS	OUTPUT
	OUTPUT10	PRFPOFF	PROTFLAG	REDERRID	R0	R1	R10	R11	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SEARCH	SYSCOM	SYSLINE	SYSNAMES	SYSNCNT	SYSNEND
	S202	TIMCCW	TIMCHAR	TXTLIBS	VCADTLKP	VCADTNXT	WRITE						
DMSRDC	ABATABND	AERASE	AESTATEW	AFINIS	ASCANN	AWRBUF	BATDCMS	BATFLAGS	BATFLAG2	BATRUN	BLKSIZE	CHKTYPE	CLOSIO
	DEVTYPE	FILE	FILEBUFF	FILEMODE	FILENAME	FMODE	IOAREA	NUCON	READ	RETURN	RPLIST	R0	R1
	R10	R11	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	
DMSREA	D2	R0	R1	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	SAVER0	SAVER1	SAVER14	SAVER15									
DMSRNE	AERASE	AFINIS	AINCORE	ARDBUF	AWRBUF	BLANK	CHKEOF	CONVERT	ERR1	ERR104	FMODE	FNAME	FSIZE
	FSTEPL	INBUFF	LOC	LOOP	NUCON	OUTBUFF	PACK	PLIST	RETURN	R0	R1	R10	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	STRTNO	TYPE	VCADTLKW	
DMSRNM	AACTLKP	ADTCHBA	ADTFLG1	ADTFRO	ADTFRW	ADTFTYP	ADTM	AESTATEW	AFTADT	AFVS	AKILLEX	ATFINIS	ATYPSRCH
	AUPDISK	DCHCHGD	DCHFLG1	DCHSECT	ERBIT	ERRCOD1	ERSFLAG	FILE	FVSECT	FVSERAS0	FVSERAS1	FVSERAS2	KXFLAG
	KXWANT	NEWMODE	NEWNAME	NEWTYPE	NUCON	REGSAV1	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	STATEFST	STATFM	STATFST2	STATLST
	UFDBUSY	VCADTLKP	VCFSTLKW										
DMSROS	ADTCYL	ADTDTA	ADTFDOS	ADTFLG1	ADTFLG2	ADTFLG3	ADTFORCE	ADTFROS	ADTFRWOS	ADTIDENT	ADTLEFT	ADTM	ADTSECT
	ALL	BALR	CSW	DOSCBID	DOSFIRST	DOSFLAGS	DOSOSDSN	DOSOSFST	DOSSECT	DOSSVC	DTAD	FCBBLKSZ	FCBDSMD
	FCBDSNAM	FCBDSTYP	FCBFIRST	FCBIOSW2	FCBLRECL	FCBMEMBR	FCBMVPDS	FCBNEXT	FCBOP	FCBOSDSN	FCBOSFST	FCBPROC	FCBRECFM
	FCBSECT	FILEBUFF	FILEBYTE	FILENAME	FILEREAD	LOC	NOTFOUND	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	R0	R1	R10	R11	R12	R14	R15	R2	R3	R4	R5	R6
	R7	R8	R9	SKIP	TYPE	UND	VAR	VCADTNXT					
DMSRRV	AERASE	AESTATE	AFINIS	AREA	ASYSREF	AWRBUF	BGCOM	BLANK	DEVTYPE	DOSDD	DOSDEV	DOSDSK	DOSFIRST
	DOSFLAGS	DOSMODE	DOSOP	DOSOSFST	DOSSECT	DSKLST	EIGHT	FNAME	FTYPE	INPUT	LUBPT	NUCON	OSFST
	OSFSTDSK	OSFSTXTN	OUTBUF	PUBADR	PUBDEVT	PUBPT	RDCOUNT	RESET	R0	R1	R10	R11	R12
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVE1	SENSE	

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSSAB	AABNSVC	ACMSSEG	ADMSFREB	ALL	AOSMODL	APGMSECT	BALR	CLEAR	CODE203	CURRSAVE	DCSSAVAL	DCSSFLAG	DCSSVTLD
	DEBDCBAD	ERRCODE	FCBDD	FCBDEV	FCBDUM	FCBFIRST	FCBSECT	LINKLAST	LOC	NUCON	PGMOPSW	PGMSECT	RESET
	RETRYBIT	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R8	R9	SCBPTR	SCBSAV12	SCBWORK	SETUP	STAEBIT	STAIBIT	TYPE		
DMSSBD	DA	DATAEND	DECAREA	DECKYADR	DECLNGTH	DECRECPT	DECSDECB	DECTYPE	DMSSBS	DMSSBSRT	DUMMY	FCBBYTE	FCBITEM
	FCBKEYS	FCBOP	FCBRECFM	FCBSECT	FCBXTENT	FINIS	IHADECB	IOBIN	IOBIOFLG	KEYCHANG	KEYCHNG	KEYCOUT	KEYEXTPL
	KEYLNGTH	KEYNAME	KEYOP	KEYSECT	KEYTBLAD	KEYTBLNO	KF1	NOTFOUND	OPSECT	PS	R0	R1	R10
	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SEBSAV
	TBLLNGTH	VAR											
DMSSBS	AOPSECT	DA	DECAREA	DECDCBAD	DECIOBPT	DECLNGTH	DECSDECB	DECTYPE	DMSSBD	DMSSEB	FCBBLKCT	FCBBUFF	FCBBYTE
	FCBCATML	FCBCOUT	FCBDEV	FCBDSMD	FCBDSNAM	FCBINIT	FCBITEM	FCBMODE	FCBOP	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	BLANK	BUFFLOC	CHNGFLAG	DECLTH	DMSGIO	EDCB	EDMSK	FLAG	FLAGLOC	FLAG2	FMODE	FNAME	FTYPE
	FV	GIOPLIST	HOLDFLAG	INMODE	ITEM	LINELOC	MSG	NUMLOC	PTR1	PTR2	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R9	SAVCNT
	SAVEAR	SCLNO	SCRBUFAD	SCRFLGS	SCRFLG2	SETUP	TABLIN	TRUNCOL	TWITCH	TYPE	TYPSCR	UTILFLAG	VERCOL1
	VERLEN												
DMSSCT	ADMSROS	AOPSECT	CMSOP	DA	DECDCBAD	DECIOBPT	DECSDECB	FCBCATLD	FCBCATML	FCBCLOSE	FCBCOUT	FCBDEV	FCBINIT
	FCBIOSW	FCBITEM	FCBMEMBR	FCBOP	FCBOS	FCBOSFST	FCBPDS	FCBR13	FCBSECT	FCBTAP	FILENAME	IHADEB	IHADECB
	IOBBFLG	IOBCSW	IOBIOFLG	IOBOUT	MACDIRC	MACLIBL	NUCLDIRC	NUCLDLIB	NUCON	OPSECT	PS	RESET	R0
	R1	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	SAVER14											
DMSSEB	ADMSERL	ADMSROS	AOPSECT	BLK	CMNDLINE	CONRDCNT	CONRDCOD	CONREAD	CONWR	CONWRBUF	CONWRCNT	CONWRCOD	CONWRITE
	DUMMY	FCBBLKCT	FCBBUFF	FCBBYTE	FCBCASE	FCBCATML	FCBCOUT	FCBDEV	FCBDSMD	FCBDSNAM	FCBDSTYP	FCBFORM	FCBINIT
	FCBIO	FCBIOSW	FCBIOSW2	FCBITEM	FCBLABT	FCBMEMBR	FCBMODE	FCBMVFIL	FCBMVPDS	FCBNOEOV	FCBNSL	FCBNSLNM	FCBOFF
	FCBOP	FCBOPCB	FCBOS	FCBPROC	FCBPRPU	FCBREAD	FCBRECFM	FCBRECL	FCBR13	FCBSECT	FCBSL	FCBTAPID	FCBTPSW
	FXD	IHADECB	IOBBCSW	IOBBECBC	IOBBECBP	IOBIN	IOBIOFLG	JFCBIND2	LOC	NUCON	OPSECT	PO	PRINTLST
	PS	PUNCHLST	RDBUFF	RDCCW	RDCOUNT	READLST	RUN	R0	R1	R11	R13	R14	R15
	R2	R3	R4	R8	SAVER14	SEBSAV	TAPE	TAPEBUFF	TAPECOUT	TAPEDEV	TAPELIST	TAPEMASK	TAPEOPER
	TAPESIZE	TLBBLOK	TLBCALL	TLBDWSZ	TLBEOV	TLBFCBPT	TLBLABT	TLBMODE	TLBNAME	TLBNSLNM	TLBOS	TLBTAPID	TLBTYPE
	TSOATCNL	TSOFLAGS	TYPE	UND	VAR	WTM							
DMSSEG	DMSEDC	DMSEDI	DMSEXE	DMSEXI	DMSEXT	DMSGIO	DMSLGT	DMSLIB	DMSLSB	DMSLSY	DMSOLD	DMSSAB	DMSSBD
	DMSSBS	DMSSCR	DMSSCT	DMSSEB	DMSSLN	DMSSMN	DMSSOP	DMSSQS	DMSSVN	DMSSVT	DMSSVU	DMSTLB	DMSXBG
DMSSET	ABAMSYS	ABATABND	ABGCOM	ACMSSEG	ADEVTAB	ADMSERL	ADMSFREB	ADMSFRT	ADOSDCSS	ADTDTA	ADTFDOS	ADTFLG2	ADTM

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	ADTSECT	AESTATE	AEXTSECT	AFREETAB	AINTRTBL	ALDRTBLS	ALL	ALOKTB	ALTASAVE	AOSMODL	AOUTRTBL	APPSAVE	AREA
	ASYSCOM	ASYSNAMS	ASYSREF	ATCBPTR	AUSRAREA	AVSAMSYS	BALR	BAMFLAGS	BATDCMS	BATFLAGS	BATFLAG2	BATNOEX	BATRUN
	BGCOM	BLANK	CMSBAM	CMSDOS	CMSSEG	CMSVSAM	CODE	CODE203	CONTROL	CPULOG	CURRDATE	DCSSAVAL	DCSSFLAG
	DCSSJLNS	DCSSLDED	DCSSVTLD	DEC	DMSDBG	DMSLBR	DOSBAM	DOSFLAGS	DOSKPART	DOSMODE	DOSSVC	DOSTRANS	DOSVSAM
	ERROR014	ERROR047	ERROR048	ERROR070	EXTSECT	FF	FRDSECT	FREELOWE	FREELOW1	FRERESPG	JCSW3	JCSW4	JOBDATE
	LINECT	LMSG	LOADIT	LOADSTRT	LOC	LOCCNT	LTK	LUBPT	MAINHIGH	MISFLAGS	MODFLGS	MSGFLAGS	NEGITS
	NOABBREV	NOIMPCP	NOIMPEX	NOPAGREL	NORDYMSG	NORDYTIM	NOTYPING	NOVMREAD	NUCKEY	NUCON	NUM	OPTFLAGS	OSADTDSK
	OSMODLDW	PIBPT	PPEND	PRFPOFF	PROTFLAG	PUBPT	REDERRID	RESET	RGPRS	SOB1	STRTADDR	SYSCODE	SYSCOM
	SYSLINE	SYSLOAD	SYSNAMES	SYSNEND	SYSREF	TBENT	TCBADR	TCBSAVE	TIMCCW	TIMCHAR	TIMER	TIMINIT	TYPE
	UPSI	USERCODE	USERKEY	VCADTLKP	VMSIZE								
DMSSFF	ADMSFREB	AREA	BALR	CODE203	DMSLOS	LOC	NUCON	NUCTIEIN	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9		
DMSSLN	ADMSFREB	ADTRANS	AESTATE	AFINIS	AFVS	ALDRTBLS	ALIASENT	APGMSECT	ARDBUF	ASVCSECT	AUSRAREA	BALR	CODE203
	COMPSWT	CURRSAVE	DMSLOS	DMSOLD	DMSSMNSB	DSKLIN	DUMCOM	DYLD	DYLTBO	DYMBRNM	DYNAEND	EGPRS	EGPR0
	EGPR1	EGPR13	EGPR14	EGPR15	FILE	FORM	FREELOWE	FRSTLOC	FVSECT	F65535	LASTLMOD	LASTTMOD	LDRFLAGS
	LINKLAST	LINKSTRT	LOC	LOCCNT	LOOP	MODLIST	MSG	NOTFOUND	NUCCBLKS	NUCON	NUCOSFLG	NUCOSRLD	NUCOSRUN
	OLDPSW	OSRESET	OSSFLAGS	OSTEMP	PGMSECT	PRFTSYS	PRFUSYS	PROTFLAG	SCAN	SCBPTR	SSAVE	STORAGE	STRTADDR
	SUBACT	SUBFLAG	SVCSECT	TBENT	USAVEPTR								
DMSSMN	ABGCOM	AUSRAREA	BALRSAVE	BGCOM	COMPSWT	CURRSAVE	DMSDBG	EGPR1	EGPR15	EOCADR	FREELOWE	FRERESPG	LOCCNT
	MAINHIGH	MAINLIST	MAINSTRT	NOPAGREL	NUCON	OPTFLAGS	OSSFLAGS	OSSMNU	PPEND	RETURN	R0	R1	R10
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SSAVE
	TIMCHAR	TOTLIBS											
DMSSOP	AACTLKP	ACBID	ACMSCVT	ADMSFREB	ADTEDF	ADTFLG1	ADTFLG4	ADTFRO	ADTM	ADTNACW	ADTSECT	AERASE	AESTATE
	AFINIS	AFTADT	AFTARP	AFTAWP	AFTFLG	AFTFST	AFTIN	AFTPFST	AFVS	AOPSECT	AOSRET	AUPDISK	BALR
	BLK	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
	FCBBLKCT	FCBBLKS7	FCBBLP	FCBBUFF	FCBBYTE	FCBCASE	FCBCATLD	FCBCATML	FCBCLEAV	FCBCLOSE	FCBCON	FCBCOUT	FCBDCBCT
	FCBDD	FCBDEV	FCBDOSL	FCBDSK	FCBDSMD	FCBDSNAM	FCBDSTYP	FCBDUM	FCBEPL	FCBFIRST	FCBFORM	FCBINIT	FCBIOSW
	FCBIOSW2	FCBITEM	FCBKEYS	FCBLABT	FCBLEAVE	FCBLRECL	FCBMEMBR	FCBMODE	FCBMVPDS	FCBNL	FCBNSL	FCBNSLNM	FCBOFF
	FCBOP	FCBOS	FCBOSFST	FCBPDS	FCBPOS	FCBPROC	FCBPROCC	FCBPROCO	FCBPTR	FCBRDR	FCBRECFM	FCBRECL	FCBRPTR
	FCBSECT	FCBSL	FCBTAP	FCBTAPID	FCBTCLOS	FCBTPSW	FCBXTENT	FF	FILEBYTE	FILEMODE	FILENAME	FILEREAD	FILETYPE
	FSTAIC	FSTD	FSTEPL	FSTFLAGS	FSTFMODE	FSTRECCT	FSTRECFM	FSTRWDSK	FSTXRDSK	FVSECT	FVSFSTAD	FXD	F6
	IHADEB	IOBDCBPT	IOBEND	IOBFLG	IOBIN	IOBIOFLG	IOBNXTAD	IOBSTART	JFCBIND2	JFCBMASK	JFCDSORG	JFCKEYLE	JFCLIMCT
	JFCOPTCD	LOC	MACDIRC	MACLIBL	NUCLDIRC	NUCLDLIB	NUCON	OPSECT	OSFST	OSFSTBLK	OSFSTCHR	OSFSTLRL	OSFSTRFM
	OSIOTYPE	PLIST	PO	PREVIOUS	PS	QS	RESET	RETURN	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVER1	SAVER15
	SSAVE	STATERO	S202	TAPEBUFF	TAPECOUT	TAPEDEV	TAPELIST	TAPEMASK	TAPEOPER	TAPESIZE	TLBBIOK	TLBCALL	TLBCLIN
	TLBCLOUT	TLBDWSZ	TLBFCBPT	TLBLABT	TLBMODE	TLBNAME	TLBNSLNM	TLBOPIN	TLBOPOUT	TLBOS	TLBTAPID	TLBTYPE	TPFACB
	TYPE	TYPFLAG	UND	USAVEPTR	VAR								
DMSSPR	ADMSERL	CAW	CSW	LOAD	LOC	NOCHARS	NUCON	NUM	RESET	R0	R1	R10	R11
	R12	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVER14	SAVER15

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)											
	SENCCW											
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	D1	ERR70E	FINIS	FLAG	INSIZE	LOOP	MISFLAGS	NUCON
	NUM	RELPAGES	RESET	R0	R1	R12	R14	R15	R2	R3	R4	R5	R6
	R7	SAVEAREA	SKIP	VCADTLKW	WRBUF	WRMODE	WRSIZE						
DMSSRV	AERASE	AESTATE	AFINIS	ASYSREF	AWRBUF	BGCOM	BLANK	DEVTYPE	DOSDD	DOSDEV	DOSDSK	DOSFIRST	DOSFLAGS
	DOSMODE	DOSOP	DOSOSFST	DOSSECT	DSKLST	FNAME	FTYPE	INPUT	LUBPT	NUCON	OSFST	OSFSTDSK	OSFSTXTN
	OUTBUF	PUBADR	PUBDEVT	PUBPT	RDCOUNT	RESET	R0	R1	R10	R12	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SAVE1	SENSE	START	XFF		
DMSSSK	CONTROL	DEC	HEX	LOOP	NUCON	NUM	R0	R1	R12	R14	R15	R2	R3
	R4	R5	R6	R8	R9	VMSIZE							
DMSSTG	ABGCOM	ACMSCVT	ADMSFREB	AEXTSECT	AFINIS	ALDRTBLS	ANCHENDA	ANCHSECT	ANCHSIZ	APGMSECT	ASTATEXT	ASYSCOM	ATSOCPPL
	AUSRAREA	BALR	BALRSAVE	BBOXADR	BGCOM	CODE203	COMPSWT	CURRSAVE	DMSDBG	DMSLGTA	DOSFLAGS	DOSKPART	DOSVSAM
	DYLD	DYLIBO	DYMBRNM	EGPR12	EGPR14	EGPR15	EOCADR	EXTSECT	FREELOWE	FRERESPG	IJBBOX	LINKLAST	LINKSTRT
	LOC	LOCCNT	MACDIRC	MACLIBL	MAINHIGH	MAINLIST	MAINSTRT	MISFLAGS	NOPAGREL	NUCAFCHS	NUCCBLKS	NUCON	NUCOSFLG
	NUCSYSDF	OLDPSW	OPTFLAGS	OPTNBYTE	OSSFLAGS	PCTVSAM	PDSDIRSZ	PDSLEN	PDSSECT	PENDLOG	PGEND	PGMSECT	PICADDR
	PPEND	PS	RELPAGES	RETURN	R0	R1	R10	R11	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	ADTCHBA	ADTEDF	ADTFLG1	ADTFLG2	ADTFLG4	ADTFRO	ADTFROS	ADTFRW	ADTFSTSZ	ADTM	ADTMX
	ADTSECT	AFTADT	AFTFLG	AFTFST	AFTRD	AFTWRT	AFVS	BALR	DMSERR	DMSLAD	DMSLADW	DMSLFS	DMSLFSW
	FILE	FVSECT	FVSFSTAD	FVSFSTDB	FVSFSTDT	FVSFSTFV	FVSFSTHP	FVSFSTIC	FVSFSTM	FVSFSTN	FVSFSTRP	FVSFSTWP	FVSFSTYR
	NUCON	NUM	OSFST	OSFSTFLG	OSFSTFM	REGSAV3	R0	R1	R10	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	STATEFST	STATER0	STATFST2	VCFSTLKW	
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									
DMSSVT	ADMPEXEC	ADMSERL	ADMSFREB	ADMSROS	ADTEDF	ADTFLG1	ADTFLG4	ADTFRW	AERASE	AEXTSECT	AOPSECT	APGMSECT	APIE
	ARDBUF	ATFINIS	AUPDISK	AUSRAREA	AWRBUF	BALR	CALLER	CHNGBYTE	CMNDLINE	CMSNAME	CMSOP	CMSTAXE	CODE203
	CONRDCNT	CONREAD	CONVERT	CONWRBUF	CONWRCNT	CONWRITE	CORESIZE	CURRDATE	CURRSAVE	DECSDECB	DEVTYPE	DIAGTIME	DIRNAME
	DIRPTR	DMSDBG	DMSLGT	DMSLSB	DMSSAB	DMSSBDFR	DMSSBS	DMSSCT	DMSSFF	DMSSLN	DMSSLN3	DMSSLN42	DMSSLN6
	DMSSLN7	DMSSLN8	DMSSLN9	DMSSMN	DMSSMN10	DMSSMN4	DMSSMN5	DMSSOP	DMSSOP19	DMSSOP20	DMSSOP22	DMSSOP23	DMSSQS
	DMSSVN	DMSSVN1	DMSSVN2	DMSSVN93	DMSSVN94	DMSSVU	DOSDD	DOSDIRC	DOSFIRST	DOSLIBL	DOSNEXT	DOSSECT	DUMPLIST

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)											
	D1	D3	EFPRS	EGPR0	EGPR1	EGPR13	EGPR14	EGPR15	EIGHT	EXTSECT	FCBBUFF	FCBBYTE	FCBCATLD
	FCBCATML	FCBCOUT	FCBDD	FCBDEV	FCBDOSL	FCBDSK	FCBDSMD	FCBDSNAM	FCBDUM	FCBFIRST	FCBFORM	FCBINIT	FCBIOSW2
	FCBITEM	FCBMEMBR	FCBMMV	FCBMVPDS	FCBOP	FCBOS	FCBOSFST	FCBPDS	FCBSECT	FCBTAB	FCBTAP	FCBTBSP	FF
	FILEBUFF	FILEBYTE	FILECOUT	FILEITEM	FILEMODE	FILENAME	FILETYPE	FLAG	FREELOWE	IHADEB	IHADECB	IHAJFCB	JFCBMASK
	JFCLRECL	LINKSTRT	LOC	LOWSAVE	MACDIRC	MACLIBL	NEWBLKS	NOLOAD1	NOTFOUND	NUCLDIRC	NUCLDLIB	NUCON	NUM
	OLDPSW	OPSECT	OSIOTYPE	OSRESET	OSSFLAGS	OSTEMP	PDSBLKSI	PDSDIR	PDSDIRIT	PDSDIRSZ	PDSENTSZ	PDSFLG1	PDSFNEW
	PDSHDRSZ	PDSIDENT	PDSLEN	PDSSECT	PDSTEMPF	PGMSECT	PLIST	PREVIOUS	PS	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVER14
	SCBPTR	SEBSAV	SSAVE	STIMEXIT	TAXEADDR	TAXEDEF	TAXEEXIT	TAXELNK	TEMPBYTE	TIMBUF	TIMCHAR	TIMER	TYPE
	USAVEPTR	VAR	VCADTLKP	VMSIZE	WAITLIST								
DMSSVU	ADMSFREB	ADTFLG1	ADTFRW	ADTSECT	AERASE	AESTATE	ARDBUF	ATFINIS	AUPDISK	AWRBUF	BALR	CODE203	DATAEND
	EGPR0	EGPR15	EGPR2	EIGHT	FCBBUFF	FCBBYTE	FCBCOUT	FCBDSNAM	FCBDSTYP	FCBFORM	FCBITEM	FCBKEYS	FCBOP
	FCBSECT	FCBXTENT	FILEBUFF	FILENAME	FILETYPE	IOBIN	IOBIOFLG	KEYCHANG	KEYCHNG	KEYCOUT	KEYEOF	KEYEXTPL	KEYFORM
	KEYLNGTH	KEYMARK	KEYNAME	KEYOP	KEYPTR1	KEYPTR2	KEYSECT	KEYTABLE	KEYTBLAD	KEYTBLNO	KEYTYPE	KEYXTNT1	KEYXTNT2
	LOC	NUCON	OPSECT	OSTEMP	PLIST	PS	RESET	R0	R1	R10	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SETUP	SSAVE	TBLLNGTH	VAR
	VCADTLKP												
DMSSYN	AESTATE	AFINIS	AFST	ARDBUF	AUSABRV	CLEAR	ERRCODE	ERR1	FILE	LOC	NOSTDSYN	NUCON	OPTFLAGS
	R0	R1	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7	R8
	SCAN	SETUP	SYSCOM	TYPE									
DMSTIO	ADEVTAB	ATABEND	CC	CSW	DBLWRD1	DBLWRD2	DEVADDR	DEVMISC	DEVNAME	DEVSECT	DEVSIZE	LOOP	NUCON
	PBUFF	PBUFFSZ	PLIST	RETURN	R0	R1	R11	R12	R13	R14	R15	R5	R6
	R7	SILI	TAPE	TYP8809									
DMSTLA	ACMSSEG	ASYSNAMS	ATLBMODL	CMSSEG	DCSSAVAL	DCSSFLAG	DCSSLDED	NUCON	R0	R12	R14	R15	R8
	R9	SYSNAMES	SYSNEND										
DMSTLB	ADMSERL	ADMSFREB	BALR	BSF	BSR	BUFFL	BUFF2	CODE203	CONREAD	CURRDATE	CURRSAVE	DEVSECT	DOSFLAGS
	DOSSVC	D2	EGPR2	EIGHT	ERRET	FCBBLKCT	FCBBLKSZ	FCBDD	FCBLABPT	FCBLEAVE	FCBLRECL	FCBMODE	FCBPOS
	FCBRECFM	FCBSECT	FCBTPSW	FILE	FINE	FSF	HEX	IOBDCBPT	JFCBIND2	JFCBMASK	LABCRD	LABDCRD	LABDEXD
	LABDFID	LABDFSEQ	LABDGENN	LABDGENV	LABDSEC	LABDVID	LABDVSEQ	LABEXD	LABFID	LABFILE	LABFIRST	LABFLAG1	LABFLAG2
	LABFSEQ	LABGENN	LABGENV	LABNEXT	LABNUM	LABSEC	LABSECT	LABVOLID	LABVSEQ	LOC	NRMRET	NUCON	OSSFLAGS
	PACK	PLIST	PS	READ	RESET	RETURN	REW	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SENSE	SETUP	SPEC
	SSAVE	SSAVEPRV	SWTCH	S202	TLBBLKCT	TLBBLOK	TLBCALL	TLBCLIN	TLBCMAC	TLBCMS	TLBDOS	TLBDTFPT	TLBEOV
	TLBFCBPT	TLBLABID	TLBLABT	TLBMODE	TLBMSPC	TLBNSL	TLBNSLMD	TLBNSLNM	TLBOPIN	TLBOPOUT	TLBOS	TLBSL	TLBSUL
	TLBTAPID	TLBTYPE	TYPE	USAVEPTR	WRITE	WTM							
DMSTMA	AADTLKP	ADEVTAB	ADTEDF	ADTFLG4	ADTSECT	ATABEND	BLK	CLPAREN	CSW	DEVADDR	DEVNAME	DEVSECT	DEVSIZE
	DMSLIB	ERROR105	ERROR110	FINIS	FXD	LMSG	LOADING	NUCON	PACK	RESET	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SAVER14	S202	TAPE	TLBBLOK	TLBCMS	TLBLABID	TLBLABT	TLBNSL	TLBNSLMD	TLBNSLNM	TLBOPIN	TLBSL	TLBTAPID
	TYP8809	XFF											

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSTPD	ADEVTAB	ATABEND	BLK	CLPAREN	CSW	DEC	DEVADDR	DEVNAME	DEVSECT	DEVSIZE	DOSFLAGS	DOSSVC	FILE
	FILEBUFF	FILEMODE	FILENAME	FILETYPE	FLAG	FLAG2	FLAG3	FXD	NUCON	RETURN	R0	R1	R10
	R11	R12	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	S202
	TLBBLOK	TLBCMS	TLBLABID	TLBLABT	TLBNSL	TLBNSLMD	TLBNSLNM	TLBOPIN	TLBSL	TLBTAPID	TYP8809	VAR	XFF
DMSTPE	AACTLKP	AADTLKP	ACFILE10	ACFILE20	ACTERS	ACTFLAG	ACTFM	ACTFN	ADEVTAB	ADMSBLKR	ADMSBLKW	ADTCHBA	ADTDBSIZ
	ADTEDF	ADTFLG4	ADTFTYP	ADTM	AERASE	AESTATE	AFINIS	AFSTPLST	AFTARP	AFTAWP	AFTRLST	AFVS	AIOBUFF
	AKILLEX	ARDBUF	AROUND	ATABEND	ATYPSRCH	AUPDISK	AWRBUF	BLANK	BLKSIZE	BLOCKCNT	BRR8	BSF	BSR
	BUFFADR	BYTESRD	CARCTL	CARDIN	CARDOUT	CHCKFILE	CHECKSCN	CHKEOF	CHKFT	CHKFT10	CHKINPUT	CHKMODE	CHKSCNSW
	CHKTYPE	CHLINK	CL	CLASTAPE	CLEAR	CLPAREN	CLR	CMDACT	CNTLADDR	COMPOPT	CONTLSWT	CONTROL	CONVERT
	CONVERT1	CONVERT2	COPYEND	COPYEOF	COPY010	CPCLOSE	CW	CZERO	C6250	DATAIN	DATAOUT	DBLWRD1	DBLWRD2
	DCHCHGD	DCHFLG1	DCHSECT	DENSITY	DEVADDR	DEVMISC	DEVNAME	DEVSECT	DEVSIZE	DRESET	DUMPIT	DUMPMOD	DUMPOK
	DUMPSWT	D1	D2	D200	D3	D556	D6	D6250	D8007TRK	D8009TRK	EDFD010	EDFD020	EDFD040
	EDFL000	EDFL002	EDFL004	EDFL013	EDFL016	EDFL020	EDFL030	EDFL040	EDFL050	EDFL060	EDFL100	EDFL105	EDFL110
	EDFL120	EDF001	EDF004	EDF010	EDF015	EDF018	EDF020	EDF030	EDF080	EDF090	EDF110	EDF120	EDF180
	EDF190	EDF200	EIGHT	EJECTRTN	ENTRDWR	EOFCHK	EOFM	EOFML	EOFN	EOFNEOT	EOTF	EPOINT	ERG
	ERRDCONV	ERRDLDNS	ERRET10	ERRET20	ERRET30	ERRET40	ERRHIDEN	ERROROUT	ERROR002	ERROR003	ERROR010	ERROR014	ERROR017
	ERROR023	ERROR027	ERROR029	ERROR037	ERROR042	ERROR043	ERROR047	ERROR048	ERROR057	ERROR058	ERROR070	ERROR096	ERROR1
	ERROR104	ERROR105	ERROR110	ERROR111	ERROR113	ERROR2	ERROR3	ERROR431	ERROR47M	ERROR70M	ERRO105M	ERRO110B	ERRO111B
	ERRO111M	ERRTRANS	ERR1	ERR104	ERR111	ERR115S	ERR16BP	ERR2RC	ERR7TRK	ERR70E	ERR800BP	ERR9TRK	EXECSVC1
	EXECSVC2	EXTYPEOI	FBLOCK	FDIAG	FF	FFORMAT	FILE	FINE	FIRSTOPT	FLAGIN	FLAGS	FLAGS2	FLAG2
	FMACT	FMOK1	FMOK2	FNACT	FORMOK1	FORMOK2	FREESTOR	FSPARSE	FSR	FSTCMMD	FSTSAVAD	FSTSAVE	FTRDCONV
	FTRDLDNS	FTRTRANS	FTR7TRK	FVSECT	FVSFLG0	FVSFSTM	FVSL1	FVSUFSTC	F65535	HEADER	HEADERTR	HIERR	INBUFF
	INCOMM	INDEXS	INFILE	INFV	INITNO	INMODE	INNAME	INNOIT	INNORD	INPUT	INRPTR	INSIZE	INTYPE
	INWPTR	IOBUFF	KABUFFSZ	KAEJECTR	KALEND	KAOUTPRT	KCFID	KCFROM	KCFTYPE	KCFUNC	KCOPTION	KCPARAM	KCTO
	KC1600B	KC4096B	KC800B	KEEPDEN7	KEEPTRK7	KF1	KF4096	KF800	KH12	KH2	KH3	KH5	KH8
	KXFLAG	KXWANT	LEFTOVER	LHEADER	LINECT	LMSG	LOADBASE	LOADIT	LOADMACH	LOADMNME	LOADMOD	LOADMPL	LOADPROC
	LOADSWT	LOADWR	LOOP	LOOPEY	L18	MATCH	MATCHALL	MATCHFM	MATCHFN	MATCHFT	MESSAGE	MODEL3	MODEL5
	MODEL7	MODEL8	MODESETB	MODESWT	MODNAME	MSGADDR	MSG701	MULTBLK	MVCFILID	MVCLFRNT	NINEOFF	NINETK	NOCOPY
	NODISK	NOEOFN	NOEOT	NOLOAD1	NOLOAD2	NOPRINT	NOSPARSE	NOTACTV	NOTEOT	NOTERM	NOTFOUND	NOTUSED	NOWORK
	NOWRITE	NOWTM	NUCON	OPTBYTE	OUTCOMM	OUTDISK	OUTMODE	OUTNAME	OUTPRINT	OUTPUT	OUTPUT10	OUTSIZE	OUTSVC
	OUTTERM	OVERLAP	PACKNUM	PCT	PERASE	PERFORM	PERFORM1	PERF003	PERF004	PERF005	PERF010	PERF015	PERF020
	PERF030	PERF040	PERF110	PERF120	PERF130	PERF140	PERF150	PERF210	PERF220	POUTPUT	PREPSTAT	PRTERR	PRTMATCH
	PTAPEIO	READ	RESET	RESETM7	RESET7	RETURN	REW	REWIND	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVEFM	SAVEFN
	SAVEFORM	SAVEFT	SAVELNTH	SAVEMODE	SAVERR	SAVER1	SAVER14	SAVE10R	SAVE5	SCANNULL	SCAN2	SCNSWT	SETD7TRK
	SETRTCH	SETUP	SFSTADAT	SFSTAIC	SFSTDAT	SFSTFOP	SFSTFV	SFSTIC	SFSTRP	SFSTWP	SFSTYR	SKPSWT	STATEFST
	STATFM	STATLST	STDEVTAB	STFILE	STORMODE	SYMTAPA	S17	S17A	S202	TAB	TABN	TAPE	TAPEBUF
	TAPECCU	TAPEOF	TAPE05	TAPE06	TAPE07	TAPE08	TAPE10	TAPE100	TAPE110	TAPE120	TAPE13	TAPE130	TAPE140
	TAPE15	TAPE150	TAPE151	TAPE152	TAPE153	TAPE154	TAPE155	TAPE156	TAPE157	TAPE160	TAPE161	TAPE163	TAPE165
	TAPE167	TAPE170	TAPE20	TAPE200	TAPE201	TAPE23	TAPE30	TAPE300	TAPE301	TAPE302	TAPE303	TAPE40	TAPE45
	TAPE50	TAPE60	TAPE61	TAPE62	TAPE65	TAPE70	TAPE80	TAPE90	TAP1	TEMPFILE	TESTDDEN	TESTMOD2	TESTMOD3
	TESTOC	TEST2420	TEST3420	TEST6250	TEST8809	TEST9TRK	TMARK	TPCONTL	TPDUMP	TPDUMP10	TPDVOLMV	TPDVOL1	TPEDIS
	TPEENA	TPEFLG	TPEND	TPINIT	TPLD0	TPLD1	TPLOAD	TPMODEST	TPSBSR	TPSCAN	TPSKIP	TPSLOOP	TPSRSET
	TPVOLEND	TPVOLHDR	TPVOLREW	TPWVOLCK	TPWVOL1	TRACK9	TRTCHE	TRTCHET	TRTCHO	TRTCHOC	TRTCHOT	TRTVOLID	TST34208

```
MODULE        EXTERNAL REFERENCES (LABELS AND MODULES)

          TTRANS   TYP2401  TYP2420  TYP3420  TYP8809  UFDBUSY  UP1BLK   VBLOCK   VCFSTLKP VCFSTLKW VFORMAT  VOLERR   WRBIT
          WRBUF    WRBUFF   WRFILE   WRFV     WRITE    WRITEOUT WRITERTN WRITNO   WRMODE   WRNAME   WRNOIT   WRSIZE   WTM
          WTM2     WVOL1    XFF

DMSTPF    AACTLKP  AADTLKP  ACFILE10 ACFILE20 ACTERS   ACTFLAG  ACTFM    ACTFN    ADEVTAB  ADMSBLKR ADMSBLKW ADTCHBA  ADTDBSIZ
          ADTEDF   ADTFLG4  ADTFTYP  ADTM     ADTSECT  AERASE   AESTATE  AFINIS   AFSTPLST AFTARP   AFTAWP   AFTRLST  AFVS
          AIOBUFF  AKILLEX  ARDBUF   AROUND   ATABEND  ATYPSRCH AUPDISK  AWRBUF   BLANK    BLKSIZE  BLOCKCNT BRR8     BSF
          BSR      BUFFADR  BYTESRD  CARCTL   CARDIN   CARDOUT  CHCKFILE CHECKSCN CHKEOF   CHKFT    CHKFT10  CHKINPUT CHKMODE
          CHKSCNSW CHKTYPE  CHLINK   CL       CLASTAPE CLEAR    CLPAREN  CLR      CMDACT   CNTLADDR COMPOPT  CONTLSWT CONTROL
          CONVERT  CONVERT1 CONVERT2 COPYEND  COPYEOF  COPY010  CPCLOSE  CW       CZERO    C6250    DATAIN   DATAOUT  DBLWRD1
          DBLWRD2  DCHCHGD  DCHFLG1  DCHSECT  DENSITY  DEVADDR  DEVMISC  DEVNAME  DEVSECT  DEVSIZE  DRESET   DUMPIT   DUMPMOD
          DUMPOK   DUMPSWT  D1       D2       D200     D3       D556     D6       D6250    D8007TRK D8009TRK EDFD010  EDFD020
          EDFD040  EDFL000  EDFL002  EDFL004  EDFL013  EDFL016  EDFL020  EDFL030  EDFL040  EDFL050  EDFL060  EDFL100  EDFL105
          EDFL110  EDFL120  EDF001   EDF004   EDF010   EDF015   EDF018   EDF020   EDF030   EDF080   EDF090   EDF110   EDF120
          EDF180   EDF190   EDF200   EIGHT    EJECTRTN ENTRDWR  EOFCHK   EOFM     EOFML    EOFN     EOFNEOT  EOTF     EPOINT
          ERG      ERRDCONV ERRDLDNS ERRET10  ERRET20  ERRET30  ERRET40  ERRHIDEN ERROROUT ERROR002 ERROR003 ERROR010 ERROR014
          ERROR017 ERROR023 ERROR027 ERROR029 ERROR037 ERROR042 ERROR043 ERROR047 ERROR048 ERROR057 ERROR058 ERROR070 ERROR096
          ERROR1   ERROR104 ERROR105 ERROR110 ERROR111 ERROR113 ERROR2   ERROR3   ERROR431 ERROR47M ERROR70M ERRO105M ERRO110B
          ERRO111B ERRO111M ERRTRANS ERR1     ERR104   ERR111   ERR115S  ERR16BP  ERR2RC   ERR7TRK  ERR70E   ERR800BP ERR9TRK
          EXECSVC1 EXECSVC2 EXTYPEOI FBLOCK   FDIAG    FF       FFORMAT  FILE     FINE     FIRSTOPT FLAGIN   FLAGS    FLAGS2
          FLAG2    FMACT    FMOK1    FMOK2    FNACT    FORMOK1  FORMOK2  FREESTOR FSPARSE  FSR      FSTCMMD  FSTSAVAD FSTSAVE
          FTRDCONV FTRDLDNS FTRTRANS FTR7TRK  FVSECT   FVSFLG0  FVSFSTM  FVSL1    FVSUFSTC F65535   HEADER   HEADERTR HIERR
          INBUFF   INCOMM   INDEXS   INFILE   INFV     INITNO   INMODE   INNAME   INNOIT   INNORD   INPUT    INRPTR   INSIZE
          INTYPE   INWPTR   IOBUFF   KABUFFSZ KAEJECTR KALEND   KAOUTPRT KCFID    KCFROM   KCFTYPE  KCFUNC   KCOPTION KCPARAM
          KCTO     KC1600B  KC4096B  KC800B   KEEPDEN7 KEEPTRK7 KF1      KF4096   KF800    KH12     KH2      KH3      KH5
          KH8      KXFLAG   KXWANT   LEFTOVER LHEADER  LINECT   LMSG     LOADBASE LOADIT   LOADMACH LOADMNME LOADMOD  LOADMPL
          LOADPROC LOADSWT  LOADWR   LOOP     LOOPEY   L18      MATCH    MATCHALL MATCHFM  MATCHFN  MATCHFT  MESSAGE  MODEL3
          MODEL5   MODEL7   MODEL8   MODESETB MODESWT  MODNAME  MSGADDR  MSG701   MULTBLK  MVCFILID MVCLFRNT NINEOFF  NINETK
          NOCOPY   NODISK   NOEOFN   NOEOT    NOLOAD1  NOLOAD2  NOPRINT  NOSPARSE NOTACTV  NOTEOT   NOTERM   NOTFOUND NOTUSED
          NOWORK   NOWRITE  NOWTM    NUCON    OPTBYTE  OUTCOMM  OUTDISK  OUTMODE  OUTNAME  OUTPRINT OUTPUT   OUTPUT10 OUTSIZE
          OUTSVC   OUTTERM  OVERLAP  PACKNUM  PCT      PERASE   PERFORM  PERFORM1 PERF003  PERF004  PERF005  PERF010  PERF015
          PERF020  PERF030  PERF040  PERF110  PERF120  PERF130  PERF140  PERF150  PERF210  PERF220  POUTPUT  PREPSTAT PRTERR
          PRTMATCH PTAPEIO  READ     RESET    RESETM7  RESET7   RETURN   REW      REWIND   SAVEFM   SAVEFN   SAVEFORM SAVEFT
          SAVELNTH SAVEMODE SAVERR   SAVER1   SAVER14  SAVE10R  SAVE5    SCANNULL SCAN2    SCNSWT   SETD7TRK SETRTCH  SETUP
          SFSTADAT SFSTAIC  SFSTDAT  SFSTFOP  SFSTFV   SFSTIC   SFSTRP   SFSTWP   SFSTYR   SKPSWT   STATEFST STATFM   STATLST
          STDEVTAB STFILE   STORMODE SYMTAPA  S17      S17A     S202     TAB      TABN     TAPE     TAPEBUF  TAPECCU  TAPEOF
          TAPE05   TAPE06   TAPE07   TAPE08   TAPE10   TAPE100  TAPE110  TAPE120  TAPE13   TAPE130  TAPE140  TAPE15   TAPE150
          TAPE151  TAPE152  TAPE153  TAPE154  TAPE155  TAPE156  TAPE157  TAPE160  TAPE161  TAPE163  TAPE165  TAPE167  TAPE170
          TAPE20   TAPE200  TAPE201  TAPE23   TAPE30   TAPE300  TAPE301  TAPE302  TAPE303  TAPE40   TAPE45   TAPE50   TAPE60
          TAPE61   TAPE62   TAPE65   TAPE70   TAPE80   TAPE90   TAP1     TEMPFILE TESTDDEN TESTMOD2 TESTMOD3 TESTOC   TEST2420
          TEST3420 TEST6250 TEST8809 TEST9TRK TMARK    TPCONTL  TPDUMP   TPDUMP10 TPDVOLMV TPDVOL1  TPEDIS   TPEENA   TPEFLG
          TPEND    TPINIT   TPLD0    TPLD1    TPLOAD   TPMODEST TPSBSR   TPSCAN   TPSKIP   TPSLOOP  TPSRSET  TPVOLEND TPVOLHDR
          TPVOLREW TPWVOLCK TPWVOL1  TRACK9   TRTCHE   TRTCHET  TRTCHO   TRTCHOC  TRTCHOT  TRTVOLID TST34208 TTRANS   TYP2401
          TYP2420  TYP3420  TYP8809  UFDBUSY  UP1BLK   VBLOCK   VCFSTLKP VCFSTLKW VFORMAT  VOLERR   WRBIT    WRBUF    WRBUFF
          WRFILE   WRFV     WRITE    WRITEOUT WRITERTN WRITNO   WRMODE   WRNAME   WRNOIT   WRSIZE   WTM      WTM2     WVOL1
          XFF
```

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSTPG	AACTLKP	AADTLKP	ACFILE10	ACFILE20	ACTERS	ACTFLAG	ACTFM	ACTFN	ADEVTAB	ADMSBLKR	ADMSBLKW	ADTCHBA	ADTDBSIZ
	ADTEDF	ADTFLG4	ADTFTYP	ADTM	ADTSECT	AERASE	AESTATE	AFINIS	AFSTPLST	AFTARP	AFTAWP	AFTRLST	AFVS
	AIOBUFF	AKILLEX	ARDBUF	AROUND	ATABEND	ATYPSRCH	AUPDISK	AWRBUF	BLANK	BLKSIZE	BLOCKCNT	BRR8	BSF
	BSR	BUFFADR	BYTESRD	CARCTL	CARDIN	CARDOUT	CHCKFILE	CHECKSCN	CHKEOF	CHKFT	CHKFT10	CHKINPUT	CHKMODE
	CHKSCNSW	CHKTYPE	CHLINK	CL	CLASTAPE	CLEAR	CLPAREN	CLR	CMDACT	CNTLADDR	COMPOPT	CONTLSWT	CONTROL
	CONVERT	CONVERT1	CONVERT2	COPYEND	COPYEOF	COPY010	CPCLOSE	CW	CZERO	C6250	DATAIN	DATAOUT	DBLWRD1
	DBLWRD2	DCHCHGD	DCHFLG1	DCHSECT	DENSITY	DRESET	DUMPIT	DUMPMOD	DUMPOK	DUMPSWT	D1	D2	D200
	D3	D556	D6	D6250	D8007TRK	D8009TRK	EDFD010	EDFD020	EDFD040	EDFL000	EDFL002	EDFL004	EDFL013
	EDFL016	EDFL020	EDFL030	EDFL040	EDFL050	EDFL060	EDFL100	EDFL105	EDFL110	EDFL120	EDF001	EDF004	EDF010
	EDF015	EDF018	EDF020	EDF030	EDF080	EDF090	EDF110	EDF120	EDF180	EDF190	EDF200	EIGHT	EJECTRTN
	ENTRDWR	EOFCHK	EOFM	EOFML	EOFN	EOFNEOT	EOTF	EPOINT	ERG	ERRDCONV	ERRDLDNS	ERRET10	ERRET20
	ERRET30	ERRET40	ERRHIDEN	ERROROUT	ERROR002	ERROR003	ERROR010	ERROR014	ERROR017	ERROR023	ERROR027	ERROR029	ERROR037
	ERROR042	ERROR043	ERROR047	ERROR048	ERROR057	ERROR058	ERROR070	ERROR096	ERROR1	ERROR104	ERROR105	ERROR110	ERROR111
	ERROR113	ERROR2	ERROR3	ERROR431	ERROR47M	ERROR70M	ERRO105M	ERRO110B	ERRO111B	ERRO111M	ERRTRANS	ERR1	ERR104
	ERR111	ERR115S	ERR16BP	ERR2RC	ERR7TRK	ERR70E	ERR800BP	ERR9TRK	EXECSVC1	EXECSVC2	EXTYPEOI	FBLOCK	FDIAG
	FF	FFORMAT	FILE	FINE	FIRSTOPT	FLAGIN	FLAGS	FLAGS2	FLAG2	FMACT	FMOK1	FMOK2	FNACT
	FORMOK1	FORMOK2	FREESTOR	FSPARSE	FSR	FSTCMMD	FSTSAVAD	FSTSAVE	FTRDCONV	FTRDLDNS	FTRTRANS	FTR7TRK	FVSECT
	FVSFLG0	FVSFSTM	FVSL1	FVSUFSTC	F65535	HEADER	HEADERTR	HIERR	INBUFF	INCOMM	INDEXS	INFILE	INFV
	INITNO	INMODE	INNAME	INNOIT	INNORD	INPUT	INRPTR	INSIZE	INTYPE	INWPTR	IOBUFF	KABUFFSZ	KAEJECTR
	KALEND	KAOUTPRT	KCFID	KCFROM	KCFTYPE	KCFUNC	KCOPTION	KCPARAM	KCTO	KC1600B	KC4096B	KC800B	KEEPDEN7
	KEEPTRK7	KF1	KF4096	KF800	KH12	KH2	KH3	KH5	KH8	KXFLAG	KXWANT	LEFTOVER	LHEADER
	LINECT	LMSG	LOADBASE	LOADIT	LOADMACH	LOADMNME	LOADMOD	LOADMPL	LOADPROC	LOADSWT	LOADWR	LOOP	LOOPEY
	L18	MATCH	MATCHALL	MATCHFM	MATCHFN	MATCHFT	MESSAGE	MODEL3	MODEL5	MODEL7	MODEL8	MODESETB	MODESWT
	MODNAME	MSGADDR	MSG701	MULTBLK	MVCFILID	MVCLFRNT	NINEOFF	NINETK	NOCOPY	NODISK	NOEOFN	NOEOT	NOLOAD1
	NOLOAD2	NOPRINT	NOSPARSE	NOTACTV	NOTEOT	NOTERM	NOTFOUND	NOTUSED	NOWORK	NOWRITE	NOWTM	NUCON	OPTBYTE
	OUTCOMM	OUTDISK	OUTMODE	OUTNAME	OUTPRINT	OUTPUT	OUTPUT10	OUTSIZE	OUTSVC	OUTTERM	OVERLAP	PACKNUM	PCT
	PERASE	PERFORM	PERFORM1	PERF003	PERF004	PERF005	PERF010	PERF015	PERF020	PERF030	PERF040	PERF110	PERF120
	PERF130	PERF140	PERF150	PERF210	PERF220	POUTPUT	PREPSTAT	PRTERR	PRTMATCH	PTAPEIO	READ	RESET	RESETM7
	RESET7	RETURN	REW	REWIND	SAVEFM	SAVEFN	SAVEFORM	SAVEFT	SAVELNTH	SAVEMODE	SAVERR	SAVER1	SAVER14
	SAVE10R	SAVE5	SCANNULL	SCAN2	SCNSWT	SETD7TRK	SETRTCH	SETUP	SFSTADAT	SFSTAIC	SFSTDAT	SFSTFOP	SFSTFV
	SFSTIC	SFSTRP	SFSTWP	SFSTYR	SKPSWT	STATEFST	STATFM	STATLST	STDEVTAB	STFILE	STORMODE	SYMTAPA	S17
	S17A	S202	TAB	TABN	TAPE	TAPEBUF	TAPECCU	TAPEOF	TAPE05	TAPE06	TAPE07	TAPE08	TAPE10
	TAPE100	TAPE110	TAPE120	TAPE13	TAPE130	TAPE140	TAPE15	TAPE150	TAPE151	TAPE152	TAPE153	TAPE154	TAPE155
	TAPE156	TAPE157	TAPE160	TAPE161	TAPE163	TAPE165	TAPE167	TAPE170	TAPE20	TAPE200	TAPE201	TAPE23	TAPE30
	TAPE300	TAPE301	TAPE302	TAPE303	TAPE40	TAPE45	TAPE50	TAPE60	TAPE61	TAPE62	TAPE65	TAPE70	TAPE80
	TAPE90	TAP1	TEMPFILE	TESTDDEN	TESTMOD2	TESTMOD3	TESTOC	TEST2420	TEST3420	TEST6250	TEST8809	TEST9TRK	TMARK
	TPCONTL	TPDUMP	TPDUMP10	TPDVOLMV	TPDVOL1	TPEDIS	TPEENA	TPEFLG	TPEND	TPINIT	TPLD0	TPLD1	TPLOAD
	TPMODEST	TPSBSR	TPSCAN	TPSKIP	TPSLOOP	TPSRSET	TPVOLEND	TPVOLHDR	TPVOLREW	TPWVOLCK	TPWVOL1	TRACK9	TRTCHE
	TRTCHET	TRTCHO	TRTCHOC	TRTCHOT	TRTVOLID	TST34208	TTRANS	TYP2401	TYP2420	TYP3420	TYP8809	UFDBUSY	UP1BLK
	VBLOCK	VCFSTLKP	VCFSTLKW	VFORMAT	VOLERR	WRBIT	WRBUF	WRBUFF	WRFILE	WRFV	WRITE	WRITEOUT	WRITERTN
	WRITNO	WRMODE	WRNAME	WRNOIT	WRSIZE	WTM	WTM2	WVOL1	XFF				
DMSTQQ	ADTDTA	ADTFLG1	ADTFLG2	ADTFMFD	ADTFRW	ADTQQM	AQQTRK	ATRKLKP	ATRKLKPX	COUNT	DTADT	FVSECT	F65535
	NUCON	RETURN	TRKLSAVE										

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)											
DMSTRK	ADMSFREB	ADTAMHD	ADTAMHO	ADTARES	ADTCHMAP	ADTDAMAP	ADTEDFAE	ADTFLG1	ADTFLG2	ADTFLG4	ADTFMFD	ADTFRW	ADTLAST
	ADTLEFT	ADTMSK	ADTNUM	ADTRES	ADTSECT	ADTUSED	ADT1ST	ALABELWR	BALR	BLANK	CODE203	DCHBWPTR	DCHCHMAP
	DCHDAMAP	DCHDATA	DCHDTSIZ	DCHDWSIZ	DCHFLG2	DCHFULL	DCHFWPTR	DCHPFIXL	DCHSECT	DCHSEQBD	D1	D3	ERROR1
	ERROR2	NUCON	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	TYPE							
DMSTYP	AESTATE	AFINIS	ARDBUF	AREA	D1	ERR1	FILE	FMODE	FNAME	FSTAIC	FSTD	FSTEPL	FSTFMODE
	FSTLRECL	FSTRECFM	FTYPE	HEX	IOAREA	MSGFLAGS	NOTYPING	NUCON	RETURN	R0	R1	R10	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	START	TYPLIN		
DMSUPD	ADTEDF	ADTFLG1	ADTFLG4	ADTFRO	ADTFRW	ADTID	ADTM	ADTMX	ADTSECT	AERASE	AESTATE	AEXTEND	AFINIS
	AFVS	ARDBUF	AWRBUF	BALR	BUFFA	CORITEM	CTL	DOSFLAGS	DOSSVC	D2	D3	D6	FNAME
	FORM	FPTR	FREEAD	FREELEN	FVSECT	FVSFSTAC	FVSFSTAD	ITEM	MISFLAGS	NEWNAME	NOERASE	NOREP	NOTERM
	NUCON	PLIST	PTR1	PTR2	REGSAV	RELPAGES	REPL	RESET	RETCODE	RETURN	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SPARES	TYPE	VCADTLKP	VCADTLKW									
DMSUTL	BLANK	CHKEOF	CLEAR	COPYEND	DA	DEBTCBAD	DUMMY	FCBBLKSZ	FCBCATLD	FCBDD	FCBDEV	FCBDSNAM	FCBDSTYP
	FCBINIT	FCBIOSW2	FCBMVPDS	FCBSECT	FCBTAB	FF	FLAG	FLAG2	FSTD	FSTFLAGS	FSTLRECL	FSTRECFM	FSTRWDSK
	FSTXRDSK	HEADER	INPUT	IOBCSW	NEWNAME	NOERASE	NUCON	NUCTIEIN	OUTPUT	PO	PS	R0	R1
	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	SAVEAREA	SETUP	SF	SYSUT1								
DMSVIB	ACMSCVT	ADMSERL	ASYSNAMS	AVIPWORK	AVSAMSYS	BALRSAVE	CMSVSAM	CONTROL	DEC	LOADIT	NUCON	RESET	R0
	R1	R12	R14	R15	R2	R3	R5	SYSNAMES	SYSNEND	TYPE	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	CURRSAVE	DOSDD	DOSDEV	DOSDSMD	DOSDUM	DOSEXTNO	DOSEXTTB	DOSFIRST	DOSFLAGS
	DOSNEXT	DOSRC	DOSSECT	DOSSVC	DOSVOLNO	DOSVOLTB	DOSYSXXX	EIGHT	EPOINT	EXENACTB	EXENADDR	EXLEODF	EXLEODL
	EXLEODP	EXLJFN	EXLJRNL	EXLLEN	EXLLERF	EXLLERL	EXLLERP	EXLSYNF	EXLSYNL	EXLSYNP	FZERO	IKQACB	IKQEXLST
	IKQRPL	NUCON	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	TYPE	VIPINIT	VIPSOP	VIPTCLOS	VSAMFLG1				
DMSVLT	ADMSERL	AFINIS	ASYSREF	BGCOM	DOSDD	DOSFIRST	DOSF1AD	DOSNEXT	DOSOP	DOSSECT	DTFAVAIL	DTFBLKSZ	DTFCCW
	DTFFLG1	DTFFLG2	DTFFLG5	DTFGVIOA	DTFIGNOP	DTFINPUT	DTFIOA1	DTFLGMOD	DTFNAME	DTFOPEN	DTFSD	DTFWRKFL	DTFX
	DTFXBLSZ	DTFXCCWP	DTFXFBLP	DTFXLMPT	DTFXOCWP	DTFXSIO1	DTFXSIO2	DTFXXLEN	ERROR2	F1DWDS	LIOCSCOM	NUCON	OCTCPDI
	OCTS	OCTSDWDS	OCTSPMGT	OCT1FLAG	PIBPT	R0	R1	R10	R11	R12	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SVEARA	SVEA0008	TRANTAB	USRRGLEN		
DMSVSR	AAMSSYS	ABGCOM	ACBLIST	ACLOSE	ACMSCVT	ADIKQLAB	ADMSFREB	ADMSVIB	ALOKTB	ASYSNAMS	ATCBPTR	AVIPWORK	AVSAMSYS
	AVSRWORK	BALR	BGCOM	CODE203	DOSFLAGS	DOSMODE	DOSSVC	ERROR1	ERR1	NUCON	PPEND	REGSAV	TCBADR
	TCBFLAGS	VIPINIT	VSAMFLG1	VSAMRUN	VSAMSERV	VSAMSOS							

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSXBG	AREA	CMNDLINE	DMSXER	DMSXMARS	DMSXSCIM	DMSXSDAP	DMSXSDLS	DMSXSDTB	DMSXSDTX	DMSXSUFL	DMSXSUVR	DOSFLAGS	DOSSVC
	D1	D2	D3	PKE3278	LSCFWPTR	LSCRBYTE	LSCREEN	LSCRMSG	LSCZDEPT	MISFLAGS	MSG	NUCON	RELPAGES
	RESET	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R6
	R7	R8	R9	SAVBWPTR	SAVEREG	SAVLSAVB	SF	SYNFWPTR	SYNLSYND	SYNSUB	TYPE	ZDEBWPTR	ZDEFLSIZ
	ZDEFWPTR	ZDELSCPT	ZDELZDEB	ZDELZDED	ZDEPFKPT	ZDEPRSPT	ZDEQMBUF	ZDEQMPTR	ZDESC	ZDETOPPT	ZDEZDEPT	ZFOAPLON	ZFOATERM
	ZFOBLKPT	ZFOBUFIM	ZFOBUFIO	ZFOCLRSC	ZFOCSRFL	ZFOCSRSZ	ZFOCSSTK	ZFOCURFL	ZFOC2741	ZFOC3215	ZFOC3270	ZFOC3278	ZFOEDGON
	ZFOERCOD	ZFOFKCOD	ZFOFLAG1	ZFOFLAG2	ZFOFLAG3	ZFOFLAG4	ZFOHVCED	ZFOHVCP1	ZFOIOCMP	ZFOIOTBL	ZFOLFWPT	ZFOLGSCB	ZFOLNAME
	ZFOLSCPT	ZFOLSTSV	ZFOLZFOB	ZFOLZFOD	ZFOMSGCT	ZFONC	ZFONCOLS	ZFONDSPC	ZFONFILE	ZFONROWS	ZFOOPNB1	ZFOPLIST	ZFOPRFPT
	ZFORMTUB	ZFOSAVNE	ZFOSAVSV	ZFOSOSIM	ZFOSVEND	ZFOSYNPT	ZFOTWRMD	ZFOTXTON	ZFOUFLDS	ZFOWKBUF	ZFOXER	ZFOXSUFL	ZFOZMAPT
	ZMACST	ZMAFWPTR											
DMSXCG	COUNT	DMSXCN	DMSXFCIN	DMSXFCML	DMSXFCNX	DMSXFCPC	DMSXFCPL	DMSXFCRC	DMSXFCRL	DMSXFCRM	DMSXFCSP	DMSXFCSU	DMSXFCUP
	DMSXSDLW	DMSXSDUP	DMSXSUCC	DMSXSUEF	DMSXSULG	DMSXSUNC	DMSXSUPR	DMSXSURV	DMSXSUTE	DMSXSUTP	DMSXSUTY	D1	D2
	D3	MAXBUFLG	PT	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3
	R4	R5	R6	R7	R8	R9	SAVBWPTR	SAVBYTE1	SAVBYTE2	SAVEREG	SAVLSAVB	SAVREG0	SAVWORD1
	ZDEACURD	ZDEACURL	ZDEARBCH	ZDEARBON	ZDECANON	ZDECFILL	ZDECGCNT	ZDECLTGT	ZDECURCL	ZDECURLN	ZDEENDRG	ZDEFLAG1	ZDEFLAG2
	ZDEFLAG3	ZDEFLAG4	ZDEFLEOF	ZDEFLRIG	ZDEFLSIZ	ZDEFLTOF	ZDELRECL	ZDENBTBC	ZDESC	ZDESTYLN	ZDETABCL	ZDETAYON	ZDETOPPT
	ZDETOPRG	ZDETRUNC	ZDEUPDON	ZDEVERON	ZDEWIDTH	ZDEZONEL	ZDEZONER	ZFOC3270	ZFOEDGON	ZFOFLAG1	ZFOFLAG3	ZFOFREPT	ZFOLADDR
	ZFOLBWPT	ZFOLFLAG	ZFOLFWPT	ZFOLGOP1	ZFOLGOP2	ZFOLNCHG	ZFOLNDSP	ZFOLSTSV	ZFOMOVUP	ZFONC	ZFOOPABS	ZFOOPFL1	ZFOOPFL2
	ZFOOPNB1	ZFOOPNB2	ZFOOPNB3	ZFOOPNB4	ZFOOPNB5	ZFOOPNB6	ZFOOPNB7	ZFOOPNB8	ZFOOPST1	ZFOOPST2	ZFOOPST3	ZFOOPST4	ZFOOPTGT
	ZFORTBYT	ZFORTCOD	ZFOSUBCM	ZFOWKTBL	ZFOXER	ZFOXSUFL							
DMSXCM	ASCANN	ASTATE	DMSXFCML	DMSXSDUP	DMSXSURV	DMSXSUTP	DOSFLAGS	DOSSVC	D1	D2	D3	LASTCMND	MAXBUFLG
	NOIMPCP	NOIMPEX	NUCON	OPTFLAGS	RESET	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R7	R8	R9	SAVBWPTR	SAVEREG	SAVLSAVB	SAVREG0	STACK	SUBACT	SUBFLAG	ZDEACURL
	ZDEFLAG1	ZDEFLEOF	ZDEFLTOF	ZDESC	ZDETRUNC	ZFOCLRLG	ZFOCLRSC	ZFOCURSV	ZFOFLAG2	ZFOFLAG4	ZFOFNAME	ZFOFTYPE	ZFOIMPCM
	ZFOLGOP1	ZFOLSTSV	ZFONC	ZFOOPABS	ZFOOPFL1	ZFOOPNB1	ZFOOPNB2	ZFOOPNB3	ZFOOPST1	ZFOPLIST	ZFORTBYT	ZFORTCOD	ZFOXER
	ZFOXSUFL												
DMSXCN	DMSXSTLG	D1	D2	R0	R1	R10	R11	R12	R13	R14	R15	R3	R4
	R5	R6	R7	R8	R9	SAVBWPTR	SAVEREG	SAVLSAVB	SAVREG0	SAVREG1	SAVWORD1	ZDESC	ZDEWIDTH
	ZFOFREPT	ZFOLADDR	ZFOLBWPT	ZFOLFWPT	ZFOLNAME	ZFOLSTSV	ZFONC	ZFOXSUFL					
DMSXCP	AACTLKP	ADMSERL	ADMSFREB	ADTDTA	ADTFDOS	ADTFLG2	ADTFLG3	ADTFROS	ADTFRW	ADTM	AFINIS	AFVS	ARDBUF
	ASYSREF	AWRBUF	BALR	BGCOM	CCBCCW	CCBCNT	CCBCOM1	CCBCOM2	CCBCSW	CCBCSW1	CCBCSW2	CCBDC	CCBEOC
	CCBEOF	CCBERMAP	CCBILEN	CCBSUCLS	CCBSUNUM	CCBSYMU	CCBUE	CCBVER	CODE203	CONTROL	CONVERT	CONVERT2	CONWR
	CSW	DOSBLKSZ	DOSBUFF	DOSBYTE	DOSCBID	DOSCCHHR	DOSCOUT	DOSDEV	DOSDSK	DOSDSMD	DOSDSNAM	DOSDTF	DOSFLAGS
	DOSINIT	DOSITEM	DOSOP	DOSOSDSN	DOSOSFST	DOSR	DOSREAD	DOSSAVE	DOSSECT	DOSSENSE	DOSTAPID	DOSWORK	DTFFLG2
	DTFSD	DTFX	DTFXFBLP	DTFXIDEN	DTFXRCIC	EIGHT	FF	INPUT	LUBPT	NICLPT	NUCON	OUTPUT	PUBADR
	PUBCUU	PUBDEVT	PUBDSKM	PUBPT	PUBTAPM1	RDBUF	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SKIP	TAPE	VCADTLKP	WRBUF
DMSXCT	AINCORE	AREA	DMSXFCCC	DMSXFCCL	DMSXFCML	DMSXFCPC	DMSXFDFI	DMSXFDLN	DMSXIORD	DMSXIOWR	DMSXMARS	DMSXSCPR	DMSXSCRV
	DMSXSDLS	DMSXSDSC	DMSXSUCC	DMSXSUCK	DMSXSUEX	DMSXSULK	DMSXSURV	DMSXSUTE	DMSXSUTY	D1	D2	D3	D6
	INCRNO	LSCARWLG	LSCFCHGD	LSCFHIGH	LSCFRSVD	LSCFWPTR	LSCRBYTE	LSCRCURL	LSCRDSPH	LSCRDSPV	LSCREEN	LSCRFADD	LSCRFLG1
	LSCRFLG2	LSCRFLNE	LSCRIMAD	LSCRINPU	LSCRLGTH	LSCRLINE	LSCRLTBL	LSCRSIZE	LSCRWIDT	LSCZDEPT	L18	MAXBUFLG	MSG

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	PRSCB	PRSEQBFL	PRSFLAG2	PRSFNAME	PRSFWPTR	PRSLPRSB	PRSLPRSD	PRSLRECL	PRSMSKLN	PRSNSPAN	PRSSERIN	PRSSGM01	PRSSGM02
	PRSSGM03	PRSSGM04	PRSSGM05	PRSSGM06	PRSSGM07	PRSSGM08	PRSTABCL	PRSVERCL	PRSZONEL	READ	RESET	R0	R1
	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	SAVBWPTR	SAVBYTE1	SAVEREG	SAVLSAVB	SAVREG0	SAVREG12	SAVWORD1	SAVWORD2	STRTNO	ZDEACURD	ZDEATSID	ZDEATSMD
	ZDEBWPTR	ZDECGCNT	ZDECSCOL	ZDECSLIN	ZDECSRST	ZDECSSCX	ZDECSSCY	ZDECSSET	ZDECURCL	ZDECURLN	ZDEEQBFL	ZDEFLAG1	ZDEFLAG2
	ZDEFLAG4	ZDEFLEOF	ZDEFLPRX	ZDEFLSIZ	ZDEFLTOF	ZDEFMODE	ZDEFNAME	ZDEFTYPE	ZDEFWPTR	ZDELLPRX	ZDELRECL	ZDELSCPT	ZDELZDEB
	ZDELZDED	ZDEMSGMD	ZDEMSKLN	ZDENBTBC	ZDENSPAN	ZDEPRSPT	ZDEQMBUF	ZDEQMPTR	ZDERDALL	ZDERDINP	ZDERDNCH	ZDERDNUM	ZDESBCOM
	ZDESC	ZDESCABV	ZDESCBLW	ZDESERIN	ZDESFLG1	ZDESFLG2	ZDESFLG3	ZDESFLG4	ZDESTYLN	ZDETABCL	ZDETOPPT	ZDEVERCL	ZDEVERC1
	ZDEVERON	ZDEVERTR	ZDEZDEPT	ZDEZONEL	ZFOALARM	ZFOCLRLG	ZFOCLRSC	ZFOCMBLK	ZFOCMCNT	ZFOCMPNG	ZFOCSRAD	ZFOCSRFL	ZFOCURFL
	ZFOC3270	ZFOEDGON	ZFOFLAG1	ZFOFLAG2	ZFOFLAG3	ZFOFMODE	ZFOFNAME	ZFOFTYPE	ZFOINVCM	ZFOIOTBL	ZFOLFLAG	ZFOLFWPT	ZFOLGOP1
	ZFOLGOP2	ZFOLGOP3	ZFOLNAME	ZFOLNDSP	ZFOLRBUF	ZFOLSCPT	ZFOLSTSV	ZFOMCRNG	ZFOMOVUP	ZFOMSGCT	ZFONC	ZFONCOLS	ZFONROWS
	ZFOOPABS	ZFOOPFL1	ZFOOPFL2	ZFOOPFL3	ZFOOPNB1	ZFOOPNB2	ZFOOPNB3	ZFOOPNUM	ZFOOPST1	ZFOOPST2	ZFOOPST3	ZFOOPTGT	ZFOPLIST
	ZFORDBUF	ZFORTBYT	ZFORTCOD	ZFOSAVNB	ZFOSAVSV	ZFOSCRRD	ZFOSUBCM	ZFOSUSED	ZFOTWRMD	ZFOXER	ZFOXSUFL	ZFOZMAPT	ZMACST
	ZMAFNAME	ZMAFWPTR											
DMSXDC	AREA	DMSXFCPL	DMSXFDTG	DMSXMAOP	DMSXSUCH	DMSXSUCN	DMSXSUNP	DMSXSURV	DMSXSUTY	DMSXTBHC	D1	D2	D3
	MAXBUFLG	MSG	NUCON	REQADR	REQDES	REQLGMIN	REQLNAME	REQNAME	REQNBOPR	REQPARM1	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SAVBWPTR	SAVBYTE1	SAVBYTE2	SAVDWRD1	SAVDWRD2	SAVEREG	SAVLSAVB	SAVREG0	SAVREG1	SAVWORD1	SYNABBR	SYNARGS	SYNARG3
	SYNBUFF	SYNFWPTR	SYNLSYND	SYNMAXAR	SYNNAME	SYNNARG	SYNNBTYP	SYNOBUFF	SYNOSYNL	SYNSUB	SYNSYNL	ZDECLTGT	ZDECMSON
	ZDECURLN	ZDEEQBFL	ZDEEQBUF	ZDEFLAG2	ZDEFLAG3	ZDEFLAG4	ZDEHEXON	ZDEMCRON	ZDESBCOM	ZDESC	ZDESFLG2	ZDESTYLN	ZDESYNON
	ZDETAYON	ZDEWRPON	ZFOEDGON	ZFOFLAG3	ZFOFLAG4	ZFOIMPCM	ZFOLGOP1	ZFOLGOP2	ZFOLSTSV	ZFONC	ZFOOPABS	ZFOOPFL1	ZFOOPNB1
	ZFOOPNSP	ZFOOPNUM	ZFOOPSTR	ZFOOPST1	ZFOOPST2	ZFOOPTGT	ZFOPLIST	ZFOPRVTB	ZFORTBYT	ZFORTCOD	ZFOSUBCM	ZFOSYNBF	ZFOSYNBL
	ZFOSYNPT	ZFOWKBUF	ZFOXER	ZFOXSUFL									
DMSXDS	DMSXFCIN	DMSXFCSU	D1	FCBDD	FCBDSNAM	FCBFIRST	FCBRECL	FCBSECT	INPUT	NUCON	PS	R0	R1
	R10	R11	R12	R13	R14	R15	R2	R5	R8	R9	SAVBWPTR	SAVEREG	SAVLSAVB
	SAVREG0	SAVREG15	ZDEACURL	ZDEFLAG4	ZDEFNAME	ZDELRECL	ZDEMSGMD	ZDESC	ZDETRUNC	ZDEWIDTH	ZDEZONER	ZFOLGOP7	ZFOLGOP8
	ZFOLSTSV	ZFONC	ZFOOPNB1	ZFORDBUF	ZFOXER	ZFOXSUFL							
DMSXED	AREA	ASCANN	DMSXINTF	DMSXSDUP	DMSXSTCP	DMSXSUCK	DMSXSULK	DMSXSUPE	DMSXSURV	D1	D2	D3	D6
	FSTD	FSTFMODE	LSCFWPTR	LSCREEN	LSCZDEPT	L18	MAXBUFLG	MSG	NUCON	RESET	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SAVBWPTR	SAVBYTE1	SAVBYTE3	SAVEREG	SAVLSAVB	SAVREG0	SAVREG12	SAVWORD1	TYPE	XFF	ZDEBWPTR	ZDECSRST	ZDEDELPT
	ZDEFLAG3	ZDEFLSIZ	ZDEFMODE	ZDEFNAME	ZDEFTYPE	ZDEFWPTR	ZDELSCPT	ZDELZDEB	ZDELZDED	ZDEPFKPT	ZDEPRSPT	ZDEQMBUF	ZDEQMPTR
	ZDESBCOM	ZDESC	ZDESFLG2	ZDESFLG3	ZDETOPPT	ZDEUPDON	ZDEZDEPT	ZFOABUFF	ZFOCLRLG	ZFOCSRFL	ZFOCURFL	ZFOC3270	ZFOFLAG1
	ZFOFLAG2	ZFOFLAG3	ZFOFNAME	ZFOFREPT	ZFOLADDR	ZFOLBWPT	ZFOLDSCR	ZFOLFWPT	ZFOLGOP1	ZFOLNAME	ZFOLRBUF	ZFOLSCPT	ZFOLSTSV
	ZFOMCRNG	ZFONC	ZFONFILE	ZFOOPABS	ZFOOPFL1	ZFOOPST1	ZFOPROFL	ZFORDBUF	ZFORTCOD	ZFOSAVNB	ZFOSAVSV	ZFOSUBCM	ZFOWKBUF
	ZFOXER	ZFOXSUFL											
DMSXER	DMSXIOWR	DMSXSUCC	DMSXSUHC	D1	D2	MAXBUFLG	MSGADDR	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVBWPTR	SAVBYTE1	SAVBYTE2
	SAVEREG	SAVLSAVB	SAVREG0	SAVREG1	SAVREG14	SAVWORD1	XFF	ZDECFILL	ZDECGCNT	ZDEFLAG3	ZDEFLAG4	ZDEFMODE	ZDEFNAME
	ZDEFTYPE	ZDELRECL	ZDEPCKON	ZDERECFM	ZDESC	ZDESHMSG	ZDETRUNC	ZDEWIDTH	ZFOALARM	ZFOFLAG1	ZFOFLAG2	ZFOLSTSV	ZFOMCRNG
	ZFONC	ZFONFILE	ZFOWKBUF	ZFOXER	ZFOXSUFL								

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
DMSXFC	DMSXCN	DMSXSTLG	DMSXSUCC	DMSXSUCN	DMSXSUIG	DMSXUPDL	D1	D2	D3	LOC	LSCFEOF	LSCFPROT	LSCFTOP
	LSCPFXLG	LSCREEN	LSCRFLG1	LSCRFLNE	LSCRINPU	LSCRLGTH	LSCRLINE	LSCRLTBL	LSCRSIZE	LSCRWIDT	MAXBUFLG	PT	R0
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7
	R8	R9	SAVBWPTR	SAVBYTE1	SAVEREG	SAVLSAVB	SAVREG0	SAVREG2	SAVWORD1	ZDEACURD	ZDEACURL	ZDECANON	ZDECFILL
	ZDECLPON	ZDECSCOL	ZDECSLIN	ZDECSSCX	ZDECSSCY	ZDECURCL	ZDECURLN	ZDEENDRG	ZDEFLAG1	ZDEFLAG2	ZDEFLAG3	ZDEFLAG5	ZDEFLEOF
	ZDEFLLEF	ZDEFLRIG	ZDEFLSIZ	ZDEFLTOF	ZDEIMGON	ZDELRECL	ZDELSCPT	ZDENBTBC	ZDEPRFON	ZDEPRFRG	ZDESC	ZDESCABV	ZDESCBLW
	ZDESFLG2	ZDESHFLG	ZDETABCL	ZDETOPPT	ZDETOPRG	ZDETRUNC	ZDEUPDON	ZDEVERC1	ZDEVERTR	ZDEWIDTH	ZDEZDEPT	ZDEZONEL	ZDEZONER
	ZDE2INPT	ZFOCLRLG	ZFOC2741	ZFOC3270	ZFOFLAG1	ZFOFLAG2	ZFOFLAG3	ZFOFREPT	ZFOLADDR	ZFOLBWPT	ZFOLDSCR	ZFOLFLAG	ZFOLFWPT
	ZFOLNAME	ZFOLNCHG	ZFOLNCUR	ZFOLNDSP	ZFOLNNEW	ZFOLSTSV	ZFOMOVUP	ZFONC	ZFOOPST8	ZFORTBYT	ZFOTABS1	ZFOWKBUF	ZFOXER
	ZFOXSUFL												
DMSXFD	AADTLKP	ADTEDF	ADTFLG4	ADTSECT	AERASE	AFINIS	AWRBUF	DMSXCN	DMSXFCML	DMSXFCNX	DMSXFCPC	DMSXFCPL	DMSXSDUP
	DMSXSUCH	DMSXSUCN	DMSXUPBL	D1	D2	D3	FSTEPL	MAXBUFLG	MSG	NUCON	RESET	R0	R1
	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	SAVBWPTR	SAVBYTE1	SAVBYTE2	SAVBYTE4	SAVDWRD1	SAVDWRD2	SAVEREG	SAVLSAVB	SAVREG0	SAVREG1	SAVREG15	SAVREG8
	SAVWORD1	SAVWORD2	ZDEACURD	ZDEACURL	ZDEARBCH	ZDEARBON	ZDECANON	ZDECASRI	ZDECGCNT	ZDECLTGT	ZDECLTG1	ZDECLTG2	ZDECTLON
	ZDECURCL	ZDECURLN	ZDEENDRG	ZDEFLAG1	ZDEFLAG2	ZDEFLAG3	ZDEFLAG4	ZDEFLEOF	ZDEFLLEF	ZDEFLRIG	ZDEFLSIZ	ZDEFLTOF	ZDEHEXON
	ZDELRECL	ZDENSPAN	ZDEPCKON	ZDERECFM	ZDESC	ZDESCABV	ZDESCBLW	ZDESERCH	ZDESERIN	ZDESERLG	ZDESERST	ZDESFLG2	ZDESPABN
	ZDESPNON	ZDESTMON	ZDESTYLN	ZDETOPPT	ZDETOPRG	ZDETRUNC	ZDEUPDON	ZDEVRBON	ZDEWRPON	ZDEZONEL	ZDEZONER	ZFOABUFF	ZFOAITNO
	ZFOANOIT	ZFOFLAG	ZFOFLAG3	ZFOFMODE	ZFOFNAME	ZFOFTYPE	ZFOLADDR	ZFOLBUFF	ZFOLFWPT	ZFOLGOP5	ZFOLGOP8	ZFOLNAME	ZFOLSTSV
	ZFOMOVUP	ZFONC	ZFOOPNB7	ZFOOPNEG	ZFOOPST1	ZFOOPST2	ZFOOPST3	ZFOPLIST	ZFORECFM	ZFORTCOD	ZFOWKBUF	ZFOWKTBL	ZFOXER
	ZFOXSUFL	ZPACKBL	ZPAFLAG1	ZPALZPAD	ZPAPKBUF	ZPAPKBXE	ZPAPKBX2	ZPAPKBX3	ZPAPKBX4	ZPAPKCC	ZPAPKDAF	ZPAPKELF	ZPAPKERF
	ZPAPKFFF	ZPAPKSCF											
DMSXGT	AFINIS	ARDBUF	DMSXFCIN	DMSXFCSU	DMSXFCUP	DMSXSUCK	DMSXSUPR	DMSXSURV	D1	D3	FSTAIC	FSTD	FSTEPL
	FSTLRECL	FSTRECFM	MAXBUFLG	NUCON	RESET	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R5	R6	R7	R8	R9	SAVBWPTR	SAVBYTE1	SAVEREG	SAVLSAVB	SAVREG0	ZDEACURL
	ZDECGCNT	ZDEFLAG1	ZDEFLEOF	ZDEFNAME	ZDEFTYPE	ZDELRECL	ZDESC	ZDETRUNC	ZFOABUFF	ZFOAITNO	ZFOANOIT	ZFOFLAG	ZFOFMODE
	ZFOFNAME	ZFOFTYPE	ZFOLBUFF	ZFOLSTSV	ZFONC	ZFOOPABS	ZFOOPFL1	ZFOOPFL2	ZFOOPFL3	ZFOOPFL4	ZFOOPNB3	ZFOOPNB4	ZFOOPNB5
	ZFOOPNSP	ZFOOPNUM	ZFOOPSTR	ZFOOPST1	ZFOOPST2	ZFOOPST3	ZFOOPST8	ZFOPLIST	ZFORECFM	ZFORTBYT	ZFORTCOD	ZFOXER	ZFOXSUFL
DMSXHL	ASCANN	DMSXSDUP	DMSXSETB	DMSXSUCK	DMSXTBRQ	D1	MAXBUFLG	NUCON	REQDES	REQLGMIN	REQNAME	REQNBOPR	RESET
	R0	R1	R10	R11	R12	R13	R14	R15	R2	R5	R6	R7	R8
	SAVBWPTR	SAVDWRD1	SAVEREG	SAVLSAVB	SAVREG0	ZDESC	ZFOCLRLG	ZFOFLAG2	ZFOLGOP1	ZFOLGOP2	ZFOLSTSV	ZFONC	ZFOOPABS
	ZFOOPFL1	ZFOOPFL2	ZFOOPNSP	ZFOOPST1	ZFOOPST2	ZFOOPST8	ZFOPLIST	ZFORTCOD	ZFOSUBCM	ZFOXSUFL			
DMSXIN	ADTFLG2	ADTFROS	ADTM	AFINIS	AFVS	ARDBUF	ASCANN	DESFTYPE	DESLDESB	DESLRECL	DESRECFM	DESSER	DESSPEC
	DESTABS	DESTRUNC	DESTYP	DESVERIF	DMSXDSRD	DMSXFCIN	DMSXFCLR	DMSXFCSU	DMSXFCTB	DMSXMAOP	DMSXMARD	DMSXSTEX	DMSXSTLG
	DMSXSTNB	DMSXSUCK	DMSXSUCN	DMSXSUEX	DMSXSURV	DMSXSUTS	DMSXUPAT	DMSXUPCK	DMSXUPCT	D1	D2	D3	D6
	FRSTLOC	FSTAIC	FSTD	FSTEPL	FSTFMODE	FSTLRECL	FSTRECFM	FVSECT	FVSFSTAD	F0	LASTLOC	L18	MAXBUFLG
	MSG	MSGFLAGS	NOTYPING	NUCON	RESET	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	SAVBWPTR	SAVBYTE1	SAVBYTE2	SAVBYTE3	SAVBYTE4
	SAVDWRD1	SAVDWRD2	SAVEREG	SAVLSAVB	SAVREG0	SAVREG1	SAVREG15	SAVWORD1	SAVWORD2	SCBLOCK	SCBWKWRD	SYNABBR	SYNARGS
	SYNARG3	SYNBUFF	SYNFWPTR	SYNLSYND	SYNNAME	SYNNARG	SYNOBUFF	SYNOSYNL	SYNSUB	SYNSYNL	XFF	ZDEACURD	ZDEACURL
	ZDEARBCH	ZDEATCNT	ZDECANON	ZDECASRI	ZDECFILL	ZDECGCNT	ZDECLPON	ZDECMSON	ZDECSCOL	ZDECSLIN	ZDECTLON	ZDECURCL	ZDECURLN
	ZDEENDRG	ZDEESCON	ZDEFCURL	ZDEFINPU	ZDEFLAG1	ZDEFLAG2	ZDEFLAG3	ZDEFLAG4	ZDEFLAG5	ZDEFLEOF	ZDEFLPRX	ZDEFLSIZ	ZDEFLTOF

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	ZDEFMODE	ZDEFNAME	ZDEFSINP	ZDEFTYPE	ZDEIMGON	ZDELLPRX	ZDELNDON	ZDELNEND	ZDELRECL	ZDELSCPT	ZDEMRGUP	ZDEMSGMD	ZDEMSKLN
	ZDENSPAN	ZDEOSDSN	ZDEPCKON	ZDEPFKPT	ZDEPRFON	ZDEPRFRG	ZDERECFM	ZDESBCOM	ZDESC	ZDESCABV	ZDESCBLW	ZDESCLON	ZDESERCH
	ZDESERIN	ZDESERLG	ZDESERST	ZDESFLG2	ZDESIDCD	ZDESIDON	ZDESPABN	ZDESQ8ON	ZDESTMON	ZDESYNON	ZDETABCL	ZDETFLIN	ZDETOPPT
	ZDETOPRG	ZDETRUNC	ZDEUPDON	ZDEUPINC	ZDEVERC1	ZDEVERC2	ZDEVERON	ZDEWIDTH	ZDEZONEL	ZDEZONER	ZDE2INPT	ZFOABUFF	ZFOAITNO
	ZFOANOIT	ZFOBLKCR	ZFOBLKPT	ZFOCURFL	ZFOC3270	ZFOEDGON	ZFOFLAG	ZFOFLAG1	ZFOFLAG3	ZFOFNAME	ZFOFTYPE	ZFOLADDR	ZFOLBUFF
	ZFOLBWPT	ZFOLDSCR	ZFOLFLAG	ZFOLFWPT	ZFOLGOP8	ZFOLRBUF	ZFOLSTSV	ZFOMSGCT	ZFONC	ZFONFILE	ZFONOBRD	ZFOOPNB1	ZFOOPNB3
	ZFOOPNB5	ZFOPLIST	ZFOPRFER	ZFOPROFL	ZFORDBUF	ZFORECFM	ZFORETMC	ZFORTCOD	ZFOSAV01	ZFOSUBCM	ZFOSYNPT	ZFOTWRMD	ZFOWKBUF
	ZFOWKTBL	ZFOXER	ZFOXSUFL	ZPACKBL	ZPAFLAG1	ZPALZPAD	ZPAPKBUF	ZPAPKBXE	ZPAPKDAF	ZPAPKELF	ZPAPKERF	ZPAPKFFF	ZPAPKFIL
	ZPAPKSCF												
DMSXIO	DMSXSCDP	DMSXSUFL	D1	LSCREEN	LSCRMSG	LSCRWIDT	MAXBUFLG	NUCON	NUMFINRD	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R7	R8	R9	SAVBWPTR	SAVDWRD1	SAVEREG	SAVLSAVB
	SAVREG0	SAVREG1	ZDECASMU	ZDEFLAG2	ZDEFLAG4	ZDELSCPT	ZDEMSBFL	ZDEMSBUF	ZDEMSGMD	ZDESC	ZDESFLG2	ZDEVERON	ZDEWRMSG
	ZFOALARM	ZFOCLRSC	ZFOC3270	ZFOEDGON	ZFOFLAG1	ZFOFLAG2	ZFOFLAG3	ZFOIOTBL	ZFOLSTSV	ZFOMSGCT	ZFONC	ZFONCOLS	ZFORDBUF
	ZFOSCRRD	ZFOXER	ZFOXSUFL										
DMSXMA	ADMSBLKR	ADTDBSIZ	ADTEDF	ADTFLG4	AESTATE	AFINIS	AFVS	ARDBUF	AREA	ASVCSECT	CURRALOC	DMSXSUEX	D1
	D2	D3	EGPR15	FSTADBC	FSTAIC	FSTD	FSTEPL	FSTLRECL	FSTRECFM	FVSECT	FVSFSTAD	LSCREEN	LSCRMSG
	LSCRWIDT	MAXBUFLG	MSG	MSGFLAGS	NOTYPING	NUCON	RECEXELG	RECEXEPT	RECEXTLG	RECFMODE	RECFNAME	RECFTYPE	RECLGHDR
	RECLRECD	RECMCBFL	RECMCBUF	RECPLIST	RECSAVE	RECSAVRG	RECSAV13	RECZMAPT	RESET	R0	R1	R10	R11
	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVBWPTR
	SAVBYTE1	SAVEREG	SAVLSAVB	SAVREG0	SAVREG12	SAVREG2	SCBLOCK	SCBWKWRD	SSAVE	SVCSECT	TYPE	ZDELSCPT	ZDESC
	ZDESFLG2	ZDEWRMSG	ZFOABUFF	ZFOAITNO	ZFOANOIT	ZFOCURFL	ZFOCURSV	ZFOFLAG	ZFOFLAG1	ZFOFLAG3	ZFOLBUFF	ZFOLRBUF	ZFOLSTSV
	ZFOMCRNG	ZFOMSGCT	ZFONC	ZFONOBRD	ZFOPLIST	ZFOPROFL	ZFORDBUF	ZFORECFM	ZFORECPT	ZFORTCOD	ZFOSAVNB	ZFOSAVSV	ZFOSAV01
	ZFOSAV02	ZFOSUBCM	ZFOWKBUF	ZFOXER	ZFOXSUFL	ZFOZMAPT	ZMACST	ZMAEXELG	ZMAEXEPT	ZMAFNAME	ZMAFWPTR	ZMALZMAD	ZMAMCLST
	ZMANBMST												
DMSXMC	DMSXFCPC	DMSXFCPL	DMSXSUCN	DMSXSURV	DMSXSUTY	D1	D3	LSCREEN	LSCRSIZE	LSCRWIDT	R0	R1	R10
	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9
	SAVBWPTR	SAVBYTE1	SAVEREG	SAVLSAVB	SAVREG0	SAVWORD1	ZDECLTGT	ZDECURCL	ZDECURLN	ZDEFLAG2	ZDELRECL	ZDELSCPT	ZDENBVRC
	ZDESC	ZDEVERCI	ZDEVERON	ZDEVERTR	ZDEZONEL	ZDEZONER	ZFOCLRLG	ZFOC3270	ZFOFLAG1	ZFOFLAG2	ZFOLGOP1	ZFOLSTSV	ZFONC
	ZFOOPNB1	ZFOOPST1	ZFORTBYT	ZFOSUBCM	ZFOWKBUF	ZFOXER	ZFOXSUFL						
DMSXMD	DMSXFCIN	DMSXFCNX	DMSXFCPL	DMSXFCRC	DMSXFCRL	DMSXFCSP	DMSXFCSU	DMSXFCUP	DMSXIORD	DMSXSCIM	DMSXSUEX	DMSXSUNC	DMSXSUPP
	DMSXSUTY	D1	D3	INPUT	LSCRAINP	LSCRCURL	LSCREEN	LSCRINPU	LSCRSIZE	MAXBUFLG	NUCON	NUMFINRD	R0
	R1	R10	R11	R12	R13	R14	R15	R2	R3	R5	R6	R7	R8
	R9	SAVBWPTR	SAVBYTE1	SAVEREG	SAVLSAVB	SAVREG0	SAVWORD1	SAVWORD2	ZDEACURD	ZDEATCNT	ZDEATSID	ZDEATSMD	ZDECESCA
	ZDECGCNT	ZDECSCOL	ZDECSLIN	ZDECSRST	ZDECSSET	ZDECURLN	ZDEESCON	ZDEFLAG1	ZDEFLAG3	ZDEFLAG5	ZDEFLEOF	ZDEFLRIG	ZDEFLSIZ
	ZDEFLTOF	ZDEFSINP	ZDEINHLD	ZDELSCPT	ZDELSTCG	ZDEMSKLN	ZDEPFCOD	ZDEPFKEY	ZDEPFKPT	ZDEPRFON	ZDEPRFRG	ZDESBCOM	ZDESC
	ZDESCHGD	ZDESFLG2	ZDESFLG3	ZDETABCL	ZDEUPDON	ZFOALARM	ZFOCLRLG	ZFOCSRFL	ZFOC3270	ZFOEDGON	ZFOFLAG1	ZFOFLAG2	ZFOFLAG3
	ZFOFREPT	ZFOLFLAG	ZFOLFWPT	ZFOLGOP1	ZFOLRBUF	ZFOLSTSV	ZFOMSGCT	ZFONC	ZFOOPABS	ZFOOPFL1	ZFOOPFL2	ZFOOPFL3	ZFOOPNB1
	ZFOOPSTR	ZFOOPST1	ZFOOPST2	ZFOOPST3	ZFOPLIST	ZFOPRFIN	ZFORDBUF	ZFORTBYT	ZFORTCOD	ZFOXER	ZFOXSUFL		
DMSXML	DMSXFCLM	DMSXFCML	DMSXFCNX	DMSXFCPL	DMSXFCSP	DMSXSDSC	DMSXSUEF	DMSXSUNF	DMSXSUTE	DMSXSUTF	DMSXSUTY	LSCFEOF	LSCFPROT
	LSCREEN	LSCRFLG1	LSCRFLNE	LSCRINPU	LSCRLINE	LSCRLTBL	LSCRNBLN	LSCRSIZE	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R8	R9	SAVBWPTR	SAVEREG	SAVLSAVB

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	SAVREGO	ZDEACURD	ZDEACURL	ZDECURLN	ZDEENDRG	ZDEFLAG1	ZDEFLAG2	ZDEFLEOF	ZDEFLSIZ	ZDEFLTOF	ZDELSCPT	ZDESC	ZDESCABV
	ZDESCBLW	ZDESFLG2	ZDESHFLG	ZDESTYLN	ZDETOPPT	ZDETOPRG	ZDEWRPON	ZFOC3270	ZFOFLAG1	ZFOFLAG3	ZFOLADDR	ZFOLBWPT	ZFOLGOP1
	ZFOLSTSV	ZFOMOVUP	ZFONC	ZFOOPNB1	ZFOOPST1	ZFORTBYT	ZFORTCOD	ZFOSUBCM	ZFOXER	ZFOXSUFL			
DMSXMS	CZERO	D1	D2	D3	LOOP	MSG	NUCON	R0	R1	R10	R11	R12	R13
	R14	R15	R2	R3	R4	R5	R6	R7	R8	R9	SCBLOCK	SCBWKWRD	XFF
	ZDEACURD	ZDEACURL	ZDECASRI	ZDECGCNT	ZDECURLN	ZDEENDRG	ZDELRECL	ZDESC	ZDETOPPT	ZDETOPRG	ZFOCLRLG	ZFOCURFL	ZFOFLAG2
	ZFOLADDR	ZFOLBWPT	ZFOLFWPT	ZFONC									
DMSXPO	AINTRTBL	AOUTRTBL	ATRHIGH	ATRINIB	ATRMDT	ATRPRT	ATRRST	CSW	DMSXFCIN	DMSXFCUP	DMSXSCCN	DMSXSCIO	DMSXSDCT
	DMSXSDUP	DMSXSUPR	D1	D2	D3	IC	IONPSW	IOOPSW	KCLEAR	KLGTPEN	KPA1	KPA2	KPA3
	LSCFNPRF	LSCFPROT	LSCREEN	LSCRMSG	NUCON	RESET	R0	R1	R10	R11	R12	R13	R14
	R15	R2	R3	R4	R5	R6	R7	R8	R9	SAVBWPTR	SAVBYTE1	SAVBYTE2	SAVDWRD1
	SAVDWRD2	SAVEREG	SAVLSAVB	SAVREGO	SF	ZDEACURL	ZDECASMU	ZDECGCNT	ZDEFLAG1	ZDEFLAG3	ZDEFLEOF	ZDEFLTOF	ZDELNDON
	ZDELNEND	ZDELSCPT	ZDESC	ZDESFLG2	ZDETRUNC	ZDEWRMSG	ZFOALARM	ZFOATERM	ZFOBUFIO	ZFOCLRLG	ZFOC3270	ZFOERCOD	ZFOFLAG1
	ZFOFLAG2	ZFOIOTBL	ZFOLGSCB	ZFOLSTSV	ZFONC	ZFONCOLS	ZFOOPNB1	ZFORMTUB	ZFORTBYT	ZFOWKBUF	ZFOXER	ZFOXSUFL	
DMSXPT	AERASE	AFINIS	AWRBUF	DMSXFCML	DMSXFCSU	DMSXIORD	DMSXSUCK	DMSXSUPR	DMSXSURV	DMSXSUTE	DMSXSUTP	D1	D3
	FSTAIC	FSTD	FSTEPL	FSTFLAGS	FSTLRECL	FSTRECFM	FSTRWDSK	L18	MAXBUFLG	NUCON	RESET	R0	R1
	R10	R11	R12	R13	R14	R15	R2	R3	R5	R6	R7	R8	R9
	SAVBWPTR	SAVBYTE1	SAVEREG	SAVLSAVB	SAVREGO	ZDEACURL	ZDEFLAG1	ZDEFLEOF	ZDEFLSIZ	ZDEFLTOF	ZDEFMODE	ZDEFNAME	ZDEFTYPE
	ZDELRECL	ZDERECFM	ZDESC	ZFOABUFF	ZFOAITNO	ZFOANOIT	ZFOBLKPT	ZFOFLAG	ZFOFMODE	ZFOFNAME	ZFOFREPT	ZFOFTYPE	ZFOLADDR
	ZFOLBUFF	ZFOLFLAG	ZFOLSTSV	ZFONC	ZFOOPABS	ZFOOPFL2	ZFOOPFL3	ZFOOPFL4	ZFOOPNB1	ZFOOPNSP	ZFOOPST2	ZFOOPST3	ZFOOPST4
	ZFOOPST8	ZFOPLIST	ZFORECFM	ZFORTBYT	ZFORTCOD	ZFOXER	ZFOXSUFL						
DMSXPX	AREA	DMSXFCIN	DMSXFCNX	DMSXFCPC	DMSXFCPL	DMSXFCSU	DMSXFCUP	DMSXFDLN	DMSXSUCN	DMSXSUIG	LSCFCHGD	LSCFEOF	LSCFRSVD
	LSCFTOP	LSCFWPTR	LSCFWRAP	LSCPFXLG	LSCREEN	LSCRFADD	LSCRFLG1	LSCRFLG2	LSCRFLNE	LSCRIMAD	LSCRLINE	LSCRLTBL	LSCRMASK
	LSCRTABS	LSCRWIDT	LSCZDEPT	MSG	R0	R1	R10	R11	R12	R13	R14	R15	R2
	R3	R4	R5	R6	R7	R8	R9	SAVBWPTR	SAVBYTE1	SAVBYTE2	SAVBYTE4	SAVDWRD1	SAVEREG
	SAVLSAVB	SAVREGO	SAVREG1	ZDEABPNG	ZDEACURD	ZDECASMU	ZDECGCNT	ZDECLNSC	ZDECNFCT	ZDECSCOL	ZDECSERR	ZDECSINP	ZDECSLIN
	ZDECSSCX	ZDECSSCY	ZDECURLN	ZDEFLABL	ZDEFLAG1	ZDEFLAG2	ZDEFLAG5	ZDEFLDLN	ZDEFLEOF	ZDEFLNBL	ZDEFLPRX	ZDEFLSIZ	ZDEFLTOF
	ZDEINCPL	ZDEINVCM	ZDELLABL	ZDELLNBL	ZDELLPRX	ZDELNINV	ZDELRECL	ZDELSCPT	ZDEMCPNG	ZDEMSKLN	ZDENUMON	ZDEPRFEX	ZDEPRFRG
	ZDESBCOM	ZDESC	ZDESCHGD	ZDESCLON	ZDESFLG2	ZDESFLG4	ZDESYNON	ZDETABCL	ZDETBSON	ZDETOPPT	ZDETRUNC	ZDEVERC1	ZDEVERC2
	ZDEZDEPT	ZFOBUFIM	ZFOCLRLG	ZFOCMBLK	ZFOCMCNT	ZFOCMPNG	ZFOEDGON	ZFOFLAG2	ZFOFLAG3	ZFOFREPT	ZFOINVCM	ZFOKPLIN	ZFOLADDR
	ZFOLFLAG	ZFOLFWPT	ZFOLNAME	ZFOLNCUR	ZFOLNDSP	ZFOLSTSV	ZFONC	ZFONCOLS	ZFOOPST4	ZFOPRFIN	ZFOPRFPT	ZFOSCRRD	ZFOSOSIM
	ZFOXSUFL												
DMSXRE	AERASE	AFINIS	ARDBUF	AWRBUF	FSTEPL	MSG	NUCON	PLIST	R0	R1	R10	R11	R12
	R13	R14	R15	R2	R3	R4	R5	R6	R7	VCADTLKW	ZDEFLSIZ	ZDELRECL	ZDESC
	ZDETOPPT	ZFOLADDR	ZFOLFWPT										
DMSXSC	AINTRTBL	AOUTRTBL	ATRBIP	ATRHIGH	ATRINIB	ATRMDT	ATRPRT	BIPWCCMD	CSW	DMSXFCCC	DMSXFCCL	DMSXFCPL	DMSXPXDC
	DMSXPXEX	DMSXSDML	DMSXSDPH	DMSXSDSC	DMSXSDTY	DMSXSSEX	DMSXSUIG	D1	D2	D3	D6	EUA	FLDMRK
	IC	IONPSW	IOOPSW	KCLEAR	KENTER	KLGTPEN	KNOACT	KPA1	KPA2	KPA3	LSCARWLG	LSCFALTF	LSCFPROT
	LSCFTOP	LSCFWPTR	LSCFWRAP	LSCPFXLG	LSCRAINP	LSCRCTL	LSCRDSPH	LSCRDSPV	LSCREEN	LSCRFLG1	LSCRFLG2	LSCRFLNE	LSCRIBUF
	LSCRIMAD	LSCRINPU	LSCRLGTH	LSCRLINE	LSCRLTBL	LSCRMSG	LSCRSIZE	LSCRWIDT	LSCSTALG	LSCZDEPT	MAXBUFLG	MSGFLAGS	NOTYPING

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	NUCON	NUMFINRD	PT	RESET	RETURN	R0	R1	R10	R11	R12	R13	R14	R15
	R2	R3	R4	R5	R6	R7	R8	R9	SAVBWPTR	SAVBYTE1	SAVBYTE2	SAVBYTE3	SAVDWRD1
	SAVDWRD2	SAVEREG	SAVLSAVB	SAVREG0	SAVREG12	SAVREG2	SAVREG5	SAVWORD1	SAVWORD2	SBA	SENSE	SF	STDWCCMD
	ZDEATCNT	ZDEATSID	ZDEATSMD	ZDECGCNT	ZDECLNSC	ZDECSCOL	ZDECSERR	ZDECSINP	ZDECSLIN	ZDECSRST	ZDECSSCX	ZDECSSCY	ZDECSSET
	ZDECURLN	ZDEENDRG	ZDEFLAG1	ZDEFLAG5	ZDEFLDCL	ZDEFLDLN	ZDEFSINP	ZDEINHLD	ZDELNINV	ZDELSCPT	ZDELSTCG	ZDENBTBC	ZDENULON
	ZDEPFCOD	ZDEPFKEY	ZDEPFKPT	ZDEPRFEX	ZDEPRFON	ZDEPRFRG	ZDEPRFW1	ZDESBCOM	ZDESC	ZDESCABV	ZDESCBLW	ZDESCHGD	ZDESFLG2
	ZDESFLG3	ZDESFLG4	ZDETABCL	ZDETODSP	ZDETOPRG	ZDEVERC1	ZDEVERTR	ZDEWRMSG	ZDEZDEPT	ZDEZONEL	ZDEZONER	ZDE2INPT	ZFOALARM
	ZFOATERM	ZFOBUFIM	ZFOBUFIO	ZFOCLRLG	ZFOCLRSC	ZFOCSRAD	ZFOCSRFL	ZFOC3270	ZFOC3278	ZFOEDGON	ZFOERCOD	ZFOFKCOD	ZFOFLAG1
	ZFOFLAG2	ZFOFLAG3	ZFOIOCMP	ZFOIOTBL	ZFOKPLIN	ZFOKYCOD	ZFOLGSCB	ZFOLSCPT	ZFOLSTSV	ZFOMOVUP	ZFOMSGCT	ZFONC	ZFONCOLS
	ZFONROWS	ZFOOPFL1	ZFOOPFL2	ZFOOPFL3	ZFOOPNB1	ZFOOPSTR	ZFOOPST1	ZFOOPST2	ZFOOPST3	ZFOOPST4	ZFOOPST8	ZFOPLIST	ZFORDBUF
	ZFORDSVR	ZFORTCOD	ZFOSAVNB	ZFOSAVSV	ZFOSCRRD	ZFOSINDX	ZFOSOSIM	ZFOSUSED	ZFOTABS1	ZFOTWRMD	ZFOTXTON	ZFOWKBUF	ZFOWKTBL
	ZFOXER	ZFOXSUFL											
DMSXSD	AINTRTBL	AOUTRTBL	ATRHIGH	ATRMDT	ATRPRT	ATRRST	DMSXFCDC	DMSXFCPL	DMSXFCRL	DMSXFCSP	DMSXSUCC	FLDMRK	LSCARWLG
	LSCFALTF	LSCFAROW	LSCFCHGD	LSCFEOF	LSCFHIGH	LSCFMASK	LSCFMDT	LSCFNPRF	LSCFNULL	LSCFPROT	LSCFRST	LSCFRSVD	LSCFTABS
	LSCFTOP	LSCFWPTR	LSCFWRAP	LSCLLSCB	LSCPFXLG	LSCRAINP	LSCRBYTE	LSCRCTL	LSCRCTLP	LSCRCURL	LSCRDSPH	LSCRDSPV	LSCREEN
	LSCRFADD	LSCRFLG1	LSCRFLG2	LSCRFLNE	LSCRIMAD	LSCRINPU	LSCRLGTH	LSCRLINE	LSCRLTBL	LSCRMASK	LSCRMSG	LSCRNBLN	LSCRSIZE
	LSCRSTAT	LSCRTABS	LSCRULCO	LSCRWIDT	LSCSTALG	LSCZDEPT	MAXBUFLG	MSG	NUCON	PT	RA	R0	R1
	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	SAVBWPTR	SAVBYTE1	SAVBYTE2	SAVBYTE3	SAVBYTE4	SAVDWRD1	SAVDWRD2	SAVEREG	SAVLSAVB	SAVREG0	SAVREG1	SAVREG15
	SAVREG2	SAVREG3	SAVREG6	SAVREG7	SAVWORD1	SAVWORD2	SBA	SF	TYPE	ZDEABPNG	ZDEACURD	ZDEACURL	ZDECANON
	ZDECASMU	ZDECGCNT	ZDECLNSC	ZDECSINP	ZDECSLIN	ZDECSRST	ZDECSSET	ZDECURCL	ZDECURLN	ZDEENDRG	ZDEFLAG1	ZDEFLAG2	ZDEFLAG3
	ZDEFLAG4	ZDEFLAG5	ZDEFLDCL	ZDEFLDLN	ZDEFLSIZ	ZDEFSINP	ZDEIMGON	ZDEINCPL	ZDEINHLD	ZDELRECL	ZDELSCPT	ZDEMCPNG	ZDENBTBC
	ZDENBVRC	ZDENULON	ZDENUMON	ZDEPRFON	ZDEPRFRG	ZDERDALL	ZDERDINP	ZDERDNCH	ZDERDNUM	ZDESBCOM	ZDESC	ZDESCHGD	ZDESCLON
	ZDESFLG1	ZDESFLG2	ZDESFLG3	ZDESFLG4	ZDETABCL	ZDETBSON	ZDETFLIN	ZDETOPRG	ZDETRUNC	ZDEUPDON	ZDEVERCL	ZDEVERC1	ZDEVERTR
	ZDEWRMSG	ZDEZONEL	ZDEZONER	ZDE2INPT	ZFOBUFIM	ZFOBUFIO	ZFOCLRLG	ZFOCLRSC	ZFOCMBLK	ZFOCMCNT	ZFOCMPNG	ZFOCTLSZ	ZFOC3278
	ZFOEDGON	ZFOFKCOD	ZFOFLAG1	ZFOFLAG2	ZFOFLAG3	ZFOINVCM	ZFOIOCMP	ZFOIOTBL	ZFOKPLIN	ZFOLADDR	ZFOLBWPT	ZFOLFLAG	ZFOLFWPT
	ZFOLNCHG	ZFOLNDSP	ZFOLSCPT	ZFOLSTSV	ZFONC	ZFONCOLS	ZFONFILE	ZFOPLIST	ZFOPRFIN	ZFORMTUB	ZFOSCRRD	ZFOSINDX	ZFOSOSIM
	ZFOTXTON	ZFOUFLDS	ZFOWKBUF	ZFOWKTBL	ZFOXER	ZFOXSUFL							
DMSXSE	AADTLKP	ADTFLG1	ADTFRW	ATRHIGH	ATRPRT	DMSXCTPN	DMSXCTRS	DMSXCTSC	DMSXCTTE	DMSXDCOD	DMSXDCSY	DMSXFCBT	DMSXFCCC
	DMSXFCCL	DMSXFCDC	DMSXFCDR	DMSXFCGA	DMSXFCHT	DMSXFCLR	DMSXFCPL	DMSXFCTB	DMSXFCTR	DMSXIORD	DMSXIOWR	DMSXMAED	DMSXMCVR
	DMSXSCRV	DMSXSDAP	DMSXSDCT	DMSXSDMK	DMSXSDSC	DMSXSDTB	DMSXSDTX	DMSXSDUP	DMSXSSEX	DMSXSUCC	DMSXSUCK	DMSXSUCN	DMSXSULG
	DMSXSURV	FMODE	FNAME	FTYPE	LSCARWLG	LSCFRSVD	LSCFWPTR	LSCRCURL	LSCRDSPH	LSCRDSPV	LSCREEN	LSCRFLG2	LSCRINPU
	LSCRLINE	LSCRLTBL	LSCRMASK	LSCRSIZE	LSCRTABS	LSCRWIDT	MAXBUFLG	MSG	NUCON	NUMFINRD	PACK	R0	R1
	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5	R6	R7	R8
	R9	SAVBWPTR	SAVBYTE1	SAVDWRD1	SAVDWRD2	SAVEREG	SAVLSAVB	SAVREG0	SAVWORD1	SAVWORD2	SYNABBR	SYNFWPTR	SYNNAME
	SYNOBUFF	SYNOSYNL	SYNSUB	TYPE	ZDEACURD	ZDEACURL	ZDEARBCH	ZDEARBON	ZDEATCNT	ZDEATSID	ZDEATSMD	ZDECANON	ZDECASMU
	ZDECASRI	ZDECESCA	ZDECFILL	ZDECGCNT	ZDECLPON	ZDECMSON	ZDECSCOL	ZDECSLIN	ZDECSRST	ZDECSSCX	ZDECSSCY	ZDECURLN	ZDEEQBFL
	ZDEEQBUF	ZDEESCON	ZDEFCURL	ZDEFINPU	ZDEFLAG2	ZDEFLAG3	ZDEFLAG4	ZDEFLAG5	ZDEFLEOF	ZDEFLSIZ	ZDEFLTOF	ZDEFMASK	ZDEFMODE
	ZDEFNAME	ZDEFTABS	ZDEFTYPE	ZDEFWPTR	ZDEHEXON	ZDEIMGON	ZDELNDON	ZDELNEND	ZDELRECL	ZDELSCPT	ZDEMCRON	ZDEMSBFL	ZDEMSBUF
	ZDEMSGMD	ZDEMSKLN	ZDENBTBC	ZDENBVRC	ZDENSPAN	ZDENULON	ZDENUMON	ZDEPCKON	ZDEPFKPT	ZDEPRFON	ZDEPRFRG	ZDERECFM	ZDESC
	ZDESCABV	ZDESCBLW	ZDESCLON	ZDESERCH	ZDESERIN	ZDESERLG	ZDESERST	ZDESFLG2	ZDESFLG3	ZDESHMSG	ZDESPABN	ZDESPNON	ZDESQ8ON
	ZDESTMON	ZDESYNON	ZDETABCL	ZDETAYON	ZDETBSON	ZDETFLIN	ZDETOPPT	ZDETRUNC	ZDEUPDON	ZDEVERCL	ZDEVERON	ZDEVRBON	ZDEWIDTH
	ZDEWRPON	ZDE2INPT	ZFOAPLON	ZFOCLRLG	ZFOCSRSZ	ZFOC2741	ZFOC3215	ZFOC3270	ZFOC3278	ZFOEDGON	ZFOFLAG1	ZFOFLAG2	ZFOFLAG3
	ZFOFLAG4	ZFOFMODE	ZFOFNAME	ZFOFTYPE	ZFOHVCED	ZFOHVCP1	ZFOHVCP3	ZFOIMPCM	ZFOIOCMP	ZFOIOTBL	ZFOLFLAG	ZFOLFWPT	ZFOLGOP1

```
MODULE       EXTERNAL REFERENCES (LABELS AND MODULES)

             ZFOLGOP2 ZFOLGOP3 ZFOLGOP8 ZFOLNAME ZFOLNDSP ZFOLSCPT ZFOLSTSV ZFONC    ZFONCOLS ZFONDSPC ZFONROWS ZFOOPABS ZFOOPFL1
             ZFOOPFL2 ZFOOPFL3 ZFOOPNB1 ZFOOPNB2 ZFOOPNB3 ZFOOPNSP ZFOOPSTR ZFOOPST1 ZFOOPST2 ZFOOPST3 ZFOOPST7 ZFOOPST8 ZFOPLIST
             ZFOPRFPT ZFOPRVTB ZFORTBYT ZFORTCOD ZFOSOSIM ZFOSUBCM ZFOSVCED ZFOSVC04 ZFOSYNBF ZFOSYNPT ZFOTWRMD ZFOTXTON ZFOUFLDS
             ZFOWKBUF ZFOXER   ZFOXSUFL

DMSXSG       ACMSSEG  ASYSNAMS CMSSEG   DCSSAVAL DCSSFLAG DCSSLDED LASTLMOD LOCCNT   MSGFLAGS NOTYPING NUCON    R0       R1
             R10      R13      R14      R15      R3       R4       R5       R7       R8       R9       STRTADDR SUBACT   SUBFLAG
             SYSNAMES SYSNEND

DMSXSS       DMSXFCPL DMSXFCSU DMSXPXDC DMSXPXEX DMSXSCPR DMSXSDML DMSXSUCC DMSXSUCN DMSXSUEX DMSXSUPR LASTCMND LSCARWLG LSCFALTF
             LSCFAROW LSCFCHGD LSCFMDT  LSCFNPRF LSCFNULL LSCFPROT LSCFRST  LSCFWPTR LSCFWRAP LSCPFXLG LSCRDSPH LSCREEN  LSCRFADD
             LSCRFLG1 LSCRFLG2 LSCRFLNE LSCRIBUF LSCRIMAD LSCRINPU LSCRLGTH LSCRLINE LSCRLTBL LSCRSIZE LSCRWIDT LSCZDEPT MAXBUFLG
             NUCON    NUMFINRD R0       R1       R10      R11      R12      R13      R14      R15      R2       R3       R4
             R5       R6       R7       R8       R9       SAVBWPTR SAVBYTE1 SAVBYTE2 SAVEREG  SAVLSAVB SAVREG0  SAVREG1  SAVREG12
             SAVWORD1 SF       SUBACT   SUBFLAG  ZDECSLIN ZDECSRST ZDECSSCX ZDECSSCY ZDECSSET ZDECURLN ZDEFLAG2 ZDEFLAG5 ZDEFLDLN
             ZDELSCPT ZDENBTBC ZDENUMON ZDEPFKPT ZDEPRFEX ZDEPRFON ZDEPRFRG ZDESC    ZDESCHGD ZDESFLG2 ZDESFLG3 ZDESFLG4 ZDETABCL
             ZDETODSP ZDEVERC1 ZDEVERTR ZFOALARM ZFOBUFIM ZFOCLRLG ZFOCLRSC ZFOCMBLK ZFOCMCNT ZFOCMPNG ZFOCSDPT ZFOCSRAD ZFOCSRFL
             ZFOCSRSZ ZFOCSSTK ZFOCURFL ZFOEDGON ZFOFLAG2 ZFOFLAG3 ZFOFREPT ZFOINVCM ZFOKPLIN ZFOLFLAG ZFOLGOP1 ZFOLRBUF ZFOLSCPT
             ZFOLSTSV ZFONC    ZFONCCLS ZFONROWS ZFOOPNB1 ZFOOPST1 ZFOPLIST ZFORDBUF ZFORTBYT ZFOSAVNB ZFOSAVSV ZFOSOSIM ZFOWKBUF
             ZFOXER   ZFOXSUFL

DMSXST       MSG      R0       R1       R10      R11      R12      R13      R14      R15      R2       R3       R4       R5
             R6       R7       R8       R9       SAVBWPTR SAVEREG  SAVLSAVB SAVREG0  SAVREG2  SAVREG3  ZDESC    ZDEWIDTH ZFOBLKCR
             ZFOBLKPT ZFOFREPT ZFOLADDR ZFOLDSCR ZFOLFLAG ZFOLFWPT ZFOLSTSV ZFONC    ZFOXSUFL

DMSXSU       AESTATE  DMSXDCOD DMSXEDRT DMSXFCDP DMSXFCNX DMSXFCPL DMSXFCUP DMSXINLD DMSXINTF DMSXIORD DMSXIOWR DMSXPTER DMSXSCRV
             FSTD     FSTFMODE FSTFNAME LSCARWLG LSCFWPTR LSCRCURL LSCREEN  LSCRINPU LSCRMASK LSCRSIZE LSCRTABS LSCZDEPT MAXBUFLG
             MSGFLAGS NOTYPING NUCON    R0       R1       R10      R11      R12      R13      R14      R15      R2       R3
             R4       R5       R6       R7       R8       R9       SAVBWPTR SAVBYTE1 SAVBYTE2 SAVDWRD1 SAVEREG  SAVLSAVB SAVREG0
             SAVREG1  SAVREG10 SAVREG15 SAVREG2  ZDEABPNG ZDEACURL ZDEARBCH ZDEARBON ZDEATCNT ZDEATSID ZDEATSMD ZDECGCNT ZDECLNSC
             ZDECSSCX ZDECSSCY ZDECSSET ZDECURLN ZDEDELPT ZDEENDRG ZDEEQBFL ZDEEQBUF ZDEFCURL ZDEFINPU ZDEFLAG1 ZDEFLAG2 ZDEFLAG3
             ZDEFLAG4 ZDEFLAG5 ZDEFLEOF ZDEFLPRX ZDEFLSIZ ZDEFLTOF ZDEFMASK ZDEFNAME ZDEFTABS ZDEINCPL ZDELLPRX ZDELNDON ZDELNEND
             ZDELRECL ZDELSCPT ZDEMCPNG ZDEMSBUF ZDEMSGMD ZDEPFKPT ZDEQMBUF ZDEQMPTR ZDESBCOM ZDESC    ZDESFLG2 ZDESFLG3 ZDESFLG4
             ZDESTYLN ZDETAYON ZDETOPPT ZDETOPRG ZDETRUNC ZDEVERCL ZDEVERC1 ZDEVERON ZDEVERTR ZDEWRMSG ZDEZDEPT ZDE2INPT ZFOABUFF
             ZFOATSID ZFOCLRLG ZFOC3270 ZFOEDGON ZFOFLAG1 ZFOFLAG2 ZFOFLAG3 ZFOFMODE ZFOFNAME ZFOFTYPE ZFOLRBUF ZFOLSCPT ZFOLSTSV
             ZFOMCRNG ZFOMOVUP ZFONC    ZFONFILE ZFOOPABS ZFOOPFL1 ZFOOPFL2 ZFOOPFL3 ZFOOPNB1 ZFOOPSTR ZFOOPST1 ZFOOPST2 ZFOOPST3
             ZFOPLIST ZFOPRFER ZFOPROFL ZFORDBUF ZFORDSVR ZFORTBYT ZFORTCOD ZFOSAVSV ZFOSAV01 ZFOSPCVC ZFOTWRMD ZFOWKBUF ZFOXER
             ZFOXSUFL

DMSXTB       DEFABS   DEFNBUN  DEF2P31  DELCOM   FILE     MSG      RESET    STACK    SUBR     TYPCHAIN TYPCHDEL TYPE     TYPLIGNE
             TYPNUM   TYPTRGT  TYPTRGTC

DMSXUP       ADTM     AFINIS   AFVS     ARDBUF   AWRBUF   CURRDATE CURRTIME DMSXFCIN DMSXFCNX DMSXFCPL DMSXFCSU DMSXFCUP DMSXFDSR
             DMSXIOWR DMSXSUCC DMSXSUCK DMSXSUCN FSTAIC   FSTD     FSTEPL   FSTFMODE FSTLRECL FSTRECFM FVSECT   FVSFSTAD MAXBUFLG
             NUCON    R0       R1       R10      R11      R12      R13      R14      R15      R2       R3       R4       R5
             R6       R7       R8       R9       SAVBWPTR SAVBYTE1 SAVBYTE2 SAVBYTE3 SAVBYTE4 SAVDWRD1 SAVDWRD2 SAVEREG  SAVLSAVB
```

MODULE	EXTERNAL REFERENCES (LABELS AND MODULES)												
	SAVREGO	SAVREG15	SAVWORD1	SAVWORD2	TYPE	ZDEACURD	ZDEACURL	ZDECTLON	ZDECURLN	ZDEDELPT	ZDEENDRG	ZDEFLAG1	ZDEFLAG2
	ZDEFLAG3	7DEFLEOF	ZDEFLSIZ	ZDEFMODE	ZDEFNAME	ZDEFTYPE	ZDELRECL	ZDEMRGUP	ZDERECFM	ZDESC	ZDESERCH	ZDESERIN	ZDESERLG
	ZDESIDCD	ZDESIDON	ZDESQ8ON	ZDESRPNG	ZDETOPPT	ZDETRUNC	ZDEUPINC	ZFOABUFF	ZFOAITNO	ZFOANOIT	ZFOFLAG	ZFOFMODE	ZFOFNAME
	ZFOFREPT	ZFOFTYPE	ZFOIADDR	ZFOLBUFF	ZFOLBWPT	ZFOLDSCR	ZFOLFLAG	ZFOLFWPT	ZFOLGOP1	ZFOLGOP2	ZFOLGOP3	ZFOLGOP4	ZFOLGOP5
	ZFOLNCHG	ZFOLNDEL	ZFCLNNEW	ZFOLRBUF	ZFOLSTSV	ZFONC	ZFOOPNB1	ZFOOPNB3	ZFOOPNB5	ZFOOPNB8	ZFOOPST1	ZFOOPST2	ZFOOPST3
	ZFOOPST4	ZFOOPST5	ZFOOPST6	ZFOPLIST	ZFORDBUF	ZFORECFM	ZFORTBYT	ZFOWKBUF	ZFOXER	ZFOXSUFL			
DMSZAP	CLOSELIB	DOSFLAGS	DOSSVC	FILE	FLAGS	FSCBBUFF	FSCBD	FSCBFM	FSCBFN	FSCBFT	FSCBFV	LOC	NUCON
	RESET	R0	R1	R10	R11	R12	R13	R14	R15	R2	R3	R4	R5
	R6	R7	R8	R9	TYPE								
DMSZER	DMSABN	DMSALU	DMSCIO	DMSCPF	DMSDBD	DMSDBG	DMSFET	DMSITE	DMSITP	DMSLIO	DMSPIO	DMSPNT	DMSTIO
	DMSTLA	DMSTQQ	DMSVIB	DMSVSR	DMSZEX								
DMSZES	ASSTAT	ASSTATZ	AYSTATZ	DMSZER	NUCON	R0	R1	R12	R14	R15	R2	R3	R4
	R5	R6	R7	R8	R9	VCADTLKP	VMSIZE						

Label-to-Module Cross Reference

LABEL	COUNT	REFERENCES											
AABBREV	000003	DMSCPF	DMSHLI	DMSITP									
AABNGO	000006	DMSFRE	DMSINS	DMSITI	DMSITS								
AABNSVC	000002	DMSINS	DMSSAB										
AACTFREE	000005	DMSBRD	DMSBWR	DMSERD	DMSPNT								
AACTFRET	000007	DMSBWR	DMSERD	DMSERS	DMSFNS								
AACTLKP	000022	DMSBRD	DMSBWR	DMSCPY	DMSDSK	DMSERD	DMSERS	DMSFNS	DMSINT	DMSLBM	DMSPNT	DMSRNM	DMSSOP
		DMSSTT	DMSTPE	DMSTPF	DMSTPG	DMSXCP							
AACTNXT	000001	DMSERS											
AADTLKP	000016	DMSDLK	DMSDOS	DMSHLS	DMSLBM	DMSLBT	DMSMVE	DMSPRE	DMSTMA	DMSTPE	DMSTPF	DMSTPG	DMSXFD
		DMSXSE											
AADTLKW	000013	DMSARX	DMSASM	DMSCPY	DMSDLK	DMSIFC	DMSLBM	DMSLBT	DMSLKD				
AADTNXT	000001	DMSCVH											
AAMSSYS	000004	DMSAMS	DMSDOS	DMSVSR									
ABAMSYS	000005	DMSBOP	DMSCLS	DMSCVH	DMSDOS	DMSSET							
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	DMSBOP	DMSDAS	DMSDOS	DMSFET	DMSINS	DMSOPT	DMSQRY	DMSSET	DMSSMN
		DMSSTG	DMSVSR										
ABLKIND	000003	DMSACM	DMSFOR	DMSINI									
ABNBIT	000004	DMSABN	DMSBTP	DMSDOS									
ABNCMSG	000007	DMSABN											
ABNERLST	000010	DMSABN	DMSITP										
ABNPAS13	000001	DMSABN											
ABNPSW	000032	DMSABN	DMSDBG	DMSFRE	DMSITI	DMSITP	DMSITS						
ABNREGS	000014	DMSABN	DMSDBG	DMSFRE	DMSITI	DMSITP	DMSITS						
ABNRR	000002	DMSABN											
ABWSECT	000009	DMSABN	DMSDBG	DMSFRE	DMSITI	DMSITP	DMSITS						
ACALL	000004	DMSFRE											
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										

LABEL	COUNT	REFERENCES
ACBMACRF	000001	DMSVIP
ACBMACR1	000002	DMSBOP
ACBOCEXT	000001	DMSVIP
ACBOCTER	000001	DMSVIP
ACBOEMPT	000001	DMSVIP
ACBOFLGS	000003	DMSBOP DMSVIP
ACBOKBUF	000001	DMSVIP
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
ACFILE10	000004	DMSDSK DMSTPE DMSTPF DMSTPG
ACFILE20	000004	DMSDSK DMSTPE DMSTPF DMSTPG
ACITDB	000001	DMSABN
ACLOSE	000004	DMSDOS DMSVSR
ACMSCVT	000007	DMSINS DMSLOS DMSSOP DMSSTG DMSVIB DMSVSR
ACMSRET	000004	DMSDOS DMSLDR DMSVIP
ACMSSEG	000017	DMSEDX DMSEXC DMSHLL DMSINS DMSITS DMSSAB DMSSET DMSTLA DMSXSG
ACMSZER	000002	DMSINS
ACTERS	000003	DMSTPE DMSTPF DMSTPG
ACTFLAG	000009	DMSTPE DMSTPF DMSTPG
ACTFM	000006	DMSTPE DMSTPF DMSTPG
ACTFN	000009	DMSTPE DMSTPF DMSTPG
ADBGSECT	000001	DMSITE
ADEVIND	000004	DMSACM DMSFOR DMSINI
ADEVSUP	000002	DMSACM DMSINI
ADEVSUP2	000001	DMSACM
ADEVTAB	000023	DMSAMS DMSASN DMSDBD DMSEDI DMSEDX DMSFOR DMSGIO DMSINI DMSSET DMSTIO DMSTMA DMSTPD
		DMSTPE DMSTPF DMSTPG
ADIKQLAB	000004	DMSDOS DMSVSR
ADIOSECT	000005	DMSACM DMSDIO DMSFNS DMSITI
ADMPEXEC	000002	DMSINS DMSSVT
ADMSABN	000001	DMSINS
ADMSABW	000009	DMSABN DMSITP DMSITS
ADMSALU	000010	DMSACC DMSACF DMSARE DMSFOR DMSINS
ADMSBLKR	000009	DMSDSK DMSTPE DMSTPF DMSTPG DMSXMA
ADMSBLKW	000008	DMSDSK DMSTPE DMSTPF DMSTPG
ADMSCAT	000001	DMSABN
ADMSCPF	000002	DMSINS DMSINT
ADMSCRD	000003	DMSABN DMSBTP DMSDBG
ADMSCWR	000001	DMSITE

LABEL	COUNT	REFERENCES
ADMSCWT	000001	DMSABN
ADMSERL	000093	DMSAMS DMSBOP DMSBWR DMSCIO DMSCLS DMSCVH DMSDAS DMSDBG DMSDOS DMSEDI DMSERD DMSERS
		DMSEXE DMSFCH DMSFET DMSFNS DMSFRE DMSITS DMSLBR DMSMOD DMSPIO DMSPRT DMSPUN DMSSEB
		DMSSET DMSSPR DMSSTT DMSSVT DMSTLB DMSVIB DMSVLT DMSXCP
ADMSERR	000031	DMSABN DMSFET DMSITP DMSLIO
ADMSFREB	000279	DMSABN DMSACC DMSACF DMSACM DMSALU DMSAMS DMSARE DMSAUD DMSBOP DMSBRD DMSBWR DMSCAT
		DMSCIT DMSCLS DMSCMP DMSCRD DMSCWR DMSDIO DMSDLB DMSDMP DMSDOS DMSEDX DMSERD DMSERS
		DMSEXC DMSEXE DMSEXI 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 DMSSFF DMSSLN DMSSOP DMSSTG DMSSVN DMSSVT DMSSVU DMSTLB
		DMSTRK DMSVSR DMSXCP
ADMSFRES	000002	DMSABN
ADMSPRT	000002	DMSSET
ADMSHLB	000003	DMSHLB DMSHLP
ADMSHLS	000006	DMSHLS
ADMSIOW	000006	DMSCIO DMSPIO
ADMSITI	000001	DMSITE
ADMSITSR	000001	DMSABN
ADMSLADN	000004	DMSACC DMSFOR
ADMSLIO	000042	DMSINS DMSLDR DMSOLD
ADMSOVS	000008	DMSOVR
ADMSPIOC	000002	DMSINS DMSPRT
ADMSROS	000016	DMSACM DMSALU DMSLDS DMSLFS DMSSCT DMSSEB DMSSVT
ADMSTRKA	000008	DMSACC DMSAUD DMSERD DMSERS
ADMSTRKD	000008	DMSAUD DMSERD DMSERS DMSFOR
ADMSTRKM	000006	DMSACC
ADMSVIB	000002	DMSINS DMSVSR
ADOSDCSS	000002	DMSITS DMSSET
ADTADD	000018	DMSACF DMSACM DMSAUD DMSDIO DMSERD DMSERS DMSFNS DMSMOD
ADTADDED	000005	DMSABN DMSACC DMSALU DMSARE DMSLAD
ADTADD2	000002	DMSERD
ADTAMHD	000003	DMSAUD DMSTRK
ADTAMHO	000006	DMSACC DMSTRK
ADTAMP1	000022	DMSACC DMSACM DMSALU DMSAUD DMSFOR
ADTAMP2	000014	DMSACC DMSACM DMSALU DMSAUD DMSFOR
ADTAMP3	000006	DMSACC DMSALU
ADTANACW	000010	DMSERD DMSERS DMSFNS
ADTARES	000028	DMSABN DMSACC DMSACF DMSACM DMSAUD DMSERD DMSERS DMSFOR DMSLBT DMSLFS DMSTRK
ADTBWPTR	000007	DMSACC DMSARE DMSLAD
ADTCFST	000014	DMSACF DMSCPY DMSERS DMSLFS
ADTCHBA	000028	DMSACF DMSCPY DMSDSK DMSERD DMSERS DMSLFS DMSPNT DMSRNM DMSSTT DMSTPE DMSTPF DMSTPG
ADTCHMAP	000005	DMSAUD DMSTRK
ADTCYL	000015	DMSACM DMSAUD DMSFOR DMSINI DMSLDS DMSQRY DMSROS
ADTDAMAP	000008	DMSALU DMSAUD DMSTRK
ADTDBSIZ	000127	DMSACC DMSACF DMSACM DMSALU DMSAUD DMSCPY DMSDSK DMSERD DMSERS DMSFNS DMSFOR DMSINI
		DMSLAD DMSLBT DMSLFS DMSQRY DMSTPE DMSTPF DMSTPG DMSXMA

LABEL	COUNT	REFERENCES
ADTDCRED	000001	DMSFOR
ADTDFP1	000014	DMSACF DMSALU DMSAUD DMSFOR DMSLFS
ADTDFP2	000018	DMSACF DMSALU DMSAUD DMSLFS
ADTDFP3	000016	DMSACF DMSALU DMSAUD DMSLFS
ADTDIOA	000003	DMSACM DMSFOR
ADTDIOB	000003	DMSACM DMSFOR
ADTDOP	000004	DMSACC DMSACM DMSAUD
ADTDTA	000039	DMSACC DMSACM DMSARE DMSASN DMSAUD DMSBWR DMSDAS DMSDIO DMSDOS DMSERD DMSFNS DMSFOR
		DMSLAD DMSMVE DMSPRE DMSQRY DMSROS DMSSET DMSTQQ DMSXCP
ADTEDF	000124	DMSABN DMSACC DMSACF DMSACM DMSALU DMSARE DMSAUD DMSBRD DMSBWR DMSCPY DMSDSK DMSERS
		DMSFNS DMSFOR DMSHLS DMSINT DMSLAD DMSLBM DMSLBT DMSLFS DMSLST DMSMOD DMSPNT DMSQRY
		DMSSOP DMSSTT DMSSVT DMSTMA DMSTPE DMSTPF DMSTPG DMSUPD DMSXFD DMSXMA
ADTEDFAE	000005	DMSACC DMSARE DMSFOR DMSTRK
ADTFALNM	000003	DMSACF
ADTFALTY	000004	DMSACF
ADTFALUF	000005	DMSACC DMSACF DMSFOR
ADTFBABF	000004	DMSACM DMSFOR
ADTFBALB	000002	DMSACM DMSFOR
ADTFDA	000056	DMSABN DMSACC DMSACF DMSACM DMSALU DMSAUD DMSERS DMSFOR DMSINS DMSLAD DMSLFS DMSLST
ADTFDOS	000020	DMSACC DMSASN DMSBOP DMSCVH DMSDLB DMSEXT DMSFOR DMSMVE DMSQRY DMSROS DMSSET DMSXCP
ADTFFSTF	000005	DMSACC DMSACF DMSFOR DMSINS
ADTFFSTV	000009	DMSABN DMSINS DMSLAD DMSLFS
ADTFLG1	000123	DMSABN DMSACC DMSACF DMSACM DMSALU DMSARE DMSARN DMSARX DMSASM DMSASN DMSBOP DMSBWR
		DMSCPY DMSCVH DMSDAS DMSDIO DMSDLK DMSDOS DMSDSL DMSERD DMSERS DMSEXT DMSFNS DMSFOR
		DMSHLS DMSINS DMSLAD DMSLAF DMSLBM DMSLBT DMSLDS DMSLFS DMSLLU DMSLST DMSMVE DMSQRY
		DMSRNM DMSROS DMSSOP DMSSTT DMSSVT DMSSVU DMSTQQ DMSTRK DMSUPD DMSXSE
ADTFLG2	000073	DMSABN DMSACC DMSACF DMSACM DMSALU DMSARE DMSASN DMSBOP DMSCVH DMSDAS DMSDLB DMSEXT
		DMSFOR DMSLAD DMSLDS DMSLFS DMSLST DMSMVE DMSQRY DMSROS DMSSET DMSSTT DMSTQQ DMSTRK
		DMSXCP DMSXIN
ADTFLG3	000035	DMSACC DMSACF DMSACM DMSALU DMSARE DMSAUD DMSBOP DMSBWR DMSFNS DMSINS DMSLFS DMSLLU
		DMSQRY DMSROS DMSXCP
ADTFLG4	000133	DMSABN DMSACC DMSACF DMSACM DMSALU DMSARE DMSAUD DMSBRD DMSBWR DMSCPY DMSDSK DMSERS
		DMSFNS DMSFOR DMSHLS DMSINT DMSLAD DMSLBM DMSLBT DMSLFS DMSLST DMSMOD DMSPNT DMSQRY
		DMSSOP DMSSTT DMSSVT DMSTMA DMSTPE DMSTPF DMSTPG DMSTRK DMSUPD DMSXFD DMSXMA
ADTFMDRO	000003	DMSACF
ADTFMFD	000005	DMSACM DMSBOP DMSTQQ DMSTRK
ADTFMIN	000003	DMSACC DMSALU
ADTFNOAB	000004	DMSARE DMSAUD
ADTFORCE	000007	DMSACC DMSACF DMSACM DMSINS DMSROS
ADTFQQF	000005	DMSABN DMSACM DMSALU DMSFOR
ADTFRO	000043	DMSABN DMSACC DMSACF DMSACM DMSALU DMSARE DMSASN DMSBOP DMSCVH DMSDAS DMSDIO DMSDOS
		DMSERS DMSEXT DMSFNS DMSFOR DMSLAD DMSLBM DMSLBT DMSLDS DMSLFS DMSLST DMSMVE DMSQRY
		DMSRNM DMSSOP DMSSTT DMSUPD
ADTFROS	000038	DMSABN DMSACC DMSACF DMSALU DMSARE DMSASN DMSBOP DMSCVH DMSDAS DMSDLB DMSEXT DMSFOR
		DMSLAD DMSLDS DMSLFS DMSLST DMSQRY DMSROS DMSSTT DMSXCP DMSXIN
ADTFRW	000088	DMSACC DMSACF DMSACM DMSALU DMSARE DMSARN DMSARX DMSASM DMSASN DMSBOP DMSBWR DMSCPY

LABEL	COUNT	REFERENCES											
		DMSCVH	DMSDAS	DMSDIO	DMSDLK	DMSDOS	DMSDSL	DMSERD	DMSERS	DMSEXT	DMSFOR	DMSHLS	DMSLAD
		DMSLAF	DMSLBM	DMSLBT	DMSLDS	DMSLFS	DMSLLU	DMSLST	DMSMVE	DMSQRY	DMSRNM	DMSSTT	DMSSVT
		DMSSVU	DMSTQQ	DMSTRK	DMSUPD	DMSXCP	DMSXSE						
ADTFRWOS	000005	DMSLLU	DMSQRY	DMSROS									
ADTFSORT	000003	DMSACF	DMSINS	DMSLFS									
ADTFSTC	000029	DMSACC	DMSACF	DMSALU	DMSARE	DMSAUD	DMSBWR	DMSERD	DMSERS	DMSFNS	DMSFOR	DMSINS	DMSQRY
ADTFSTSZ	000043	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD	DMSCPY	DMSDSK	DMSERS	DMSFOR	DMSGND	DMSINS	DMSLAD
		DMSLFS	DMSLST	DMSSTT									
ADTFTYP	000011	DMSACF	DMSALU	DMSDSK	DMSFNS	DMSLFS	DMSRNM	DMSTPE	DMSTPF	DMSTPG			
ADTFUPD1	000006	DMSAUD	DMSFNS										
ADTFVS	000001	DMSLAD											
ADTFXCHN	000005	DMSBWR	DMSFNS										
ADTHBCT	000022	DMSABN	DMSACC	DMSACF	DMSACM	DMSAUD	DMSERS	DMSFOR	DMSLAD	DMSLFS			
ADTID	000016	DMSDOS	DMSDSK	DMSFOR	DMSLAB	DMSLDS	DMSLST	DMSPRE	DMSPUN	DMSQRY	DMSUPD		
ADTIDENT	000018	DMSACM	DMSALU	DMSFOR	DMSINI	DMSINS	DMSLDS	DMSROS					
ADTLABSZ	000005	DMSALU	DMSFOR										
ADTLAST	000013	DMSACC	DMSACM	DMSAUD	DMSFOR	DMSTRK							
ADTLB	000001	DMSLAD											
ADTLD	000004	DMSABN	DMSACC	DMSARE	DMSLAD								
ADTLEFT	000032	DMSACC	DMSACM	DMSAUD	DMSERD	DMSFOR	DMSLAD	DMSQRY	DMSROS	DMSTRK			
ADTLFST	000011	DMSACC	DMSACF	DMSERS	DMSFOR	DMSLFS							
ADTLHBA	000013	DMSACC	DMSACF	DMSAUD	DMSERS	DMSFNS	DMSFOR	DMSLFS					
ADTM	000131	DMSACC	DMSACF	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSBWR	DMSCMP	DMSCPY	DMSCVH
		DMSDAS	DMSDLK	DMSDSK	DMSDSL	DMSEDX	DMSERD	DMSERS	DMSEXC	DMSEXT	DMSFOR	DMSHLS	DMSIFC
		DMSLAD	DMSLAF	DMSLBM	DMSLDS	DMSLFS	DMSLKD	DMSLST	DMSPRE	DMSQRY	DMSRNM	DMSROS	DMSSET
		DMSSOP	DMSSTT	DMSTPE	DMSTPF	DMSTPG	DMSUPD	DMSXCP	DMSXIN	DMSXUP			
ADTMCYL	000007	DMSFOR	DMSINI										
ADTMFDA	000011	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD						
ADTMFDN	000014	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD						
ADTMSK	000034	DMSACC	DMSACM	DMSALU	DMSAUD	DMSFOR	DMSTRK						
ADTMX	000033	DMSACC	DMSACM	DMSALU	DMSARN	DMSARX	DMSASM	DMSBWR	DMSLAD	DMSLAF	DMSLFS	DMSQRY	DMSSTT
		DMSUPD											
ADTMXBML	000001	DMSACM											
ADTNACW	000008	DMSBWR	DMSFNS	DMSSOP									
ADTNFST	000012	DMSACF	DMSACM	DMSALU	DMSFOR	DMSLAD							
ADTNUM	000028	DMSACC	DMSACM	DMSAUD	DMSFOR	DMSQRY	DMSTRK						
ADTPQM1	000010	DMSACM	DMSALU	DMSAUD	DMSFOR								
ADTPQM2	000010	DMSACC	DMSACF	DMSACM	DMSAUD	DMSFOR							
ADTPQM3	000006	DMSABN	DMSACC	DMSACM	DMSALU	DMSFOR							
ADTPSTM	000007	DMSLAD	DMSLFS										
ADTPTR	000014	DMSACC	DMSARE	DMSCVH	DMSDAS	DMSDOS	DMSLAD						
ADTQQM	000009	DMSACC	DMSACM	DMSALU	DMSAUD	DMSFOR	DMSTQQ						
ADTRANS	000014	DMSSLN											
ADTRES	000013	DMSACC	DMSACF	DMSALU	DMSBWR	DMSFNS	DMSFOR	DMSLAD	DMSTRK				
ADTROX	000003	DMSACM	DMSALU										
ADTSECT	000208	DMSACC	DMSACF	DMSACM	DMSAMS	DMSARE	DMSASN	DMSAUD	DMSBRD	DMSCMP	DMSCVH	DMSDIO	DMSDLB

LABEL	COUNT	REFERENCES											
		DMSDLK	DMSDOS	DMSDSL	DMSERS	DMSEXC	DMSEXT	DMSFNS	DMSFOR	DMSGND	DMSIFC	DMSINT	DMSLAF
		DMSLBM	DMSLBT	DMSLDS	DMSLLU	DMSMOD	DMSMVE	DMSPNT	DMSQRY	DMSROS	DMSSET	DMSSOP	DMSSTT
		DMSSVU	DMSTMA	DMSTPF	DMSTPG	DMSTRK	DMSUPD	DMSXFD					
ADTUSED	000027	DMSACC	DMSACM	DMSAUD	DMSFOR	DMSQRY	DMSTRK						
ADTXNREC	000005	DMSFNS											
ADT1ST	000010	DMSACC	DMSAUD	DMSFOR	DMSTRK								
AEDLIN	000001	DMSEDX											
AEPOINT	000005	DMSDSK	DMSEXT	DMSINS	DMSINT	DMSLIB							
AERASE	000054	DMSAMS	DMSBOP	DMSDLK	DMSDSK	DMSDSL	DMSEDI	DMSFNS	DMSLIO	DMSLLU	DMSLST	DMSMOD	DMSOLD
		DMSPRV	DMSRDC	DMSRNE	DMSRRV	DMSSOP	DMSSRV	DMSSVT	DMSSVU	DMSTPE	DMSTPF	DMSTPG	DMSUPD
		DMSXFD	DMSXPT	DMSXRE									
AERR	000001	DMSITS											
AESTATE	000046	DMSAMS	DMSBOP	DMSDLK	DMSDSK	DMSDSL	DMSEDI	DMSEDX	DMSEXE	DMSEXT	DMSFCH	DMSFLD	DMSGLB
		DMSGND	DMSINS	DMSLDR	DMSLIB	DMSMOD	DMSOLD	DMSOPL	DMSPRT	DMSPUN	DMSRRV	DMSSET	DMSSLN
		DMSSOP	DMSSRV	DMSSVU	DMSSYN	DMSTPE	DMSTPF	DMSTPG	DMSTYP	DMSUPD	DMSXMA	DMSXSU	
AESTATEW	000006	DMSAMS	DMSEDX	DMSMOD	DMSRDC	DMSRNM							
AEXCAB	000001	DMSABN											
AEXEC	000001	DMSEXC											
AEXTEND	000007	DMSEDI	DMSEDX	DMSUPD									
AEXTSECT	000015	DMSINS	DMSINT	DMSIOW	DMSITE	DMSQRY	DMSSET	DMSSTG	DMSSVN	DMSSVT			
AFINIS	000100	DMSABN	DMSACC	DMSARE	DMSCMP	DMSDLK	DMSDSK	DMSEDI	DMSEDX	DMSEXC	DMSEXE	DMSEXT	DMSFOR
		DMSGLB	DMSINT	DMSLDR	DMSLIB	DMSLIO	DMSLLU	DMSMOD	DMSOLD	DMSPRT	DMSPRV	DMSPUN	DMSRDC
		DMSRNE	DMSRRV	DMSSLN	DMSSOP	DMSSRV	DMSSTG	DMSSYN	DMSTPE	DMSTPF	DMSTPG	DMSTYP	DMSUPD
		DMSVLT	DMSXCP	DMSXFD	DMSXGT	DMSXIN	DMSXMA	DMSXPT	DMSXRE	DMSXUP			
AFLAGLOC	000001	DMSEDX											
AFREETAB	000006	DMSFRE	DMSSET										
AFST	000001	DMSSYN											
AFSTFNRD	000004	DMSEDI	DMSEDX										
AFSTLKP	000005	DMSCPY	DMSEXC										
AFSTLKW	000001	DMSCPY											
AFSTPLST	000012	DMSTPE	DMSTPF	DMSTPG									
AFTADT	000048	DMSBRD	DMSBWR	DMSCPY	DMSERD	DMSERS	DMSFNS	DMSINT	DMSLAF	DMSPNT	DMSRNM	DMSSOP	DMSSTT
AFTARP	000012	DMSDSK	DMSERD	DMSFNS	DMSINT	DMSPNT	DMSSOP	DMSTPE	DMSTPF	DMSTPG			
AFTAWP	000027	DMSDSK	DMSERD	DMSFNS	DMSINT	DMSLBM	DMSPNT	DMSSOP	DMSTPE	DMSTPF	DMSTPG		
AFTBFORM	000004	DMSERD											
AFTBLKWD	000006	DMSERD											
AFTBPRCT	000004	DMSERD											
AFTCLA	000012	DMSBRD	DMSBWR	DMSFNS									
AFTCLB	000012	DMSBRD	DMSBWR	DMSERD	DMSFNS								
AFTCLD	000015	DMSBRD	DMSBWR	DMSFNS									
AFTCLDX	000005	DMSBWR	DMSFNS										
AFTCLN	000014	DMSBRD	DMSBWR	DMSFNS									
AFTCLX	000006	DMSBWR	DMSFNS										
AFTDBA	000024	DMSBRD	DMSBWR	DMSERD	DMSFNS								
AFTDBD	000010	DMSBRD	DMSBWR	DMSFNS									
AFTDBF	000003	DMSBWR											

LABEL	COUNT	REFERENCES							
AFTDBN	000010	DMSERD	DMSBWR						
AFTEBDSP	000003	DMSERD							
AFTEBLIN	000002	DMSERD							
AFTERR8	000005	DMSERD							
AFTFBA	000005	DMSBRD	DMSBWR	DMSFNS					
AFTFCLA	000008	DMSBRD	DMSBWR	DMSFNS					
AFTFCLX	000008	DMSBWR	DMSFNS						
AFTFLG	000053	DMSBRD	DMSBWR	DMSERD	DMSERS	DMSFNS	DMSLAF	DMSSOP	DMSSTT
AFTFLG2	000068	DMSBWR	DMSERD	DMSFNS					
AFTFSF	000002	DMSLAF							
AFTFST	000012	DMSBRD	DMSBWR	DMSERS	DMSFNS	DMSLAF	DMSSOP	DMSSTT	
AFTFULD	000002	DMSBWR	DMSFNS						
AFTID	000010	DMSBRD	DMSBWR						
AFTIN	000014	DMSBRD	DMSBWR	DMSSOP					
AFTLD	000002	DMSLAF							
AFTLSTRC	000007	DMSERD							
AFTMXBLK	000007	DMSERD							
AFTNEW	000010	DMSBWR	DMSERD	DMSFNS					
AFTOCLDX	000003	DMSBWR							
AFTOLDCL	000006	DMSBWR							
AFTOVLAP	000011	DMSERD							
AFTPFST	000011	DMSBRD	DMSERD	DMSERS	DMSFNS	DMSLAF	DMSSOP		
AFTPHYP	000007	DMSERD	DMSERS	DMSFNS	DMSPNT				
AFTPTR	000012	DMSLAF							
AFTRD	000012	DMSBRD	DMSBWR	DMSERD	DMSFNS	DMSSTT			
AFTRDBLK	000011	DMSERD	DMSERS	DMSFNS					
AFTRDID	000005	DMSERD							
AFTREAD	000015	DMSERD							
AFTRLST	000003	DMSTPE	DMSTPF	DMSTPG					
AFTSTART	000001	DMSLAF							
AFTSVBLK	000006	DMSERD							
AFTSVFP1	000002	DMSERD							
AFTSVFP2	000002	DMSERD							
AFTSVFP3	000002	DMSERD							
AFTSVFP4	000002	DMSERD							
AFTUBFAD	000010	DMSERD							
AFTUBFLG	000025	DMSERD							
AFTUFP1	000010	DMSERD	DMSERS	DMSFNS					
AFTUFP2	000006	DMSERD	DMSERS	DMSFNS					
AFTUFP3	000006	DMSERD	DMSERS	DMSFNS					
AFTUFP4	000007	DMSERD	DMSERS	DMSFNS					
AFTUFP5	000008	DMSERD	DMSERS	DMSFNS					
AFTUSED	000004	DMSFNS	DMSLAF						
AFTVLGTH	000006	DMSERD							
AFTVLREC	000005	DMSERD							
AFTWRT	000018	DMSBRD	DMSBWR	DMSERD	DMSERS	DMSFNS	DMSSTT		

LABEL	COUNT	REFERENCES
AFVS	000084	DMSABN DMSACC DMSACF DMSACM DMSALU DMSARN DMSARX DMSASM DMSAUD DMSBRD DMSBTB DMSBTP
		DMSBWR DMSCIT DMSCMP DMSCRD DMSCWR DMSCWT DMSDIO DMSDOS DMSDSK DMSDSL DMSERD DMSERS
		DMSEXC DMSFNS DMSFOR DMSGND DMSINT DMSITE DMSITI DMSITP DMSITS DMSLAD DMSLBM DMSLFS
		DMSMOD DMSPNT DMSPUN DMSQRY DMSRNM DMSSLN DMSSOP DMSSTT DMSTPE DMSTPF DMSTPG DMSUPD
		DMSXCP DMSXIN DMSXMA DMSXUP
AGCLOSE	000001	DMSHLI
AGETCLK	000001	DMSEXT
AGOPEN	000001	DMSHLI
AGREAD	000001	DMSHLP
AINCORE	000006	DMSEDI DMSRNE DMSXCT
AINTAB	000001	DMSABN
AINTRTBL	000011	DMSABN DMSCRD DMSQRY DMSSET DMSXPO DMSXSC DMSXSD
AIOBUFF	000018	DMSTPE DMSTPF DMSTPG
AIOSECT	000008	DMSABN DMSCIT DMSDBG DMSHDI DMSINT DMSITI
AKILLEX	000015	DMSACC DMSAUD DMSBWR DMSDBG DMSDIO DMSDSK DMSERD DMSERS DMSFNS DMSINS DMSRNM DMSTPE
		DMSTPF DMSTPG
ALABELRD	000003	DMSACM DMSFOR
ALABELWR	000004	DMSAUD DMSFOR DMSTRK
ALADAD	000001	DMSABN
ALCHAR1	000002	DMSEDI
ALCHAR2	000002	DMSEDI
ALDRTBLS	000029	DMSBTB DMSFET DMSGND DMSINS DMSLDR DMSLOA DMSMDP DMSMOD DMSOLD DMSQRY DMSSET DMSSLN
		DMSSTG
ALIASENT	000004	DMSLIO DMSSLN
ALINELOC	000001	DMSEDX
ALINKINT	000002	DMSHLI
ALINKPUT	000002	DMSHLP
ALL	000010	DMSEXE DMSFNS DMSHLI DMSINS DMSLLU DMSROS DMSSAB DMSSET
ALOKTABA	000009	DMSABN DMSLCK
ALOKTABE	000003	DMSABN DMSLCK
ALOKTABS	000004	DMSABN
ALOKTB	000009	DMSABN DMSDOS DMSLCK DMSSET DMSVSR
ALTASAVE	000008	DMSAMS DMSDOS DMSITP DMSSET
ALTFILN1	000003	DMSHLI
ALTFILN2	000002	DMSHLI
ALTFILT1	000001	DMSHLI
ALTLIST	000008	DMSEDI
ALTMODE	000008	DMSEDX
AMAIN	000002	DMSHLE DMSHLI
AMERGE	000002	DMSHLP
ANCHENDA	000003	DMSDOS DMSSTG
ANCHENTP	000001	DMSDOS
ANCHINST	000001	DMSDOS
ANCHLDPT	000002	DMSDOS
ANCHLENG	000002	DMSDOS
ANCHPHLN	000001	DMSDOS

LABEL	COUNT	REFERENCES											
ANCHPHNM	000005	DMSDOS											
ANCHSECT	000003	DMSDOS	DMSSTG										
ANCHSIZ	000005	DMSFCH	DMSSTG										
ANCHSTSW	000001	DMSDOS											
ANUCEND	000004	DMSDIO	DMSHDI	DMSHDS	DMSINS								
ANUMLOC	000001	DMSEDX											
AOPSECT	000028	DMSABN	DMSARN	DMSCRD	DMSCWR	DMSCWT	DMSDBG	DMSEXC	DMSEXI	DMSEXT	DMSINS	DMSINT	DMSSBS
		DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVN	DMSSVT						
AOSMODL	000022	DMSINS	DMSITS	DMSLDR	DMSSAB	DMSSET							
AOSRET	000003	DMSDOS	DMSSOP	DMSVIP									
AOUTRTBL	000010	DMSABN	DMSCWR	DMSQRY	DMSSET	DMSXPO	DMSXSC	DMSXSD					
APARMRET	000001	DMSHLI											
APGMSECT	000007	DMSITP	DMSSAB	DMSSLN	DMSSTG	DMSSVT							
APIE	000001	DMSSVT											
APOINT	000002	DMSEXE	DMSINS										
APPSAVE	000005	DMSAMS	DMSDOS	DMSITP	DMSSET								
APRILB	000006	DMSLDR	DMSOLD										
APSTATE	000001	DMSHLI											
APSV	000035	DMSLDR	DMSLGT	DMSLIB	DMSLIO	DMSLSB	DMSOLD						
AQQTRK	000004	DMSBWR	DMSINS	DMSTQQ									
AQQTRKX	000007	DMSBWR	DMSERS	DMSFNS	DMSINS								
AQUICKEX	000001	DMSHLE											
ARDBUF	000075	DMSCMP	DMSDLK	DMSDSK	DMSEDI	DMSEDX	DMSEXE	DMSEXI	DMSEXT	DMSGLB	DMSITS	DMSLBT	DMSLDR
		DMSLGT	DMSMOD	DMSOLD	DMSPRT	DMSPUN	DMSRNE	DMSSLN	DMSSVT	DMSSVU	DMSSYN	DMSTPE	DMSTPF
		DMSTPG	DMSTYP	DMSUPD	DMSXCP	DMSXGT	DMSXIN	DMSXMA	DMSXRE	DMSXUP			
ARDTK	000021	DMSACF	DMSACM	DMSBRD	DMSBWR	DMSERD	DMSERS	DMSFNS	DMSMOD				
AREA	000066	DMSCMP	DMSEDI	DMSEXE	DMSINS	DMSPRT	DMSRRV	DMSSET	DMSSFF	DMSTYP	DMSXBG	DMSXCT	DMSXDC
		DMSXED	DMSXMA	DMSXPX									
ARFLG	000002	DMSDOS											
ARGMAX	000001	DMSDBG											
ARGS	000049	DMSDBD	DMSDBG	DMSEXE	DMSITE								
ARGSAV	000008	DMSDBG											
ARGSCT	000016	DMSDBG											
AROUND	000009	DMSCPY	DMSLBM	DMSLDR	DMSOLD	DMSTPE	DMSTPF	DMSTPG					
ASCANN	000010	DMSAMS	DMSBTP	DMSEXE	DMSLDR	DMSOLD	DMSRDC	DMSXCM	DMSXED	DMSXHL	DMSXIN		
ASCANO	000002	DMSEXT	DMSSRT										
ASCBPTR	000002	DMSINT											
ASERR	000004	DMSHLB	DMSHLI	DMSHLS									
ASORTFST	000004	DMSACF	DMSARE	DMSINS									
ASP	000001	DMSHLP											
ASSTAT	000006	DMSINS	DMSZES										
ASSTATX	000002	DMSINS											
ASSTATZ	000004	DMSINS	DMSZES										
ASTATE	000002	DMSFLD	DMSXCM										
ASTATEW	000001	DMSERS											
ASTATEXT	000003	DMSINS	DMSSTG										

LABEL	COUNT	REFERENCES											
ASTGSB	000002	DMSABN	DMSINT										
ASTRINIT	000002	DMSARN	DMSSRT										
ASUBFST	000003	DMSABN	DMSINT										
ASUBRET	000002	DMSINT											
ASUBSECT	000006	DMSABN	DMSINM	DMSINT									
ASUBSTAT	000003	DMSABN	DMSINT										
ASVCSECT	000032	DMSCIT	DMSFRE	DMSHDS	DMSINT	DMSITE	DMSITS	DMSLAD	DMSLFS	DMSOVR	DMSOVS	DMSSLN	DMSXMA
ASWRTIO	000002	DMSHLP											
ASWRTMSG	000003	DMSHLE											
ASWRTPRP	000001	DMSHLI											
ASYSCOM	000011	DMSBOP	DMSDOS	DMSETR	DMSFET	DMSSET	DMSSTG						
ASYSNAMS	000033	DMSAMS	DMSBOP	DMSBTP	DMSDOS	DMSEDX	DMSEXC	DMSHLL	DMSINS	DMSINT	DMSITS	DMSQRY	DMSSET
		DMSTLA	DMSVIB	DMSVSR	DMSXSG								
ASYSREF	000035	DMSASN	DMSBOP	DMSCLS	DMSCVH	DMSDLB	DMSDMP	DMSDOS	DMSETR	DMSFCH	DMSINS	DMSITP	DMSLLU
		DMSOPL	DMSPRV	DMSQRY	DMSRRV	DMSSET	DMSSRV	DMSVLT	DMSXCP				
ATABEND	000011	DMSAMS	DMSTIO	DMSTMA	DMSTPD	DMSTPE	DMSTPF	DMSTPG					
ATBLIND	000004	DMSACM	DMSFOR	DMSINI									
ATCBPTR	000013	DMSAMS	DMSBAB	DMSDOS	DMSITP	DMSSET	DMSVSR						
ATFINIS	000008	DMSBWR	DMSERS	DMSRNM	DMSSVT	DMSSVU							
ATLBMODL	000002	DMSINS	DMSTLA										
ATRBIP	000001	DMSXSC											
ATRHIGH	000010	DMSXPO	DMSXSC	DMSXSD	DMSXSE								
ATRINIB	000002	DMSXPO	DMSXSC										
ATRKLKP	000003	DMSAUD	DMSBWR	DMSTQQ									
ATRKLKPX	000012	DMSAUD	DMSBWR	DMSERS	DMSFNS	DMSTQQ							
ATRMDT	000006	DMSXPO	DMSXSC	DMSXSD									
ATRPRT	000011	DMSXPO	DMSXSC	DMSXSD	DMSXSE								
ATRRST	000003	DMSXPO	DMSXSD										
ATRUEEND	000004	DMSHLE	DMSHLI	DMSHLP	DMSHLS								
ATRUNC	000002	DMSERD	DMSLBM										
ATSOCPPL	000001	DMSSTG											
ATTCHFST	000002	DMSHLS											
ATTLAD	000001	DMSHLS											
ATTN	000020	DMSABN	DMSCIT	DMSEDI	DMSEXE	DMSFNC	DMSSVN						
ATTNHIT	000004	DMSCIT	DMSITI										
ATTNLEN	000007	DMSEDI											
ATTRELO	000001	DMSHLI											
ATYPSRCH	000008	DMSACF	DMSDSK	DMSFNS	DMSRNM	DMSTPE	DMSTPF	DMSTPG					
AUGSW	000016	DMSHLP	DMSHLS										
AUPDISK	000021	DMSARE	DMSBWR	DMSDSK	DMSERS	DMSFNS	DMSFOR	DMSRNM	DMSSOP	DMSSVT	DMSSVU	DMSTPE	DMSTPF
		DMSTPG											
AUPIE	000002	DMSITP											
AUSABRV	000004	DMSABN	DMSINA	DMSQRY	DMSSYN								
AUSERRST	000003	DMSERR											
AUSRAREA	000042	DMSABN	DMSBRD	DMSBTB	DMSDOS	DMSFCH	DMSFET	DMSFRE	DMSINS	DMSINT	DMSLDR	DMSLOA	DMSLSB
		DMSMOD	DMSOLD	DMSSET	DMSSLN	DMSSMN	DMSSTG	DMSSVT					

LABEL	COUNT	REFERENCES											
AUSRILST	000008	DMSABN	DMSHDI										
AUSRITBL	000007	DMSABN	DMSHDI										
AUTOCNT	000005	DMSEDI											
AUTOCURR	000003	DMSEDI											
AUTOREG	000002	DMSEDI											
AUXDIR	000001	DMSHLI											
AVIPWORK	000009	DMSVIB	DMSVIP	DMSVSR									
AVRADR	000002	DMSDOS											
AVRDEVC	000009	DMSDOS											
AVRFLAG	000004	DMSDOS											
AVRLNO	000001	DMSDOS											
AVRNLNO	000001	DMSDOS											
AVRPUB	000001	DMSDOS											
AVRPUT	000002	DMSHLP											
AVRTYPE	000003	DMSDOS											
AVRVCI	000001	DMSDOS											
AVRVNUM	000001	DMSDOS											
AVRVOLID	000002	DMSDOS											
AVRVTOC	000003	DMSDOS											
AVSAMSYS	000011	DMSABN	DMSBOP	DMSCLS	DMSDOS	DMSSET	DMSVIB	DMSVIP	DMSVSR				
AVSREOJ	000002	DMSDOS	DMSINS										
AVSRWORK	000005	DMSCLS	DMSVSR										
AWAIT	000001	DMSITS											
AWNGRET	000004	DMSHLE	DMSHLI										
AWRBUF	000044	DMSDLK	DMSDSK	DMSEDI	DMSITS	DMSLBT	DMSLIO	DMSLLU	DMSMOD	DMSOLD	DMSPRV	DMSRDC	DMSRNE
		DMSRRV	DMSSRV	DMSSVT	DMSSVU	DMSTPE	DMSTPF	DMSTPG	DMSUPD	DMSXCP	DMSXFD	DMSXPT	DMSXRE
		DMSXUP											
AWRTK	000011	DMSACC	DMSAUD	DMSBWR	DMSERD	DMSERS	DMSFNS						
AYSTATX	000002	DMSINS											
AYSTATZ	000004	DMSINS	DMSZES										
BALR	000344	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSAUD	DMSBOP	DMSBRD	DMSBWR	DMSCAT
		DMSCIT	DMSCLS	DMSCMP	DMSCRD	DMSCWR	DMSDIO	DMSDLB	DMSDLK	DMSDMP	DMSDOS	DMSEDX	DMSERD
		DMSERS	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFET	DMSFNS	DMSFOR	DMSFRE	DMSHDI	DMSHDS
		DMSINS	DMSINT	DMSITE	DMSITP	DMSITS	DMSLAD	DMSLAF	DMSLDR	DMSLFS	DMSLGT	DMSLIB	DMSLSB
		DMSMOD	DMSOLD	DMSOPL	DMSOR1	DMSROS	DMSSAB	DMSSET	DMSSFF	DMSSLN	DMSSOP	DMSSTG	DMSSTT
		DMSSVN	DMSSVT	DMSSVU	DMSTLB	DMSTRK	DMSUPD	DMSVSR	DMSXCP				
BALRSAVE	000041	DMSAUD	DMSCPF	DMSDBG	DMSFNS	DMSINA	DMSINM	DMSSCN	DMSSMN	DMSSTG	DMSVIB		
BAMFLAGS	000007	DMSBOP	DMSCLS	DMSDOS	DMSSET								
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											

LABEL	COUNT	REFERENCES											
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											
BBOXADR	000001	DMSSTG											
BEGAT	000003	DMSDBG											
BGCOM	000060	DMSAMS	DMSASN	DMSBOP	DMSCLS	DMSCVH	DMSDAS	DMSDLB	DMSDLK	DMSDMP	DMSDOS	DMSDSV	DMSETR
		DMSFCH	DMSFET	DMSINS	DMSITP	DMSLLU	DMSOPL	DMSOPT	DMSPRV	DMSQRY	DMSRRV	DMSSET	DMSSMN
		DMSSRV	DMSSTG	DMSVLT	DMSVSR	DMSXCP							
BIPWCCMD	000001	DMSXSC											
BITS	000012	DMSDBG	DMSPRT	DMSPUN									
BLANK	000162	DMSAMS	DMSARN	DMSARX	DMSASM	DMSASN	DMSBOP	DMSBTP	DMSCIT	DMSCLS	DMSCMP	DMSCVH	DMSCWR
		DMSDLB	DMSDLK	DMSDSK	DMSDSL	DMSDSV	DMSEDX	DMSEXE	DMSEXT	DMSFET	DMSFLD	DMSINS	DMSLDS
		DMSLLU	DMSPIO	DMSPRE	DMSPRV	DMSQRY	DMSRNE	DMSRRV	DMSSCR	DMSSET	DMSSRV	DMSTPE	DMSTPF
		DMSTPG	DMSTRK	DMSUTL									
BLANK1	000001	DMSEDX											
BLANK2	000002	DMSDSV	DMSEDX										
BLANK3	000001	DMSEDX											
BLANK80	000007	DMSHLB	DMSHLI	DMSHLP									
BLK	000015	DMSBTP	DMSSEB	DMSSOP	DMSSQS	DMSTMA	DMSTPD						
BLKSIZE	000073	DMSDLK	DMSDSL	DMSFCH	DMSFLD	DMSFOR	DMSMVG	DMSNCP	DMSRDC	DMSTPE	DMSTPF	DMSTPG	
BLOC	000006	DMSEDI	DMSEDX										
BLOCKCNT	000012	DMSCLS	DMSTPE	DMSTPF	DMSTPG								
BLOCKLEN	000010	DMSFRE											
BRAD	000021	DMSLDR	DMSLSB	DMSLST	DMSOLD								
BRKPNTBL	000003	DMSDBG											
BRR8	000006	DMSTPE	DMSTPF	DMSTPG									
BS	000001	DMSCPF											
BSF	000005	DMSIFC	DMSTLB	DMSTPE	DMSTPF	DMSTPG							
BSR	000021	DMSBOP	DMSCLS	DMSTLB	DMSTPE	DMSTPF	DMSTPG						
BUFAD	000010	DMSCPY											

LABEL	COUNT	REFERENCES									
BUFFA	000013	DMSOVS	DMSUPD								
BUFFADR	000012	DMSTPE	DMSTPF	DMSTPG							
BUFFL	000001	DMSTLB									
BUFFLOC	000001	DMSSCR									
BUFF1	000013	DMSHLB	DMSHLP	DMSHLS	DMSLLU						
BUFF1LGZ	000009	DMSHLB	DMSHLP								
BUFF1M1	000011	DMSHLB	DMSHLI	DMSHLP							
BUFF2	000033	DMSBOP	DMSHLB	DMSHLP	DMSHLS	DMSLLU	DMSTLB				
BUFF2LG	000003	DMSHLP	DMSHLS								
BUFF2LGZ	000005	DMSHLB	DMSHLP								
BUFF2M1	000003	DMSHLI	DMSHLP								
BXBUFF	000009	DMSHLB									
BXBUFFND	000003	DMSHLB	DMSHLI								
BYTE	000004	DMSEDI	DMSNCP								
BYTESRD	000017	DMSTPE	DMSTPF	DMSTPG							
B1LG	000004	DMSHLB	DMSHLP								
B2LG	000001	DMSHLP									
CALLEE	000028	DMSERR	DMSITP	DMSITS	DMSLDR	DMSOVS					
CALLER	000008	DMSDOS	DMSFRE	DMSITS	DMSOVS	DMSSVT					
CARCTL	000009	DMSTPE	DMSTPF	DMSTPG							
CARDIN	000041	DMSDSK	DMSTPE	DMSTPF	DMSTPG						
CARDINCR	000003	DMSEDI	DMSEDX								
CARDNO	000003	DMSEDI									
CARDOUT	000035	DMSDSK	DMSTPE	DMSTPF	DMSTPG						
CASEREAD	000001	DMSEDI									
CASESW	000006	DMSEDI	DMSEDX								
CAW	000018	DMSCIO	DMSCIT	DMSDBD	DMSDBG	DMSDIO	DMSERR	DMSINI	DMSINS	DMSPIO	DMSSPR
CC	000470	DMSINI	DMSINS	DMSTIO							
CCBCCW	000004	DMSXCP									
CCBCNT	000017	DMSXCP									
CCBCOM1	000005	DMSXCP									
CCBCOM2	000011	DMSXCP									
CCBCSW	000004	DMSXCP									
CCBCSW1	000008	DMSXCP									
CCBCSW2	000002	DMSXCP									
CCBDC	000001	DMSXCP									
CCBEOC	000006	DMSXCP									
CCBEOF	000005	DMSXCP									
CCBERMAP	000016	DMSXCP									
CCBILEN	000003	DMSXCP									
CCBSUCLS	000002	DMSXCP									
CCBSUNUM	000002	DMSXCP									
CCBSYMU	000004	DMSXCP									
CCBUE	000007	DMSXCP									
CCBVER	000005	DMSXCP									
CCS	000001	DMSHLP									

LABEL	COUNT	REFERENCES										
CCWPRINT	000017	DMSDBD										
CCWX	000002	DMSDIO										
CCW1	000006	DMSDIO										
CCW1A	000004	DMSDIO										
CCW2	000003	DMSDIO	DMSOR3									
CDMSROS	000006	DMSACM	DMSALU									
CE	000004	DMSCIT	DMSINI									
CHANO	000002	DMSINI	DMSINS									
CHCKFILE	000006	DMSTPE	DMSTPF	DMSTPG								
CHECKSCN	000006	DMSTPE	DMSTPF	DMSTPG								
CHGTRUNC	000002	DMSEDI										
CHKEOF	000007	DMSCLS	DMSRNE	DMSTPE	DMSTPF	DMSTPG	DMSUTL					
CHKFT	000003	DMSTPE	DMSTPF	DMSTPG								
CHKFT10	000006	DMSTPE	DMSTPF	DMSTPG								
CHKINPUT	000006	DMSTPE	DMSTPF	DMSTPG								
CHKMODE	000013	DMSARE	DMSASN	DMSBOP	DMSLDS	DMSTPE	DMSTPF	DMSTPG				
CHKSCNSW	000009	DMSTPE	DMSTPF	DMSTPG								
CHKTYPE	000011	DMSDLK	DMSLDR	DMSOLD	DMSRDC	DMSTPE	DMSTPF	DMSTPG				
CHKWRD1	000002	DMSITS										
CHKWRD2	000002	DMSITS										
CHLINK	000012	DMSTPE	DMSTPF	DMSTPG								
CHNGBYTE	000011	DMSSVT										
CHNGCNT	000003	DMSEDI										
CHNGFLAG	000022	DMSEDI	DMSSCR									
CHNGMSG	000003	DMSEDI	DMSEDX									
CHNGNUM	000005	DMSEDI										
CL	000017	DMSCPY	DMSEXE	DMSFRE	DMSHLE	DMSMOD	DMSTPE	DMSTPF	DMSTPG			
CLASTAPE	000004	DMSASN	DMSTPE	DMSTPF	DMSTPG							
CLEAR	000020	DMSAMS	DMSDLB	DMSDLK	DMSFLD	DMSMOD	DMSSAB	DMSSYN	DMSTPE	DMSTPF	DMSTPG	DMSUTL
CLEAROP	000004	DMSLSB										
CLKVALMD	000005	DMSDOS	DMSFNS	DMSINS								
CLOSELIB	000016	DMSLDR	DMSLIB	DMSOLD	DMSZAP							
CLOSIO	000003	DMSPRT	DMSPUN	DMSRDC								
CLPAREN	000018	DMSFLD	DMSLBD	DMSTMA	DMSTPD	DMSTPE	DMSTPF	DMSTPG				
CLR	000008	DMSDLB	DMSFLD	DMSIFC	DMSLKD	DMSTPE	DMSTPF	DMSTPG				
CMD	000006	DMSLDR	DMSOLD									
CMDACT	000006	DMSTPE	DMSTPF	DMSTPG								
CMDBLOK	000002	DMSEDX	DMSGIO									
CMNDLINE	000020	DMSABN	DMSARX	DMSASM	DMSCPF	DMSEXI	DMSINS	DMSINT	DMSOSR	DMSSEB	DMSSVT	DMSXBG
CMNDLIST	000026	DMSCAT	DMSCPF	DMSEXI	DMSINS	DMSLDR	DMSOLD	DMSSCN				
CMODE	000021	DMSEDI	DMSOSR									
CMSAMS	000005	DMSAMS										
CMSBAM	000002	DMSSET										
CMSCVT	000004	DMSINS	DMSSOP									
CMSDOS	000002	DMSSET										
CMSNAME	000002	DMSSOP	DMSSVT									

LABEL	COUNT	REFERENCES											
CMSOP	000016	DMSDLB	DMSSCT	DMSSOP	DMSSVT								
CMSSEG	000023	DMSBTP	DMSEDX	DMSEXC	DMSHLL	DMSINS	DMSINT	DMSITS	DMSQRY	DMSSET	DMSTLA	DMSXSG	
CMSTAXE	000007	DMSCIT	DMSITE	DMSITI	DMSSVT								
CMSTIM	000007	DMSINT											
CMSVSAM	000011	DMSBOP	DMSDOS	DMSSET	DMSVIB								
CMSZER	000012	DMSINS											
CNTLADDR	000003	DMSTPE	DMSTPF	DMSTPG									
CODE	000014	DMSITS	DMSNCP	DMSSET									
CODE203	000294	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSAUD	DMSBOP	DMSBRD	DMSBWR	DMSCAT
		DMSCIT	DMSCLS	DMSCMP	DMSCRD	DMSCWR	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSEDX	DMSERD	DMSERS
		DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFET	DMSFNS	DMSFOR	DMSFRE	DMSHDI	DMSHDS	DMSINS
		DMSINT	DMSITE	DMSITP	DMSITS	DMSLAD	DMSLAF	DMSLDR	DMSLFS	DMSLGT	DMSLIB	DMSLSB	DMSMOD
		DMSOLD	DMSOPL	DMSOR1	DMSSAB	DMSSET	DMSSFF	DMSSLN	DMSSOP	DMSSTG	DMSSVN	DMSSVT	DMSSVU
		DMSTLB	DMSTRK	DMSVSR	DMSXCP								
COF	000001	DMSHLP											
COFF	000001	DMSHLB											
COMLINE	000003	DMSEXE											
COMMONEX	000006	DMSLDR	DMSOLD										
COMNAME	000014	DMSAMS	DMSDLK	DMSDOS	DMSDSV	DMSFCH	DMSFET						
COMPOPT	000003	DMSTPE	DMSTPF	DMSTPG									
COMPSWT	000016	DMSARN	DMSARX	DMSASM	DMSIFC	DMSSLN	DMSSMN	DMSSTG					
CON	000004	DMSCLS	DMSHLP	DMSOR1									
CONCCWS	000008	DMSCIT	DMSERR										
CONHCT	000003	DMSDBD	DMSDBG	DMSITE									
CONHXT	000002	DMSDBG											
CONINBLK	000004	DMSCRD											
CONINBUF	000005	DMSCRD											
CONRDBUF	000001	DMSSVN											
CONRDCNT	000009	DMSABN	DMSEXI	DMSINS	DMSINT	DMSSEB	DMSSVN	DMSSVT					
CONRDCOD	000007	DMSABN	DMSINS	DMSINT	DMSSEB	DMSSVN							
CONREAD	000011	DMSABN	DMSDLB	DMSFLD	DMSFNC	DMSINS	DMSINT	DMSLBD	DMSSEB	DMSSVN	DMSSVT	DMSTLB	
CONSOLE	000021	DMSEDI	DMSEDX	DMSINI									
CONSTACK	000013	DMSABN	DMSCIT	DMSCWR	DMSSVN								
CONTLSWT	000033	DMSTPE	DMSTPF	DMSTPG									
CONTROL	000070	DMSBOP	DMSNCP	DMSOR1	DMSSET	DMSSSK	DMSTPE	DMSTPF	DMSTPG	DMSVIB	DMSXCP		
CONVERT	000033	DMSBTP	DMSDLB	DMSDMP	DMSDOS	DMSFLD	DMSLDS	DMSRNE	DMSSVT	DMSTPE	DMSTPF	DMSTPG	DMSXCP
CONVERT1	000004	DMSQRY	DMSTPE	DMSTPF	DMSTPG								
CONVERT2	000008	DMSBOP	DMSCLS	DMSDOS	DMSQRY	DMSTPE	DMSTPF	DMSTPG	DMSXCP				
CONWAIT	000014	DMSABN	DMSBTP	DMSEDI	DMSEDX	DMSEXT	DMSFNC	DMSINT	DMSLDR				
CONWR	000005	DMSARX	DMSASM	DMSDBG	DMSSEB	DMSXCP							
CONWRBUF	000005	DMSINT	DMSSEB	DMSSVN	DMSSVT								
CONWRCNT	000004	DMSSEB	DMSSVN	DMSSVT									
CONWRCOD	000008	DMSINT	DMSSEB	DMSSVN									
CONWRITE	000005	DMSINT	DMSSEB	DMSSVN	DMSSVT								
CONWRL	000001	DMSDBG											
COPYEND	000021	DMSDSK	DMSTPE	DMSTPF	DMSTPG	DMSUTL							

LABEL	COUNT	REFERENCES											
COPYEOF	000004	DMSDSK	DMSTPE	DMSTPF	DMST[illegible]								
COPY010	000003	DMSTPE	DMSTPF	DMSTPG									
CORESIZE	000002	DMSSVT											
CORITEM	000007	DMSEDI	DMSEDX	DMSUPD									
COUNT	000078	DMSDBG	DMSEDI	DMSTQQ	DMSXCG								
CPCLOSE	000003	DMSTPE	DMSTPF	DMSTPG									
CPULOG	000005	DMSDBD	DMSSET										
CRBIT	000002	DMSEDI											
CRDPTR	000006	DMSLDR	DMSOLD										
CSINCLUD	000007	DMSHLI											
CSSAVE	000003	DMSHLP	DMSHLS										
CSSWS	000004	DMSHLP											
CSW	000071	DMSCIO	DMSCIT	DMSCRD	DMSCWR	DMSDBG	DMSDIO	DMSDLK	DMSFCH	DMSGIO	DMSINI	DMSIOW	DMSITE
		DMSITI	DMSLDS	DMSPIO	DMSROS	DMSSPR	DMSTIO	DMSTMA	DMSTPD	DMSXCP	DMSXPO	DMSXSC	
CS1	000002	DMSHLI											
CS2	000001	DMSHLI											
CS3	000001	DMSHLI											
CTL	000002	DMSUPD											
CURRALOC	000014	DMSXMA											
CURRCPUT	000001	DMSINM											
CURRDATE	000013	DMSDOS	DMSEXT	DMSINM	DMSINS	DMSSET	DMSSVT	DMSTLB	DMSXUP				
CURRIOOP	000003	DMSCIT											
CURRSAVE	000067	DMSABN	DMSACC	DMSDBG	DMSDLB	DMSDOS	DMSERR	DMSFLD	DMSFRE	DMSIFC	DMSITP	DMSITS	DMSLDR
		DMSOVS	DMSSAB	DMSSLN	DMSSMN	DMSSOP	DMSSTG	DMSSVN	DMSSVT	DMSTLB	DMSVIP		
CURRTIME	000002	DMSEXT	DMSXUP										
CURRVIRT	000002	DMSINM											
CVBSAVE	000002	DMSHLP											
CVHCLOSE	000001	DMSCVH											
CVHCOV	000001	DMSCVH											
CVHDLIST	000003	DMSCVH											
CVHFLAGS	000001	DMSCVH											
CVHFLG1	000006	DMSCVH											
CVHFLG2	000004	DMSCVH											
CVHIOA	000005	DMSCVH											
CVHNAME	000002	DMSCVH											
CVHOPEN	000004	DMSCVH											
CVHRADR	000001	DMSCVH											
CVHRETA	000001	DMSCVH											
CVHRF1	000001	DMSCVH											
CVHSCR	000001	DMSCVH											
CVHSYSNO	000002	DMSCVH											
CVHWADR	000001	DMSCVH											
CVHWANY	000001	DMSCVH											
CVTAVIB	000003	DMSINS	DMSSOP										
CVTMDL	000001	DMSINS											
CVTMZ00	000001	DMSINS											

LABEL	COUNT	REFERENCES
CVTNUCB	000001	DMSINS
CVTOPTA	000001	DMSINS
CVTSECT	000001	DMSINS
CW	000006	DMSTPE DMSTPF DMSTPG
CZERO	000007	DMSTPE DMSTPF DMSTPG DMSXMS
C1	000002	DMSCWR
C12	000001	DMSLDR
C6250	000003	DMSTPE DMSTPF DMSTPG
C7	000002	DMSLDR
C9	000001	DMSLDR
DA	000024	DMSDSL DMSMVE DMSMVG DMSNCP DMSSBD DMSSBS DMSSCT DMSSOP DMSUTL
DACTIVE	000010	DMSDOS DMSFCH DMSFET
DATAEND	000016	DMSSBD DMSSVU
DATAIN	000110	DMSDSK DMSTPE DMSTPF DMSTPG
DATAOUT	000040	DMSDSK DMSTPE DMSTPF DMSTPG
DATIPCMS	000006	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	000033	DMSDBD DMSDBG DMSITE
DBGPGMCK	000004	DMSDBG
DBGRECUR	000017	DMSDBD DMSDBG
DBGSAV1	000002	DMSDBG
DBGSAV2	000001	DMSDBG
DBGSECT	000006	DMSDBD DMSDBG DMSITE
DBGSET	000003	DMSDBG
DBGSHR	000001	DMSABN
DBGSWTCH	000012	DMSDBD DMSDBG
DBLWRD1	000015	DMSTIO DMSTPE DMSTPF DMSTPG
DBLWRD2	000017	DMSTIO DMSTPE DMSTPF DMSTPG
DBSWS	000002	DMSHLI DMSHLS
DCBSAV	000003	DMSSOP
DCHBWPTR	000033	DMSACF DMSACM DMSAUD DMSERS DMSFOR DMSINS DMSLAD DMSLFS DMSTRK
DCHCHGD	000034	DMSALU DMSAUD DMSDSK DMSERD DMSERS DMSFNS DMSRNM DMSTPE DMSTPF DMSTPG
DCHCHMAP	000007	DMSAUD DMSTRK
DCHCHOP	000012	DMSAUD DMSERS
DCHDA	000007	DMSERS
DCHDALLO	000006	DMSAUD
DCHDAMAP	000012	DMSACM DMSAUD DMSERD DMSFOR DMSLFS DMSTRK
DCHDATA	000137	DMSACC DMSACF DMSACM DMSALU DMSAUD DMSERD DMSERS DMSFNS DMSFOR DMSINS DMSLAD DMSLFS
		DMSLST DMSTRK
DCHDTSIZ	000053	DMSACC DMSACF DMSACM DMSALU DMSERD DMSERS DMSFOR DMSINS DMSLAD DMSLFS DMSLST DMSTRK

LABEL	COUNT	REFERENCES											
DCHDUM	000002	DMSINS											
DCHDWSIZ	000040	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD	DMSERD	DMSERS	DMSFNS	DMSFOR	DMSINS	DMSLAD
		DMSLFS	DMSTRK										
DCHFLG1	000049	DMSALU	DMSAUD	DMSDSK	DMSERD	DMSERS	DMSFNS	DMSFOR	DMSLFS	DMSRNM	DMSTPE	DMSTPF	DMSTPG
DCHFLG2	000024	DMSABN	DMSACC	DMSALU	DMSAUD	DMSERS	DMSINS	DMSTRK					
DCHFLG4	000005	DMSERS											
DCHFULL	000005	DMSACC	DMSAUD	DMSTRK									
DCHFWPTR	000094	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD	DMSERD	DMSERS	DMSFOR	DMSINS	DMSLAD	DMSLFS
		DMSLST	DMSTRK										
DCHLHBLK	000007	DMSERS											
DCHNEW	000001	DMSAUD											
DCHPFIXL	000067	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD	DMSERD	DMSERS	DMSFOR	DMSINS	DMSLAD	DMSLFS	DMSTRK
DCHRSV	000002	DMSERS											
DCHSECT	000186	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD	DMSDSK	DMSERD	DMSERS	DMSFNS	DMSFOR	DMSINS
		DMSLAD	DMSLFS	DMSLST	DMSRNM	DMSTPE	DMSTPF	DMSTPG	DMSTRK				
DCHSEQBD	000036	DMSACF	DMSACM	DMSERD	DMSERS	DMSFOR	DMSLFS	DMSTRK					
DCHSHR	000004	DMSABN	DMSALU										
DCHTDISP	000083	DMSACC	DMSACF	DMSACM	DMSAUD	DMSERD	DMSERS	DMSFNS	DMSFOR	DMSLFS			
DCHTRUNK	000071	DMSACC	DMSACF	DMSACM	DMSAUD	DMSERD	DMSERS	DMSFNS	DMSFOR	DMSLFS			
DCSSAVAL	000017	DMSEDX	DMSEXC	DMSHLL	DMSINS	DMSITS	DMSSAB	DMSSET	DMSTLA	DMSXSG			
DCSSFLAG	000049	DMSABN	DMSEDX	DMSEXC	DMSHLL	DMSINS	DMSINT	DMSITS	DMSSAB	DMSSET	DMSTLA	DMSXSG	
DCSSJLNS	000004	DMSINT	DMSSET										
DCSSLDED	000016	DMSEDX	DMSEXC	DMSHLL	DMSINT	DMSITS	DMSSET	DMSTLA	DMSXSG				
DCSSOVLP	000001	DMSINS											
DCSSVTLD	000018	DMSABN	DMSINS	DMSITS	DMSSAB	DMSSET							
DCTACYL	000002	DMSDOS											
DCTADR	000001	DMSDOS											
DCTBTRK	000002	DMSDOS											
DCTMAXR	000002	DMSDOS											
DCTPCYL	000003	DMSDOS											
DCTROH	000003	DMSDOS											
DCTTCYL	000002	DMSDOS											
DCTTPIX	000001	DMSDOS											
DCTUCBC	000001	DMSDOS											
DDNAM	000001	DMSMVE											
DE	000006	DMSCIT	DMSINI										
DEBDCBAD	000003	DMSBWR	DMSSAB	DMSSOP									
DEBDEBID	000001	DMSSOP											
DEBOPATB	000004	DMSSOP											
DEBTCBAD	000006	DMSSQS	DMSUTL										
DEC	000093	DMSBOP	DMSDBD	DMSDBG	DMSDLK	DMSDOS	DMSDSK	DMSDSV	DMSEDI	DMSEDX	DMSEXE	DMSLIB	DMSLST
		DMSOVR	DMSQRY	DMSSET	DMSSRT	DMSSSK	DMSTPD	DMSVIB					
DECAREA	000007	DMSSBD	DMSSBS										
DECDCBAD	000002	DMSSBS	DMSSCT										
DECDEC	000020	DMSDBD	DMSDBG	DMSITE	DMSQRY								
DECIMAL	000009	DMSEDI											

LABEL	COUNT	REFERENCES								
DECIOBPT	000003	DMSSBS	DMSSCT							
DECKYADR	000004	DMSSBD								
DECLNGTH	000005	DMSSBD	DMSSBS							
DECLTH	000002	DMSSCR								
DECM	000006	DMSHLP	DMSHLS							
DECRECPT	000002	DMSSBD								
DECSDECB	000024	DMSSBD	DMSSBS	DMSSCT	DMSSVT					
DECTYPE	000025	DMSSBD	DMSSBS							
DEFABS	000087	DMSXTB								
DEFNBUN	000030	DMSXTB								
DEF1IN7	000003	DMSHLP								
DEF2P31	000003	DMSXTB								
DELCOM	000002	DMSXTB								
DELTA7	000003	DMSHLP								
DENSITY	000021	DMSASN	DMSFLD	DMSTPE	DMSTPF	DMSTPG				
DEPTH	000013	DMSOVS								
DESC	000001	DMSHLI								
DESFTYPE	000002	DMSXIN								
DESLDESB	000001	DMSXIN								
DESLRECL	000001	DMSXIN								
DESRECFM	000002	DMSXIN								
DESSER	000001	DMSXIN								
DESSPEC	000001	DMSXIN								
DESTABS	000001	DMSXIN								
DESTRUNC	000001	DMSXIN								
DESTYP	000001	DMSXIN								
DESVERIF	000001	DMSXIN								
DEVADDR	000049	DMSFOR	DMSPRE	DMSTIO	DMSTMA	DMSTPD	DMSTPE	DMSTPF		
DEVFLAG	000001	DMSMVE								
DEVICE	000004	DMSARX	DMSASM	DMSIOW	DMSITI					
DEVMISC	000009	DMSTIO	DMSTPE	DMSTPF						
DEVNAME	000009	DMSTIO	DMSTMA	DMSTPD	DMSTPE	DMSTPF				
DEVSECT	000020	DMSLAB	DMSMVE	DMSPRE	DMSTIO	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTPF
DEVSIZE	000009	DMSTIO	DMSTMA	DMSTPD	DMSTPE	DMSTPF				
DEVTAB	000011	DMSASN	DMSDBD	DMSEDI	DMSEDX	DMSINI				
DEVTYP	000027	DMSDIO	DMSFNS	DMSLLU	DMSSOP					
DEVTYPE	000130	DMSASM	DMSASN	DMSDSV	DMSPRV	DMSRDC	DMSRRV	DMSSRV	DMSSVT	
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								

LABEL	COUNT	REFERENCES					
DIRBUF	000003	DMSHLS					
DIRBYTES	000001	DMSHLS					
DIRC	000017	DMSDOS	DMSFCH				
DIRDISK	000003	DMSHLS					
DIREEE	000001	DMSFCH					
DIRITEM	000035	DMSLBM	DMSLBT	DMSLIB	DMSPRT		
DIRITEMX	000015	DMSLIB	DMSPRT				
DIRLL	000005	DMSDOS	DMSFCH				
DIRMEMB	000003	DMSPRT					
DIRN	000006	DMSDOS	DMSFCH	DMSFET			
DIRNAME	000039	DMSDOS	DMSDSL	DMSFCH	DMSFET	DMSGND	DMSSVT
DIRPPP	000003	DMSFCH					
DIRPTR	000002	DMSSVT					
DIRR	000001	DMSDSL					
DIRREC	000002	DMSHLS					
DIRRR	000001	DMSFCH					
DIRSECT	000031	DMSLBM	DMSLIB	DMSPRT			
DIRSIZ	000002	DMSHLS					
DIRTT	000006	DMSDOS	DMSDSL	DMSFCH			
DIRTTR	000002	DMSFCH					
DIRTYP	000001	DMSHLS					
DISK$SEG	000008	DMSBRD	DMSFNS	DMSLFS			
DITCNT	000005	DMSEDI					
DLBLAREA	000001	DMSBOP					
DLCVHADR	000001	DMSCVH					
DLDLGOT	000002	DMSCVH					
DLFLAGS	000003	DMSCVH					
DLLEN	000002	DMSCVH					
DLOPENED	000001	DMSCVH					
DLSTDWDS	000002	DMSCVH					
DLSYSNO	000002	DMSCVH					
DMPTITLE	000003	DMSDBG					
DMSABN	000001	DMSZER					
DMSABNGO	000001	DMSITP					
DMSABNRT	000001	DMSDBG					
DMSABNSV	000001	DMSFNC					
DMSABW	000003	DMSDBG	DMSFRE	DMSITI			
DMSALU	000004	DMSFRE	DMSINS	DMSZER			
DMSARD	000001	DMSARX					
DMSASD	000001	DMSASM					
DMSBWR	000002	DMSFNC					
DMSCAT	000002	DMSFNC					
DMSCATMK	000002	DMSFNC					
DMSCATNB	000002	DMSFNC					
DMSCIO	000001	DMSZER					
DMSCIOSI	000002	DMSFNC					

LABEL	COUNT	REFERENCES											
DMSCITA	000001	DMSCWR											
DMSCITB	000002	DMSCRD	DMSCWR										
DMSCITDB	000002	DMSFNC											
DMSCITDK	000002	DMSFNC											
DMSCPF	000003	DMSFNC	DMSZER										
DMSCRD	000004	DMSFNC											
DMSCVH	000001	DMSDOS											
DMSCWR	000004	DMSDBG	DMSERR	DMSFNC									
DMSCWT	000006	DMSDBG	DMSERR	DMSFNC	DMSITS								
DMSDAS	000001	DMSDOS											
DMSDBD	000002	DMSDBG	DMSZER										
DMSDBG	000014	DMSABN	DMSFNC	DMSINS	DMSINT	DMSIOW	DMSITE	DMSQRY	DMSSET	DMSSMN	DMSSTG	DMSSVN	DMSSVT
		DMSZER											
DMSDBGP	000001	DMSINI											
DMSEDC	000001	DMSSEG											
DMSEDI	000001	DMSSEG											
DMSERDBF	000002	DMSBRD											
DMSERR	000057	DMSBWR	DMSCIT	DMSCRD	DMSCWR	DMSDBG	DMSERD	DMSERS	DMSFNC	DMSFNS	DMSFRE	DMSITS	DMSMOD
		DMSSTT											
DMSERT	000002	DMSERR											
DMSETR	000001	DMSDOS											
DMSEXC	000002	DMSFNC											
DMSEXE	000002	DMSEXI	DMSSEG										
DMSEXI	000001	DMSSEG											
DMSEXT	000002	DMSEXI	DMSSEG										
DMSFCH	000003	DMSDOS											
DMSFET	000003	DMSFNC	DMSZER										
DMSFNC	000001	DMSITS											
DMSFNC3	000001	DMSITS											
DMSFREB	000002	DMSFNC											
DMSFREES	000002	DMSFNC											
DMSFREEX	000002	DMSFNC											
DMSFRES	000003	DMSFNC	DMSINS										
DMSFRETS	000002	DMSFNC											
DMSFRETX	000001	DMSFNC											
DMSFRT	000002	DMSFRE											
DMSGIO	000002	DMSSCR	DMSSEG										
DMSHLD	000001	DMSHLI											
DMSINS	000001	DMSINI											
DMSINSE	000001	DMSINI											
DMSIOWR	000001	DMSDBG											
DMSITE	000001	DMSZER											
DMSITET	000002	DMSFNC											
DMSITP	000002	DMSDBG	DMSZER										
DMSITSK	000001	DMSFNC											
DMSITSSB	000002	DMSFNC											

LABEL	COUNT	REFERENCES				
DMSITSXS	000001	DMSFNC				
DMSITS1	000001	DMSINI				
DMSLAB	000001	DMSDOS				
DMSLAD	000005	DMSBWR	DMSERS	DMSINS	DMSLFS	DMSSTT
DMSLADAD	000002	DMSFNC				
DMSLADN	000002	DMSLFS				
DMSLADW	000002	DMSERS	DMSSTT			
DMSLBR	000001	DMSSET				
DMSLCK	000001	DMSDOS				
DMSLDRA	000002	DMSFNC				
DMSLDRB	000001	DMSLOA				
DMSLDRC	000001	DMSLSB				
DMSLDRD	000003	DMSLGT	DMSLIB	DMSLSB		
DMSLFS	000004	DMSBRD	DMSINT	DMSSTT		
DMSLFSW	000005	DMSBWR	DMSERS	DMSFNS	DMSSTT	
DMSLGT	000002	DMSSEG	DMSSVT			
DMSLGTA	000003	DMSLDR	DMSOLD	DMSSTG		
DMSLGTB	000002	DMSLDR	DMSOLD			
DMSLIB	000004	DMSLDR	DMSOLD	DMSSEG	DMSTMA	
DMSLIO	000001	DMSZER				
DMSLOA	000005	DMSFNC	DMSINS			
DMSLOS	000002	DMSSFF	DMSSLN			
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			
DMSMVG	000002	DMSMVE				
DMSOLD	000002	DMSSEG	DMSSLN			
DMSOVS	000001	DMSOVR				
DMSPIO	000005	DMSFNC	DMSZER			
DMSPIOCC	000002	DMSFNC				
DMSPIOSI	000002	DMSFNC				
DMSPNT	000001	DMSZER				
DMSPNTE	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			

LABEL	COUNT	REFERENCES		
DMSSCTCE	000002	DMSSOP	DMSSQS	
DMSSCTCK	000003	DMSSOP	DMSSQS	
DMSSCTNP	000001	DMSSOP		
DMSSEB	000005	DMSSBS	DMSSEG	DMSSQS
DMSSFF	000002	DMSSVT		
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		
DMSSQSUP	000001	DMSSOP		
DMSSTGAT	000002	DMSFNC		
DMSSTGCL	000001	DMSFNC		
DMSSTGSB	000003	DMSFNC	DMSLDR	DMSMOD
DMSSTGSV	000003	DMSFNC		
DMSSTTN	000002	DMSFNC		
DMSSTTNW	000002	DMSFNC		
DMSSTTR	000001	DMSLFS		
DMSSVN	000002	DMSSEG	DMSSVT	
DMSSVN1	000002	DMSSVT		
DMSSVN2	000002	DMSSVT		
DMSSVN93	000002	DMSSVT		
DMSSVN94	000002	DMSSVT		
DMSSVT	000001	DMSSEG		
DMSSVU	000002	DMSSEG	DMSSVT	
DMSTIO	000001	DMSZER		
DMSTLA	000001	DMSZER		
DMSTLABL	000002	DMSFNC		
DMSTLB	000002	DMSFNC	DMSSEG	
DMSTQQ	000001	DMSZER		
DMSVIB	000001	DMSZER		

LABEL	COUNT	REFERENCES
DMSVSR	000003	DMSFNC DMSZER
DMSXBG	000001	DMSSEG
DMSXCN	000008	DMSXCG DMSXFC DMSXFD
DMSXCP	000001	DMSDOS
DMSXCTPN	000002	DMSXSE
DMSXCTRS	000002	DMSXSE
DMSXCTSC	000002	DMSXSE
DMSXCTTE	000002	DMSXSE
DMSXDCOD	000006	DMSXSE DMSXSU
DMSXDCSY	000002	DMSXSE
DMSXDSRD	000002	DMSXIN
DMSXEDRT	000002	DMSXSU
DMSXER	000001	DMSXBG
DMSXFCBT	000004	DMSXSE
DMSXFCCC	000008	DMSXCT DMSXSC DMSXSE
DMSXFCCL	000008	DMSXCT DMSXSC DMSXSE
DMSXFCDC	000006	DMSXSD DMSXSE
DMSXFCDP	000002	DMSXSU
DMSXFCDR	000004	DMSXSE
DMSXFCGA	000004	DMSXSE
DMSXFCHT	000004	DMSXSE
DMSXFCIN	000036	DMSXCG DMSXDS DMSXGT DMSXIN DMSXMD DMSXPO DMSXPX DMSXUP
DMSXFCLM	000002	DMSXML
DMSXFCLR	000008	DMSXIN DMSXSE
DMSXFCML	000034	DMSXCG DMSXCM DMSXCT DMSXFD DMSXML DMSXPT
DMSXFCNX	000030	DMSXCG DMSXFD DMSXMD DMSXML DMSXPX DMSXSU DMSXUP
DMSXFCPC	000014	DMSXCG DMSXCT DMSXFD DMSXMC DMSXPX
DMSXFCPL	000088	DMSXCG DMSXDC DMSXFD DMSXMC DMSXMD DMSXML DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS DMSXSU DMSXUP
DMSXFCRC	000016	DMSXCG DMSXMD
DMSXFCRL	000008	DMSXCG DMSXMD DMSXSD
DMSXFCRM	000004	DMSXCG
DMSXFCSP	000012	DMSXCG DMSXMD DMSXML DMSXSD
DMSXFCSU	000028	DMSXCG DMSXDS DMSXGT DMSXIN DMSXMD DMSXPT DMSXPX DMSXSS DMSXUP
DMSXFCTB	000004	DMSXIN DMSXSE
DMSXFCTR	000002	DMSXSE
DMSXFCUP	000036	DMSXCG DMSXGT DMSXMD DMSXPO DMSXPX DMSXSU DMSXUP
DMSXFDFI	000002	DMSXCT
DMSXFDLN	000006	DMSXCT DMSXPX
DMSXFDSR	000002	DMSXUP
DMSXFDTG	000004	DMSXDC
DMSXINLD	000002	DMSXSU
DMSXINTP	000004	DMSXED DMSXSU
DMSXIORD	000010	DMSXCT DMSXMD DMSXPT DMSXSE DMSXSU
DMSXIOWR	000026	DMSXCT DMSXER DMSXSE DMSXSU DMSXUP
DMSXMAED	000001	DMSXSE

LABEL	COUNT	REFERENCES								
DMSXMAOP	000002	DMSXDC	DMSXIN							
DMSXMARD	000002	DMSXIN								
DMSXMARS	000004	DMSXBG	DMSXCT							
DMSXMCVR	000002	DMSXSE								
DMSXPTER	000001	DMSXSU								
DMSXPXDC	000004	DMSXSC	DMSXSS							
DMSXPXEX	000004	DMSXSC	DMSXSS							
DMSXSCCN	000002	DMSXPO								
DMSXSCDP	000002	DMSXIO								
DMSXSCIM	000003	DMSXBG	DMSXMD							
DMSXSCIO	000004	DMSXPO								
DMSXSCPR	000006	DMSXCT	DMSXSS							
DMSXSCRV	000008	DMSXCT	DMSXSE	DMSXSU						
DMSXSDAP	000003	DMSXBG	DMSXSE							
DMSXSDCT	000004	DMSXPO	DMSXSE							
DMSXSDLS	000004	DMSXBG	DMSXCT							
DMSXSDLW	000001	DMSXCG								
DMSXSDMK	000001	DMSXSE								
DMSXSDML	000004	DMSXSC	DMSXSS							
DMSXSDPH	000002	DMSXSC								
DMSXSDSC	000012	DMSXCT	DMSXML	DMSXSC	DMSXSE					
DMSXSDTB	000003	DMSXBG	DMSXSE							
DMSXSDTX	000003	DMSXBG	DMSXSE							
DMSXSDTY	000001	DMSXSC								
DMSXSDUP	000008	DMSXCG	DMSXCM	DMSXED	DMSXFD	DMSXHL	DMSXPO	DMSXSE		
DMSXSETB	000002	DMSXHL								
DMSXSSEX	000003	DMSXSC	DMSXSE							
DMSXSTCP	000002	DMSXED								
DMSXSTEX	000002	DMSXIN								
DMSXSTLG	000008	DMSXCN	DMSXFC	DMSXIN						
DMSXSTNB	000002	DMSXIN								
DMSXSUCC	000032	DMSXCG	DMSXCT	DMSXER	DMSXFC	DMSXSD	DMSXSE	DMSXSS	DMSXUP	
DMSXSUCH	000004	DMSXDC	DMSXFD							
DMSXSUCK	000028	DMSXCT	DMSXED	DMSXGT	DMSXHL	DMSXIN	DMSXPT	DMSXSE	DMSXUP	
DMSXSUCN	000036	DMSXDC	DMSXFC	DMSXFD	DMSXIN	DMSXMC	DMSXPX	DMSXSE	DMSXSS	DMSXUP
DMSXSUEF	000006	DMSXCG	DMSXML							
DMSXSUEX	000014	DMSXCT	DMSXIN	DMSXMA	DMSXMD	DMSXSS				
DMSXSUFL	000003	DMSXBG	DMSXIO							
DMSXSUHC	000002	DMSXER								
DMSXSUIG	000018	DMSXFC	DMSXPX	DMSXSC						
DMSXSULG	000006	DMSXCG	DMSXSE							
DMSXSULK	000004	DMSXCT	DMSXED							
DMSXSUNC	000020	DMSXCG	DMSXMD							
DMSXSUNF	000002	DMSXML								
DMSXSUNP	000002	DMSXDC								
DMSXSUPE	000002	DMSXED								

LABEL	COUNT	REFERENCES											
DMSXSUPR	000026	DMSXCG	DMSXGT	DMSXMD	DMSXPO	DMSXPT	DMSXSS						
DMSXSURV	000040	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXED	DMSXGT	DMSXIN	DMSXMC	DMSXPT	DMSXSE		
DMSXSUTE	000018	DMSXCG	DMSXCT	DMSXML	DMSXPT								
DMSXSUTF	000002	DMSXML											
DMSXSUTP	000022	DMSXCG	DMSXCM	DMSXPT									
DMSXSUTS	000002	DMSXIN											
DMSXSUTY	000058	DMSXCG	DMSXCT	DMSXDC	DMSXMC	DMSXMD	DMSXML						
DMSXSUVR	000002	DMSXBG											
DMSXTBHC	000001	DMSXDC											
DMSXTBRQ	000001	DMSXHL											
DMSXUPAT	000003	DMSXIN											
DMSXUPBL	000002	DMSXFD											
DMSXUPCK	000002	DMSXIN											
DMSXUPCT	000002	DMSXIN											
DMSXUPDL	000002	DMSXFC											
DMSZER	000002	DMSINS	DMSZES										
DMSZEX	000002	DMSINS	DMSZER										
DOSBAM	000005	DMSBOP	DMSCLS	DMSDOS	DMSSET								
DOSBLKSZ	000004	DMSBOP	DMSXCP										
DOSBUFF	000012	DMSBOP	DMSCVH	DMSXCP									
DOSBUFSP	000004	DMSDLB	DMSLAB	DMSQRY									
DOSBYTE	000013	DMSXCP											
DOSCBID	000005	DMSDLB	DMSROS	DMSXCP									
DOSCCHHR	000001	DMSXCP											
DOSCMS	000004	DMSDLB	DMSLAB										
DOSCOMP	000005	DMSFET	DMSLDR										
DOSCOUT	000002	DMSXCP											
DOSDD	000030	DMSAMS	DMSBOP	DMSCVH	DMSDLB	DMSDLK	DMSDSV	DMSFCH	DMSLAB	DMSOPL	DMSQRY	DMSRRV	DMSSRV
		DMSSVT	DMSVIP	DMSVLT									
DOSDDCAT	000006	DMSDLB											
DOSDEV	000019	DMSAMS	DMSBOP	DMSDLB	DMSDLK	DMSFCH	DMSLAB	DMSQRY	DMSRRV	DMSSRV	DMSVIP	DMSXCP	
DOSDIRC	000005	DMSSOP	DMSSVT										
DOSDOS	000004	DMSDLB	DMSQRY										
DOSDSK	000006	DMSDLB	DMSDLK	DMSEXT	DMSRRV	DMSSRV	DMSXCP						
DOSDSMD	000032	DMSAMS	DMSBOP	DMSCVH	DMSDLB	DMSLAB	DMSVIP	DMSXCP					
DOSDSNAM	000008	DMSDLB	DMSQRY	DMSXCP									
DOSDSTYP	000003	DMSCVH	DMSDLB	DMSQRY									
DOSDTF	000003	DMSBOP	DMSLAB	DMSXCP									
DOSDUM	000013	DMSAMS	DMSBOP	DMSDLB	DMSLAB	DMSQRY	DMSVIP						
DOSEND	000001	DMSDLB											
DOSENSIZ	000006	DMSDLB											
DOSEPL	000002	DMSBOP											
DOSEXT	000002	DMSBOP											
DOSEXTCX	000004	DMSLAB											
DOSEXTNO	000013	DMSAMS	DMSDLB	DMSLAB	DMSQRY	DMSVIP							
DOSEXTTB	000009	DMSAMS	DMSDLB	DMSLAB	DMSQRY	DMSVIP							

LABEL	COUNT	REFERENCES											
DOSFIRST	000029	DMSABN	DMSAMS	DMSBOP	DMSCVH	DMSDLB	DMSDLK	DMSDSV	DMSFCH	DMSLAB	DMSOPL	DMSQRY	DMSROS
		DMSRRV	DMSSRV	DMSSVT	DMSVIP	DMSVLT							
DOSFLAGS	000179	DMSABN	DMSALU	DMSAMS	DMSASM	DMSASN	DMSBOP	DMSBWR	DMSCPY	DMSDLB	DMSDLK	DMSDOS	DMSDSL
		DMSDSV	DMSEDI	DMSEDX	DMSEXT	DMSFCH	DMSFET	DMSHDI	DMSHDS	DMSIFC	DMSINT	DMSITE	DMSITP
		DMSITS	DMSLDR	DMSLDS	DMSLLU	DMSMOD	DMSMVE	DMSMVG	DMSOPT	DMSPIO	DMSPRV	DMSQRY	DMSROS
		DMSRRV	DMSSET	DMSSRT	DMSSRV	DMSSTG	DMSTLB	DMSTPD	DMSUPD	DMSVIP	DMSVSR	DMSXBG	DMSXCM
		DMSXCP	DMSZAP										
DOSFORM	000006	DMSBOP											
DOSF1AD	000009	DMSCVH	DMSVLT										
DOSINIT	000028	DMSBOP	DMSDLB	DMSLAB	DMSQRY	DMSXCP							
DOSITEM	000006	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	DMSBWR	DMSDLB	DMSDLK	DMSDSV	DMSEXT	DMSFET	DMSINT	DMSITP
		DMSLDR	DMSLLU	DMSMOD	DMSOPT	DMSPRV	DMSQRY	DMSRRV	DMSSET	DMSSRV	DMSVSR		
DOSNEXT	000013	DMSAMS	DMSBOP	DMSCVH	DMSDLB	DMSLAB	DMSOPL	DMSSVT	DMSVIP	DMSVLT			
DOSNUM	000014	DMSABN	DMSBOP	DMSDLB	DMSLAB	DMSQRY							
DOSOP	000048	DMSBOP	DMSCVH	DMSDLK	DMSDSV	DMSFCH	DMSOPL	DMSRRV	DMSSRV	DMSVLT	DMSXCP		
DOSOS	000006	DMSDLB	DMSQRY										
DOSOSDSN	000011	DMSDLB	DMSLAB	DMSQRY	DMSROS	DMSXCP							
DOSOSFST	000017	DMSBOP	DMSDLB	DMSDLK	DMSDSV	DMSFCH	DMSOPL	DMSROS	DMSRRV	DMSSRV	DMSXCP		
DOSPERM	000004	DMSDLB	DMSQRY										
DOSR	000001	DMSXCP											
DOSRC	000015	DMSAMS	DMSBAB	DMSBOP	DMSDOS	DMSFET	DMSLDR	DMSVIP					
DOSREAD	000004	DMSXCP											
DOSSAVE	000006	DMSIFC	DMSXCP										
DOSSECT	000040	DMSAMS	DMSBOP	DMSCLS	DMSCVH	DMSDLB	DMSDLK	DMSDOS	DMSDSV	DMSFCH	DMSLAB	DMSOPL	DMSQRY
		DMSROS	DMSRRV	DMSSRV	DMSSVT	DMSVIP	DMSVLT	DMSXCP					
DOSSENSE	000011	DMSXCP											
DOSSVC	000060	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	DMSTLB	DMSTPD	DMSUPD	DMSVIP	DMSVSR	DMSXBG	DMSXCM	DMSZAP	
DOSSYS	000004	DMSBOP	DMSDLB	DMSOPL	DMSQRY								
DOSTAPID	000002	DMSXCP											
DOSTRANS	000014	DMSABN	DMSBOP	DMSCLS	DMSDOS	DMSFCH	DMSSET						
DOSTYPE	000016	DMSDLB	DMSLAB	DMSQRY									
DOSUCAT	000006	DMSBOP	DMSDLB										
DOSUCNAM	000011	DMSBOP	DMSDLB	DMSLAB	DMSQRY								
DOSVOLNO	000015	DMSAMS	DMSDLB	DMSLAB	DMSQRY	DMSVIP							
DOSVOLTB	000009	DMSAMS	DMSDLB	DMSLAB	DMSQRY	DMSVIP							
DOSVSAM	000010	DMSASN	DMSBOP	DMSDOS	DMSFCH	DMSSET	DMSSTG						
DOSWORK	000006	DMSXCP											
DOSXXX	000004	DMSBOP	DMSDLB	DMSQRY									
DOSYSXXX	000011	DMSAMS	DMSDLB	DMSLAB	DMSVIP								

LABEL	COUNT	REFERENCES
DOUBLE	000017	DMSDIO DMSDLB DMSLBM DMSLBT
DRESET	000015	DMSTPE DMSTPF DMSTPG
DSKAD	000002	DMSLIO
DSKADR	000037	DMSACC DMSACF DMSACM DMSAUD DMSERD DMSERS DMSFNS DMSMOD
DSKADR2	000004	DMSERD
DSKCHAIN	000002	DMSERD
DSKLIN	000067	DMSEXT DMSLIO DMSMOD DMSSLN
DSKLOC	000023	DMSACC DMSACF DMSACM DMSAUD DMSERD DMSERS DMSFNS DMSMOD
DSKLOC2	000002	DMSERD
DSKLST	000039	DMSACC DMSACF DMSACM DMSAUD DMSERD DMSERS DMSFNS DMSHLS DMSLLU DMSMOD DMSPRV DMSRRV DMSSRV
DSKLST2	000004	DMSERD
DSKPTRSZ	000007	DMSERD
DSKPTRS2	000002	DMSERD
DSYM	000002	DMSLSY
DTAD	000034	DMSACC DMSACM DMSAMS DMSARE DMSASN DMSDIO DMSFOR DMSINS DMSQRY DMSROS
DTADC	000005	DMSACC DMSACM DMSASN DMSQRY
DTADT	000022	DMSACM DMSASN DMSAUD DMSDIO DMSQRY DMSTQQ
DTAS	000003	DMSAMS
DTFAVAIL	000003	DMSVLT
DTFBLHLD	000001	DMSBOP
DTFBLKSZ	000004	DMSBOP DMSMVG DMSVLT
DTFCCW	000003	DMSBOP DMSVLT
DTFCCWA	000001	DMSCLS
DTFCPDTL	000001	DMSBOP
DTFCSW	000001	DMSCLS
DTFCTRLF	000001	DMSBOP
DTFDEVTP	000006	DMSBOP
DTFFLG1	000003	DMSBOP DMSCLS DMSVLT
DTFFLG2	000012	DMSBOP DMSCLS DMSLAB DMSVLT DMSXCP
DTFFLG5	000001	DMSVLT
DTFFMT1R	000001	DMSCLS
DTFGVIOA	000001	DMSVLT
DTFIEND	000002	DMSMVG
DTFIGNOP	000002	DMSCLS DMSVLT
DTFINPUT	000005	DMSBOP DMSLAB DMSVLT
DTFIOA1	000001	DMSVLT
DTFLGMOD	000001	DMSVLT
DTFLOGRS	000001	DMSMVG
DTFNAME	000009	DMSBOP DMSMVG DMSVLT
DTFOPEN	000003	DMSBOP DMSCLS DMSVLT
DTFSD	000009	DMSBOP DMSCLS DMSLAB DMSOR1 DMSVLT DMSXCP
DTFTPDI	000001	DMSBOP
DTFTPSD	000001	DMSBOP
DTFTYPE	000002	DMSBOP
DTFWFUNB	000001	DMSBOP

LABEL	COUNT	REFERENCES											
DTFWKRLT	000001	DMSBOP											
DTFWRKFL	000002	DMSBOP	DMSVLT										
DTFX	000004	DMSBOP	DMSCLS	DMSVLT	DMSXCP								
DTFXBLSZ	000001	DMSVLT											
DTFXCCWP	000001	DMSVLT											
DTFXFBLP	000002	DMSVLT	DMSXCP										
DTFXIDEN	000003	DMSBOP	DMSCLS	DMSXCP									
DTFXLMPT	000001	DMSVLT											
DTFXOCWP	000002	DMSVLT											
DTFXORSP	000002	DMSBOP	DMSCLS										
DTFXRCIC	000002	DMSXCP											
DTFXSIO1	000001	DMSVLT											
DTFXSIO2	000001	DMSVLT											
DTFXXLEN	000001	DMSVLT											
DUALNOS	000008	DMSEDC											
DUMCOM	000004	DMSSLN											
DUMMY	000022	DMSARN	DMSARX	DMSASM	DMSDLB	DMSFLD	DMSGRN	DMSLAB	DMSQRY	DMSSBD	DMSSEB	DMSUTL	
DUMP	000008	DMSDSK	DMSEXE	DMSOPT	DMSQRY								
DUMPING	000001	DMSEXE											
DUMPIT	000003	DMSTPE	DMSTPF	DMSTPG									
DUMPLIST	000002	DMSDBG	DMSSVT										
DUMPMOD	000006	DMSTPE	DMSTPF	DMSTPG									
DUMPOK	000003	DMSTPE	DMSTPF	DMSTPG									
DUMPSWT	000027	DMSTPE	DMSTPF	DMSTPG									
DYLD	000013	DMSLDR	DMSLIO	DMSOLD	DMSSLN	DMSSTG							
DYLIBO	000005	DMSSLN	DMSSTG										
DYMBRNM	000006	DMSLIB	DMSSLN	DMSSTG									
DYNAEND	000004	DMSLDR	DMSOLD	DMSSLN									
D1	000546	DMSACC	DMSACF	DMSACM	DMSALU	DMSAUD	DMSBRD	DMSCAT	DMSCIT	DMSCPY	DMSCRD	DMSDAS	DMSDOS
		DMSDSK	DMSERS	DMSFLD	DMSFOR	DMSGND	DMSHLS	DMSIFC	DMSINI	DMSINS	DMSLCK	DMSPRT	DMSPUN
		DMSSRT	DMSSVT	DMSTPE	DMSTPF	DMSTPG	DMSTRK	DMSTYP	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCT
		DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC
		DMSXMD	DMSXMS	DMSXPO	DMSXPT	DMSXSC							
D2	000091	DMSACC	DMSCIT	DMSCPY	DMSDAS	DMSDOS	DMSERS	DMSFLD	DMSFOR	DMSHLL	DMSIFC	DMSINI	DMSINS
		DMSPRT	DMSREA	DMSTLB	DMSTPE	DMSTPF	DMSTPG	DMSUPD	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCT
		DMSXDC	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXIN	DMSXMA	DMSXMS	DMSXPO	DMSXSC		
D200	000004	DMSACC	DMSTPE	DMSTPF	DMSTPG								
D3	000126	DMSACC	DMSACF	DMSACM	DMSALU	DMSCAT	DMSCIT	DMSCRD	DMSERS	DMSFOR	DMSINI	DMSINS	DMSLCK
		DMSLFS	DMSLGT	DMSSVT	DMSTPE	DMSTPF	DMSTPG	DMSTRK	DMSUPD	DMSXBG	DMSXCG	DMSXCM	DMSXCT
		DMSXDC	DMSXED	DMSXFC	DMSXFD	DMSXGT	DMSXIN	DMSXMA	DMSXMC	DMSXMD	DMSXMS	DMSXPO	DMSXPT
		DMSXSC											
D556	000006	DMSTPE	DMSTPF	DMSTPG									
D6	000027	DMSCRD	DMSDSK	DMSFOR	DMSINI	DMSTPE	DMSTPF	DMSTPG	DMSUPD	DMSXCT	DMSXED	DMSXIN	DMSXSC
D6250	000006	DMSTPE	DMSTPF	DMSTPG									
D8007TRK	000009	DMSTPE	DMSTPF	DMSTPG									
D8009TRK	000018	DMSTPE	DMSTPF	DMSTPG									

LABEL	COUNT	REFERENCES									
EDCB	000005	DMSEDC	DMSEDI	DMSEDX	DMSGIO	DMSSCR					
EDCBEND	000001	DMSEDX									
EDCBLTH	000002	DMSEDX									
EDCT	000026	DMSEDI									
EDFCHAR	000002	DMSHLS									
EDFD010	000003	DMSTPE	DMSTPF	DMSTPG							
EDFD020	000003	DMSTPE	DMSTPF	DMSTPG							
EDFD040	000006	DMSTPE	DMSTPF	DMSTPG							
EDFL000	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL002	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL004	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL013	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL016	000006	DMSTPE	DMSTPF	DMSTPG							
EDFL020	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL030	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL040	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL050	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL060	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL100	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL105	000003	DMSTPE	DMSTPF	DMSTPG							
EDFL110	000006	DMSTPE	DMSTPF	DMSTPG							
EDFL120	000003	DMSTPE	DMSTPF	DMSTPG							
EDFSW	000003	DMSHLS									
EDF001	000003	DMSTPE	DMSTPF	DMSTPG							
EDF004	000003	DMSTPE	DMSTPF	DMSTPG							
EDF010	000006	DMSAUD	DMSDSK	DMSMOD	DMSTPE	DMSTPF	DMSTPG				
EDF015	000004	DMSDSK	DMSTPE	DMSTPF	DMSTPG						
EDF018	000004	DMSDSK	DMSTPE	DMSTPF	DMSTPG						
EDF020	000006	DMSAUD	DMSDSK	DMSMOD	DMSTPE	DMSTPF	DMSTPG				
EDF030	000004	DMSDSK	DMSTPE	DMSTPF	DMSTPG						
EDF040	000001	DMSDSK									
EDF080	000003	DMSTPE	DMSTPF	DMSTPG							
EDF090	000009	DMSTPE	DMSTPF	DMSTPG							
EDF110	000005	DMSFOR	DMSLFS	DMSTPE	DMSTPF	DMSTPG					
EDF120	000005	DMSFOR	DMSLFS	DMSTPE	DMSTPF	DMSTPG					
EDF180	000007	DMSFOR	DMSTPE	DMSTPF	DMSTPG						
EDF190	000003	DMSTPE	DMSTPF	DMSTPG							
EDF200	000008	DMSDSK	DMSTPE	DMSTPF	DMSTPG						
EDLIN	000013	DMSEDI	DMSEDX								
EDMSK	000003	DMSSCR									
EDRET	000003	DMSEDI	DMSEDX								
EDWORK	000002	DMSEDX									
EFPRS	000008	DMSITS	DMSOVS	DMSSVT							
EGPRS	000025	DMSABN	DMSITS	DMSOVS	DMSSLN						
EGPR0	000068	DMSDLB	DMSDOS	DMSFLD	DMSITS	DMSOVS	DMSSLN	DMSSOP	DMSSVN	DMSSVT	DMSSVU
EGPR1	000046	DMSDOS	DMSLDR	DMSSLN	DMSSMN	DMSSOP	DMSSVN	DMSSVT			

LABEL	COUNT	REFERENCES											
EGPR11	000002	DMSITS											
EGPR12	000003	DMSSTG											
EGPR13	000008	DMSSLN	DMSSVT										
EGPR14	000007	DMSDOS	DMSSLN	DMSSTG	DMSSVT								
EGPR15	000053	DMSDOS	DMSIFC	DMSITS	DMSOVS	DMSSLN	DMSSMN	DMSSOP	DMSSTG	DMSSVN	DMSSVT	DMSSVU	DMSXMA
EGPR2	000007	DMSITS	DMSSOP	DMSSVU	DMSTLB								
EGPR9	000004	DMSDOS											
EIGHT	000112	DMSACC	DMSBOP	DMSBWR	DMSCLS	DMSDOS	DMSDSK	DMSDSV	DMSEXT	DMSFCH	DMSLBM	DMSLDS	DMSLFS
		DMSMVG	DMSPIO	DMSRRV	DMSSVT	DMSSVU	DMSTLB	DMSTPE	DMSTPF	DMSTPG	DMSVIP	DMSXCP	
EIGHTOF	000002	DMSHLP											
EJECTRTN	000006	DMSTPE	DMSTPF	DMSTPG									
ENDBLOC	000003	DMSEDI	DMSEDX										
ENDCDADR	000006	DMSLDR	DMSLSB	DMSOLD									
ENDFREE	000002	DMSEXT	DMSLBT										
ENDLOAD	000001	DMSMOD											
ENDTABS	000006	DMSEDI	DMSEDX										
ENTADR	000008	DMSLDR	DMSOLD										
ENTNAME	000005	DMSLDR	DMSLSB	DMSOLD									
ENTRDWR	000018	DMSTPE	DMSTPF	DMSTPG									
ENTSIZ	000001	DMSHLI											
EOCADR	000006	DMSDMP	DMSSMN	DMSSTG									
EOFCHK	000005	DMSDSK	DMSLBT	DMSTPE	DMSTPF	DMSTPG							
EOFM	000003	DMSTPE	DMSTPF	DMSTPG									
EOFML	000003	DMSTPE	DMSTPF	DMSTPG									
EOFN	000012	DMSTPE	DMSTPF	DMSTPG									
EOFNEOT	000006	DMSTPE	DMSTPF	DMSTPG									
EOFSW	000008	DMSCLS	DMSHLS										
EOTF	000015	DMSTPE	DMSTPF	DMSTPG									
EPOINT	000006	DMSAMS	DMSFNC	DMSTPE	DMSTPF	DMSTPG	DMSVIP						
ERBIT	000011	DMSACF	DMSERS	DMSRNM									
ERBL	000001	DMSERR											
ERDSECT	000002	DMSERR											
ERENTRY	000006	DMSHLE											
ERF1BF	000002	DMSERR											
ERF1HD	000003	DMSERR											
ERF1SBN	000005	DMSERR											
ERF1SB1	000003	DMSERR											
ERF1TX	000002	DMSERR											
ERF2CM	000004	DMSERR											
ERF2DI	000001	DMSERR											
ERF2DT	000001	DMSERR											
ERF2PR	000001	DMSERR											
ERF2SI	000001	DMSERR											
ERG	000003	DMSTPE	DMSTPF	DMSTPG									
ERINDEX	000011	DMSHLB	DMSHLE	DMSHLI	DMSHLS								
ERLET	000001	DMSERR											

LABEL	COUNT	REFERENCES							
ERMESS	000002	DMSERR							
ERM01	000001	DMSHLS							
ERM02	000002	DMSHLP							
ERM03	000001	DMSHLP							
ERM04	000003	DMSHLP							
ERM05	000007	DMSHLB	DMSHLP						
ERM06	000003	DMSHLP							
ERM07	000001	DMSHLP							
ERM08	000001	DMSHLS							
ERM09	000001	DMSHLI							
ERM11	000005	DMSHLP							
ERM12	000001	DMSHLI							
ERM13	000002	DMSHLI	DMSHLS						
ERM14	000001	DMSHLS							
ERM15	000001	DMSHLI							
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	000010	DMSEXE	DMSEXT						
ERRBF	000011	DMSHLE	DMSHLI	DMSHLP	DMSHLS				
ERRCODE	000070	DMSACC	DMSDIO	DMSHDI	DMSHDS	DMSLBM	DMSPRE	DMSSAB	DMSSYN
ERRCOD0	000017	DMSACM							
ERRCOD1	000032	DMSACF	DMSERS	DMSRNM					
ERRDCONV	000003	DMSTPE	DMSTPF	DMSTPG					
ERRDLDNS	000009	DMSTPE	DMSTPF	DMSTPG					
ERRET	000041	DMSITS	DMSTLB						
ERRET10	000036	DMSTPE	DMSTPF	DMSTPG					
ERRET20	000018	DMSTPE	DMSTPF	DMSTPG					
ERRET30	000015	DMSTPE	DMSTPF	DMSTPG					
ERRET40	000006	DMSTPE	DMSTPF	DMSTPG					
ERRHIDEN	000006	DMSTPE	DMSTPF	DMSTPG					
ERRLG	000006	DMSHLE	DMSHLI	DMSHLP	DMSHLS				
ERRNUM	000002	DMSINT							
ERRORNN	000002	DMSHLE							
ERROROUT	000003	DMSTPE	DMSTPF	DMSTPG					
ERROR002	000009	DMSTPE	DMSTPF	DMSTPG					
ERROR003	000036	DMSTPE	DMSTPF	DMSTPG					
ERROR010	000003	DMSTPE	DMSTPF	DMSTPG					

LABEL	COUNT	REFERENCES								
ERROR014	000005	DMSQRY	DMSSET	DMSTPE	DMSTPF	DMSTPG				
ERROR017	000006	DMSTPE	DMSTPF	DMSTPG						
ERROR023	000006	DMSTPE	DMSTPF	DMSTPG						
ERROR027	000006	DMSTPE	DMSTPF	DMSTPG						
ERROR029	000033	DMSTPE	DMSTPF	DMSTPG						
ERROR037	000003	DMSTPE	DMSTPF	DMSTPG						
ERROR042	000003	DMSTPE	DMSTPF	DMSTPG						
ERROR043	000003	DMSTPE	DMSTPF	DMSTPG						
ERROR047	000008	DMSQRY	DMSSET	DMSTPE	DMSTPF	DMSTPG				
ERROR048	000008	DMSSET	DMSTPE	DMSTPF	DMSTPG					
ERROR057	000012	DMSTPE	DMSTPF	DMSTPG						
ERROR058	000003	DMSTPE	DMSTPF	DMSTPG						
ERROR070	000044	DMSQRY	DMSSET	DMSTPE	DMSTPF	DMSTPG				
ERROR096	000003	DMSTPE	DMSTPF	DMSTPG						
ERROR1	000031	DMSACM	DMSARE	DMSERD	DMSEXE	DMSINA	DMSLAD	DMSPIO	DMSTPE	DMSTPF
		DMSTPG	DMSTRK	DMSVSR						
ERROR104	000007	DMSAMS	DMSTPE	DMSTPF	DMSTPG					
ERROR105	000016	DMSTMA	DMSTPE	DMSTPF	DMSTPG					
ERROR110	000018	DMSTMA	DMSTPE	DMSTPF	DMSTPG					
ERROR111	000015	DMSTPE	DMSTPF	DMSTPG						
ERROR113	000003	DMSTPE	DMSTPF	DMSTPG						
ERROR2	000025	DMSARE	DMSBRD	DMSCVH	DMSEXE	DMSINA	DMSLFS	DMSPIO	DMSTPE	DMSTPF
		DMSTPG	DMSTRK							
		DMSVLT								
ERROR3	000061	DMSACM	DMSBWR	DMSCIO	DMSERD	DMSERS	DMSEXE	DMSPIO	DMSTPE	DMSTPF
		DMSTPG								
ERROR431	000012	DMSTPE	DMSTPF	DMSTPG						
ERROR47M	000006	DMSTPE	DMSTPF	DMSTPG						
ERROR70M	000006	DMSTPE	DMSTPF	DMSTPG						
ERRO105M	000003	DMSTPE	DMSTPF	DMSTPG						
ERRO110B	000003	DMSTPE	DMSTPF	DMSTPG						
ERRO111B	000012	DMSTPE	DMSTPF	DMSTPG						
ERRO111M	000003	DMSTPE	DMSTPF	DMSTPG						
ERRTRANS	000003	DMSTPE	DMSTPF	DMSTPG						
ERR1	000028	DMSARX	DMSASM	DMSBOP	DMSERS	DMSFET	DMSFOR	DMSINA	DMSLST	DMSNCP
		DMSRNE	DMSSYN	DMSTPE						
		DMSTPF	DMSTPG	DMSTYP	DMSVSR					
ERR104	000008	DMSDSL	DMSFCH	DMSRNE	DMSTPE	DMSTPF	DMSTPG			
ERR111	000009	DMSTPE	DMSTPF	DMSTPG						
ERR115S	000039	DMSTPE	DMSTPF	DMSTPG						
ERR16BP	000006	DMSTPE	DMSTPF	DMSTPG						
ERR2RC	000003	DMSTPE	DMSTPF	DMSTPG						
ERR7TRK	000003	DMSTPE	DMSTPF	DMSTPG						
ERR70E	000049	DMSASN	DMSDLB	DMSDSK	DMSFLD	DMSLBD	DMSSRT	DMSTPE	DMSTPF	DMSTPG
ERR800BP	000015	DMSTPE	DMSTPF	DMSTPG						
ERR9TRK	000003	DMSTPE	DMSTPF	DMSTPG						
ERSAVE	000007	DMSERR								
ERSBD	000013	DMSERR								
ERSBF	000010	DMSERR								
ERSBL	000005	DMSERR								

LABEL	COUNT	REFERENCES			
ERSECT	000001	DMSERR			
ERSFA	000004	DMSERR			
ERSFL	000005	DMSERR			
ERSFLAG	000068	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			
ERT2	000013	DMSERR			
ESD1ST	000011	DMSDLK	DMSLDR	DMSOLD	
ESIDTB	000040	DMSLDR	DMSOLD		
EUA	000001	DMSXSC			
EXADD	000012	DMSEXC	DMSEXT		
EXAMLC	000005	DMSDBG			
EXAMLG	000006	DMSDBG			
EXBUF	000001	DMSHLI			
EXC2	000001	DMSHLI			
EXECFLAG	000003	DMSEXC			
EXECRUN	000004	DMSEXC	DMSGRN		
EXECSVC1	000003	DMSTPE	DMSTPF	DMSTPG	
EXECSVC2	000003	DMSTPE	DMSTPF	DMSTPG	
EXEC2	000004	DMSEXE	DMSHLI	DMSLDR	DMSOSR
EXENACTB	000009	DMSVIP			
EXENADDR	000002	DMSVIP			
EXLEODF	000004	DMSVIP			
EXLEODL	000001	DMSVIP			
EXLEODP	000001	DMSVIP			
EXLEVEL	000005	DMSEXC			
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	000005	DMSEXC			
EXSAVE	000009	DMSITE			
EXSAVE1	000007	DMSITE			
EXTFLAG	000006	DMSIOW	DMSITE	DMSSVN	
EXTM	000001	DMSQRY			

LABEL	COUNT	REFERENCES											
EXTNPSW	000002	DMSINI	DMSINS										
EXTOPSW	000021	DMSDBG	DMSITE										
EXTPSW	000005	DMSINT	DMSITE										
EXTRET	000007	DMSITE											
EXTSECT	000014	DMSINS	DMSINT	DMSIOW	DMSITE	DMSQRY	DMSSET	DMSSTG	DMSSVN	DMSSVT			
EXTYPEOI	000003	DMSTPE	DMSTPF	DMSTPG									
FBACCWL1	000001	DMSINI											
FBACD1	000006	DMSACM	DMSDIO	DMSFNS	DMSFOR	DMSINS	DMSITI						
FBACL1	000006	DMSACM	DMSDIO	DMSFNS	DMSFOR	DMSINS	DMSITI						
FBADEF	000007	DMSINI											
FBADWDT	000001	DMSINI											
FBAIPL	000001	DMSINI											
FBALOC	000008	DMSINI											
FBALWDT	000001	DMSINI											
FBARD	000007	DMSINI											
FBAWR	000002	DMSINI											
FBLOCK	000011	DMSDSK	DMSERD	DMSEXE	DMSTPE	DMSTPF	DMSTPG						
FCBBLKCT	000013	DMSSBS	DMSSEB	DMSSOP	DMSTLB								
FCBBLKSZ	000010	DMSFLD	DMSMVE	DMSMVG	DMSROS	DMSSOP	DMSTLB	DMSUTL					
FCBBLP	000003	DMSFLD	DMSSOP										
FCBBUFF	000052	DMSARN	DMSARX	DMSASM	DMSLOS	DMSSBS	DMSSEB	DMSSOP	DMSSQS	DMSSVT	DMSSVU		
FCBBYTE	000061	DMSARN	DMSARX	DMSASM	DMSLOS	DMSSBD	DMSSBS	DMSSEB	DMSSOP	DMSSQS	DMSSVT	DMSSVU	
FCBCASE	000005	DMSFLD	DMSSEB	DMSSOP									
FCBCATLD	000017	DMSFLD	DMSSCT	DMSSOP	DMSSVT	DMSUTL							
FCBCATML	000024	DMSARN	DMSARX	DMSASM	DMSFLD	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSVT			
FCBCLEAV	000004	DMSSOP											
FCBCLOSE	000011	DMSARN	DMSARX	DMSASM	DMSSCT	DMSSOP	DMSSQS						
FCBCON	000003	DMSFLD	DMSSOP										
FCBCOUT	000027	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVT	DMSSVU					
FCBDCBCT	000004	DMSSOP											
FCBDD	000031	DMSARN	DMSARX	DMSASM	DMSBWR	DMSFLD	DMSLBD	DMSLOS	DMSMVE	DMSQRY	DMSSAB	DMSSOP	DMSSVT
		DMSTLB	DMSUTL	DMSXDS									
FCBDEV	000059	DMSARN	DMSARX	DMSASM	DMSFLD	DMSLBD	DMSMVE	DMSQRY	DMSSAB	DMSSBS	DMSSCT	DMSSEB	DMSSOP
		DMSSQS	DMSSVT	DMSUTL									
FCBDOSL	000011	DMSFLD	DMSSOP	DMSSVT									
FCBDSK	000012	DMSARX	DMSASM	DMSFCH	DMSFLD	DMSMVE	DMSSOP	DMSSVT					
FCBDSMD	000041	DMSALU	DMSFLD	DMSMVE	DMSROS	DMSSBS	DMSSEB	DMSSOP	DMSSQS	DMSSVT			
FCBDSNAM	000064	DMSARX	DMSASM	DMSFCH	DMSFLD	DMSLOS	DMSMVE	DMSQRY	DMSROS	DMSSBS	DMSSEB	DMSSOP	DMSSVT
		DMSSVU	DMSUTL	DMSXDS									
FCBDSORG	000004	DMSFLD											
FCBDSTYP	000022	DMSFLD	DMSQRY	DMSROS	DMSSEB	DMSSOP	DMSSVU	DMSUTL					
FCBDUM	000005	DMSFLD	DMSSAB	DMSSOP	DMSSVT								
FCBEND	000001	DMSFLD											
FCBENSIZ	000006	DMSFLD											
FCBEPL	000001	DMSSOP											
FCBFIRST	000020	DMSABN	DMSALU	DMSBWR	DMSFLD	DMSLBD	DMSLOS	DMSQRY	DMSROS	DMSSAB	DMSSOP	DMSSVT	DMSXDS

LABEL	COUNT	REFERENCES
FCBFORM	000014	DMSARN DMSARX DMSASM DMSSEB DMSSOP DMSSVT DMSSVU
FCBINIT	000093	DMSARN DMSARX DMSASM DMSFCH DMSFLD DMSLOS DMSMVE DMSSBS DMSSCT DMSSEB DMSSOP DMSSQS
		DMSSVT DMSUTL
FCBIO	000001	DMSSEB
FCBIORD	000003	DMSSQS
FCBIOSW	000034	DMSARN DMSARX DMSASM DMSFLD DMSSCT DMSSEB DMSSOP DMSSQS
FCBIOSW2	000027	DMSDSL DMSLDS DMSMVE DMSROS DMSSEB DMSSOP DMSSVT DMSUTL
FCBIOWR	000003	DMSSQS
FCBITEM	000065	DMSARN DMSARX DMSASM DMSDSL DMSLOS DMSMVE DMSSBD DMSSBS DMSSCT DMSSEB DMSSOP DMSSQS
		DMSSVT DMSSVU
FCBKEYS	000009	DMSSBD DMSSOP DMSSVU
FCBLABPT	000009	DMSFLD DMSLBD DMSQRY DMSTLB
FCBLABT	000039	DMSFLD DMSLBD DMSQRY DMSSEB DMSSOP
FCBLEAVE	000003	DMSFLD DMSSOP DMSTLB
FCBLRECL	000008	DMSFLD DMSMVE DMSMVG DMSROS DMSSOP DMSTLB
FCBMEMBR	000024	DMSFLD DMSLDS DMSROS DMSSCT DMSSEB DMSSOP DMSSVT
FCBMMV	000004	DMSMVE DMSSVT
FCBMODE	000017	DMSFLD DMSSBS DMSSEB DMSSOP DMSTLB
FCBMVFIL	000002	DMSMVE DMSSEB
FCBMVPDS	000020	DMSDSL DMSLDS DMSMVE DMSROS DMSSEB DMSSOP DMSSVT DMSUTL
FCBNEXT	000006	DMSALU DMSFLD DMSLBD DMSLOS DMSROS
FCBNL	000004	DMSFLD DMSQRY DMSSOP
FCBNOEOV	000002	DMSFLD DMSSEB
FCBNSL	000009	DMSFLD DMSQRY DMSSEB DMSSOP
FCBNSLMD	000002	DMSFLD
FCBNSLNM	000006	DMSFLD DMSQRY DMSSEB DMSSOP
FCBNUM	000015	DMSABN DMSBWR DMSFLD DMSLBD DMSQRY
FCBOFF	000006	DMSFLD DMSQRY DMSSEB DMSSOP
FCBOP	000120	DMSMVE DMSROS DMSSBD DMSSBS DMSSCT DMSSEB DMSSOP DMSSQS DMSSVT DMSSVU
FCBOPCB	000006	DMSLOS DMSMVE DMSSEB
FCBOS	000017	DMSSBS DMSSCT DMSSEB DMSSOP DMSSVT
FCBOSDSN	000018	DMSFLD DMSLDS DMSROS
FCBOSFST	000020	DMSALU DMSMVE DMSMVG DMSROS DMSSCT DMSSOP DMSSVT
FCBPCH	000002	DMSFLD
FCBPDS	000012	DMSSBS DMSSCT DMSSOP DMSSVT
FCBPOS	000007	DMSFLD DMSQRY DMSSOP DMSTLB
FCBPROC	000010	DMSARN DMSFLD DMSROS DMSSEB DMSSOP
FCBPROCC	000005	DMSARN DMSARX DMSASM DMSSOP
FCBPROCO	000003	DMSARN DMSSOP
FCBPRPU	000006	DMSSEB
FCBPTR	000003	DMSFLD DMSSOP
FCBPVMB	000003	DMSSQS
FCBRDR	000005	DMSARX DMSASM DMSFLD DMSSOP
FCBREAD	000023	DMSARN DMSARX DMSASM DMSSBS DMSSEB DMSSQS
FCBRECFM	000013	DMSFLD DMSMVE DMSMVG DMSROS DMSSBD DMSSEB DMSSOP DMSTLB
FCBRECL	000006	DMSSEB DMSSOP DMSXDS

LABEL	COUNT	REFERENCES											
FCBRPTR	000001	DMSSOP											
FCBR13	000002	DMSSCT	DMSSEB										
FCBSECT	000064	DMSALU	DMSARN	DMSARX	DMSASM	DMSBWR	DMSDSL	DMSFCH	DMSFLD	DMSLBD	DMSLDS	DMSLOS	DMSMVE
		DMSMVG	DMSQRY	DMSROS	DMSSAB	DMSSBD	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVN	DMSSVT
		DMSSVU	DMSTLB	DMSUTL	DMSXDS								
FCBSL	000010	DMSFLD	DMSLBD	DMSQRY	DMSSEB	DMSSOP							
FCBTAB	000003	DMSSVT	DMSUTL										
FCBTAP	000014	DMSARX	DMSASM	DMSFLD	DMSLBD	DMSMVE	DMSSBS	DMSSCT	DMSSOP	DMSSVT			
FCBTAPID	000017	DMSFLD	DMSMVE	DMSQRY	DMSSEB	DMSSOP							
FCBTBSP	000004	DMSSBS	DMSSVT										
FCBTCLOS	000003	DMSSOP											
FCBTPSW	000005	DMSFLD	DMSSEB	DMSSOP	DMSTLB								
FCBXTENT	000011	DMSFLD	DMSSBD	DMSSBS	DMSSOP	DMSSVU							
FCHAPHNM	000002	DMSFET											
FCHLENG	000005	DMSABN	DMSDOS	DMSFET	DMSLOS								
FCHOPT	000002	DMSFET											
FCHTAB	000008	DMSDOS	DMSFET										
FDIAG	000045	DMSTPE	DMSTPF	DMSTPG									
FEIGHT	000001	DMSHLP											
FF	000075	DMSALU	DMSAMS	DMSARE	DMSASN	DMSBOP	DMSBWR	DMSEXT	DMSGRN	DMSHLP	DMSOR2	DMSSET	DMSSOP
		DMSSVT	DMSTPE	DMSTPF	DMSTPG	DMSUTL	DMSXCP						
FFD	000005	DMSACM	DMSAUD	DMSEXC									
FFE	000002	DMSACM	DMSAUD										
FFF	000004	DMSACM	DMSAUD										
FFIFTEEN	000004	DMSHLB	DMSHLE	DMSHLP									
FFORMAT	000013	DMSERD	DMSTPE	DMSTPF	DMSTPG								
FFOUR	000001	DMSHLB											
FILE	000085	DMSACM	DMSARX	DMSASM	DMSBOP	DMSCLS	DMSCMP	DMSDLB	DMSDSK	DMSDSL	DMSEDI	DMSEDX	DMSEXE
		DMSFLD	DMSGLB	DMSGND	DMSIFC	DMSIMA	DMSLBM	DMSLBT	DMSLGT	DMSLIB	DMSLIO	DMSLKD	DMSMOD
		DMSNCP	DMSPRT	DMSPUN	DMSRDC	DMSRNM	DMSSLN	DMSSTT	DMSSYN	DMSTLB	DMSTPD	DMSTPE	DMSTPF
		DMSTPG	DMSTYP	DMSXTB	DMSZAP								
FILEBUFF	000025	DMSEXC	DMSPRT	DMSPUN	DMSRDC	DMSROS	DMSSVT	DMSSVU	DMSTPD				
FILEBYTE	000007	DMSEXC	DMSROS	DMSSOP	DMSSVT								
FILECOUT	000002	DMSSVT											
FILEITEM	000007	DMSSVT											
FILEMODE	000015	DMSEXC	DMSEXE	DMSNCP	DMSPRT	DMSPUN	DMSRDC	DMSSOP	DMSSVT	DMSTPD			
FILEMS	000006	DMSEDI											
FILENAME	000049	DMSEXE	DMSINT	DMSNCP	DMSPRT	DMSPUN	DMSRDC	DMSROS	DMSSCT	DMSSOP	DMSSVT	DMSSVU	DMSTPD
FILEREAD	000002	DMSROS	DMSSOP										
FILETYPE	000014	DMSBOP	DMSEXE	DMSINT	DMSPRT	DMSPUN	DMSSOP	DMSSVT	DMSSVU	DMSTPD			
FILNAM	000012	DMSBRD	DMSHLI	DMSHLS	DMSMDP								
FILNUM	000003	DMSHLS											
FILTYP	000004	DMSHLI											
FINE	000005	DMSTLB	DMSTPE	DMSTPF	DMSTPG								
FINIS	000061	DMSARN	DMSFNC	DMSFRE	DMSLBT	DMSLDR	DMSLIB	DMSLLU	DMSOLD	DMSSBD	DMSSRT	DMSTMA	
FINISLST	000004	DMSAUD	DMSFNS	DMSINT									

LABEL	COUNT	REFERENCES											
FIRSTDMP	000002	DMSDBG											
FIRSTOPT	000003	DMSTPE	DMSTPF	DMSTPG									
FKE3278	000001	DMSXBG											
FLAG	000190	DMSEDI	DMSEDX	DMSEXE	DMSEXT	DMSFOR	DMSLST	DMSMVE	DMSSCR	DMSSRT	DMSSVT	DMSTPD	DMSUTL
FLAGIN	000009	DMSTPE	DMSTPF	DMSTPG									
FLAGLOC	000004	DMSEDX	DMSSCR										
FLAGS	000233	DMSFRE	DMSITS	DMSLBM	DMSLBT	DMSLDR	DMSLIB	DMSLSB	DMSLST	DMSOLD	DMSOVS	DMSTPE	DMSTPF
		DMSTPG	DMSZAP										
FLAGS2	000053	DMSLBM	DMSLBT	DMSTPE	DMSTPF	DMSTPG							
FLAG1	000079	DMSARX	DMSASM	DMSEXT	DMSFLD	DMSLDR	DMSLIO	DMSLSB	DMSOLD				
FLAG2	000211	DMSARX	DMSASM	DMSASN	DMSEDI	DMSEDX	DMSFLD	DMSLDR	DMSLIB	DMSLIO	DMSLSB	DMSOLD	DMSSCR
		DMSTPD	DMSTPE	DMSTPF	DMSTPG	DMSUTL							
FLAG3	000027	DMSASN	DMSEDI	DMSFLD	DMSLDR	DMSOLD	DMSTPD						
FLCLN	000011	DMSFRE											
FLDMRK	000003	DMSXSC	DMSXSD										
FLGSAVE	000002	DMSALU											
FLHC	000008	DMSFRE											
FLNU	000007	DMSFRE											
FLPA	000016	DMSFRE											
FMACT	000003	DMSTPE	DMSTPF	DMSTPG									
FMODE	000048	DMSEDI	DMSEDX	DMSEXT	DMSLDS	DMSLGT	DMSLIB	DMSLST	DMSRDC	DMSRNE	DMSSCR	DMSTYP	DMSXSE
FMOK1	000003	DMSTPE	DMSTPF	DMSTPG									
FMOK2	000006	DMSTPE	DMSTPF	DMSTPG									
FNACT	000003	DMSTPE	DMSTPF	DMSTPG									
FNAME	000054	DMSDSK	DMSEDI	DMSEDX	DMSEXT	DMSLGT	DMSLIB	DMSLIO	DMSLST	DMSPRV	DMSRNE	DMSRRV	DMSSCR
		DMSSRV	DMSTYP	DMSUPD	DMSXSE								
FNBIT	000004	DMSFNS											
FNINE	000001	DMSHLB											
FNONE	000002	DMSHLP											
FONE	000001	DMSHLS											
FORM	000027	DMSDSV	DMSGRN	DMSHLI	DMSIFC	DMSLST	DMSNCP	DMSOVR	DMSSLN	DMSUPD			
FORMOK1	000003	DMSTPE	DMSTPF	DMSTPG									
FORMOK2	000003	DMSTPE	DMSTPF	DMSTPG									
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	000015	DMSBOP	DMSFRE										
FREELOWE	000062	DMSABN	DMSARX	DMSASM	DMSDLK	DMSDOS	DMSDSV	DMSERD	DMSFCH	DMSFRE	DMSINS	DMSINT	DMSLBM
		DMSLDR	DMSLSB	DMSMOD	DMSNCP	DMSOLD	DMSSET	DMSSLN	DMSSMN	DMSSTG	DMSSVT		
FREELOWR	000001	DMSFRE											

LABEL	COUNT	REFERENCES											
FREELOW1	000006	DMSFRE	DMSSET										
FREELU	000006	DMSFRE											
FREERO	000003	DMSDIO											
FREESAVE	000013	DMSFRE											
FREESTOR	000034	DMSBOP	DMSCLS	DMSDAS	DMSOPL	DMSOR1	DMSTPE	DMSTPF	DMSTPG				
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	000009	DMSMOD	DMSSLN	DMSXIN									
FRSTSDID	000002	DMSLDR	DMSLSB										
FSCBAITN	000015	DMSDLK	DMSEXE										
FSCBANIT	000003	DMSEXE											
FSCBBUFF	000010	DMSDLK	DMSEXE	DMSIFC	DMSZAP								
FSCBD	000030	DMSBRD	DMSDLK	DMSEXE	DMSIFC	DMSZAP							
FSCBEPL	000004	DMSEXE	DMSLGT										
FSCBFLG	000008	DMSBRD	DMSEXE										
FSCBFM	000011	DMSDLK	DMSEXE	DMSIFC	DMSZAP								
FSCBFN	000031	DMSDLK	DMSEXE	DMSIFC	DMSZAP								
FSCBFT	000010	DMSEXE	DMSZAP										
FSCBFV	000015	DMSBRD	DMSDLK	DMSEXE	DMSIFC	DMSLGT	DMSZAP						
FSCBITNO	000006	DMSDLK	DMSEXE										
FSCBNOIT	000002	DMSEXE											
FSCBNORD	000002	DMSEXE											
FSCBSIZE	000002	DMSEXE											
FSF	000009	DMSBOP	DMSCLS	DMSIFC	DMSTLB								
FSIZE	000012	DMSEDI	DMSEXT	DMSRNE									
FSPARSE	000012	DMSDSK	DMSTPE	DMSTPF	DMSTPG								
FSR	000004	DMSBOP	DMSTPE	DMSTPF	DMSTPG								
FSTADBC	000020	DMSXMA											
FSTAIC	000059	DMSEDI	DMSEDX	DMSGND	DMSLAD	DMSLBT	DMSSOP	DMSTYP	DMSXGT	DMSXIN	DMSXMA	DMSXPT	DMSXUP
FSTBLKCT	000002	DMSGND	DMSLAD										
FSTCMMD	000006	DMSTPE	DMSTPF	DMSTPG									
FSTD	000070	DMSEDI	DMSEDX	DMSEXC	DMSGND	DMSHLS	DMSLAD	DMSLBT	DMSNCP	DMSSOP	DMSTYP	DMSUTL	DMSXED
		DMSXGT	DMSXIN	DMSXMA	DMSXPT	DMSXSU	DMSXUP						

LABEL	COUNT	REFERENCES											
FSTDATEW	000003	DMSGND	DMSLAD										
FSTEPL	000045	DMSEXC	DMSEXT	DMSGLB	DMSLBT	DMSLDR	DMSLIO	DMSLLU	DMSOLD	DMSPRV	DMSPUN	DMSRNE	DMSSOP
		DMSTYP	DMSXFD	DMSXGT	DMSXIN	DMSXMA	DMSXPT	DMSXRE	DMSXUP				
FSTFINRD	000011	DMSCIT	DMSCRD	DMSEDX	DMSINT	DMSSVN							
FSTFLAGS	000006	DMSSOP	DMSUTL	DMSXPT									
FSTFMODE	000025	DMSEDX	DMSGND	DMSHLS	DMSLAD	DMSLBT	DMSNCP	DMSSOP	DMSTYP	DMSXED	DMSXIN	DMSXSU	DMSXUP
FSTFNAME	000006	DMSLAD	DMSXSU										
FSTFOP	000023	DMSGND	DMSLAD										
FSTLRECL	000018	DMSEXC	DMSGND	DMSLAD	DMSLBT	DMSTYP	DMSUTL	DMSXGT	DMSXIN	DMSXMA	DMSXPT	DMSXUP	
FSTNLVL	000021	DMSGND	DMSLAD										
FSTRECCT	000003	DMSGND	DMSLAD	DMSSOP									
FSTRECFM	000019	DMSEDI	DMSEDX	DMSGND	DMSLAD	DMSLBT	DMSSOP	DMSTYP	DMSUTL	DMSXGT	DMSXIN	DMSXMA	DMSXPT
		DMSXUP											
FSTRWDSK	000003	DMSSOP	DMSUTL	DMSXPT									
FSTSAVAD	000012	DMSTPE	DMSTPF	DMSTPG									
FSTSAVE	000020	DMSDSK	DMSTPE	DMSTPF	DMSTPG								
FSTXRDSK	000003	DMSSOP	DMSUTL										
FSTXTADR	000007	DMSLDR	DMSLOA	DMSLSB	DMSOLD								
FTHREE	000002	DMSHLP											
FTRDCONV	000003	DMSTPE	DMSTPF	DMSTPG									
FTRDLDNS	000012	DMSTPE	DMSTPF	DMSTPG									
FTRTRANS	000003	DMSTPE	DMSTPF	DMSTPG									
FTR7TRK	000003	DMSTPE	DMSTPF	DMSTPG									
FTWO	000001	DMSHLP											
FTYPE	000020	DMSEDI	DMSEDX	DMSLDR	DMSLGT	DMSLIB	DMSLST	DMSOLD	DMSPRV	DMSRRV	DMSSCR	DMSSRV	DMSTYP
		DMSXSE											
FV	000014	DMSEDI	DMSEDX	DMSSCR									
FVS	000001	DMSINS											
FVSDIOPL	000004	DMSAUD											
FVSDSKA	000002	DMSACM	DMSAUD										
FVSECT	000095	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSARN	DMSARX	DMSASM	DMSAUD	DMSBRD	DMSBTB	DMSBTP
		DMSBWR	DMSCIT	DMSCMP	DMSCRD	DMSCWR	DMSCWT	DMSDIO	DMSDOS	DMSDSK	DMSDSL	DMSERD	DMSERS
		DMSFNS	DMSGND	DMSINT	DMSITE	DMSITI	DMSITP	DMSITS	DMSLAD	DMSLBM	DMSLFS	DMSMOD	DMSPNT
		DMSPUN	DMSQRY	DMSRNM	DMSSLN	DMSSOP	DMSSTT	DMSTPE	DMSTPF	DMSTPG	DMSTQQ	DMSUPD	DMSXIN
		DMSXMA	DMSXUP										
FVSELMNL	000002	DMSAUD											
FVSELMNT	000003	DMSAUD											
FVSERAS0	000023	DMSERS	DMSRNM										
FVSERAS1	000012	DMSERS	DMSRNM										
FVSERAS2	000023	DMSERS	DMSRNM										
FVSERAS3	000007	DMSERS											
FVSERAS4	000017	DMSERS											
FVSERAS5	000005	DMSERS											
FVSFLG0	000009	DMSBRD	DMSDSK	DMSTPE	DMSTPF	DMSTPG							
FVSFSTAC	000005	DMSCMP	DMSGND	DMSMOD	DMSUPD								
FVSFSTAD	000019	DMSARN	DMSARX	DMSASM	DMSCMP	DMSDSK	DMSDSL	DMSGND	DMSLBM	DMSMOD	DMSPUN	DMSSOP	DMSSTT

LABEL	COUNT	REFERENCES								
		DMSUPD	DMSXIN	DMSXMA	DMSXUP					
FVSFSTDB	000002	DMSSTT								
FVSFSTDT	000005	DMSSTT								
FVSFSTFV	000001	DMSSTT								
FVSFSTHP	000002	DMSERS	DMSSTT							
FVSFSTIC	000002	DMSSTT								
FVSFSTM	000007	DMSDSK	DMSSTT	DMSTPE	DMSTPF	DMSTPG				
FVSFSTN	000005	DMSSTT								
FVSFSTRP	000001	DMSSTT								
FVSFSTWP	000001	DMSSTT								
FVSFSTYR	000002	DMSSTT								
FVSL1	000011	DMSALU	DMSDSK	DMSFNS	DMSTPE	DMSTPF	DMSTPG			
FVSPATCH	000006	DMSFNS	DMSINT							
FVSUFSTC	000009	DMSBRD	DMSDSK	DMSTPE	DMSTPF	DMSTPG				
FWADDR	000019	DMSACC	DMSACF	DMSACM	DMSAUD	DMSERD	DMSERS	DMSFNS	DMSMOD	
FXD	000023	DMSDSL	DMSSEB	DMSSOP	DMSSQS	DMSTMA	DMSTPD			
FZERO	000012	DMSHLP	DMSVIP							
F0	000028	DMSDBG	DMSINS	DMSITE	DMSITS	DMSXIN				
F1DSI	000001	DMSCVH								
F1DSN	000005	DMSCVH								
F1DWDS	000003	DMSCVH	DMSVLT							
F1END	000002	DMSCVH								
F1EXSEQ	000001	DMSCVH								
F1EXTYP	000001	DMSCVH								
F1FTYPE	000001	DMSCVH								
F1ID	000001	DMSCVH								
F1LEN	000003	DMSCVH								
F1LVOL	000001	DMSCVH								
F1START	000001	DMSCVH								
F2	000015	DMSITE								
F256	000008	DMSCWR	DMSHDI	DMSHDS						
F4	000016	DMSITE								
F4096	000003	DMSAMS	DMSDBD	DMSHLI						
F6	000033	DMSDBG	DMSITE	DMSITS	DMSSOP					
F65535	000020	DMSACF	DMSDSK	DMSMOD	DMSPNT	DMSSLN	DMSTPE	DMSTPF	DMSTPG	DMSTQQ
F800	000002	DMSACM	DMSAUD							
GETFLAG	000007	DMSEDI								
GETFST	000001	DMSEDI								
GET1	000002	DMSLSY								
GIOPLIST	000001	DMSSCR								
GOFNAM	000001	DMSHLS								
GOFTYP	000001	DMSHLS								
GPRLOG	000011	DMSDBG	DMSITS							
GPRSAV	000004	DMSLDR	DMSOLD							
GRAFDEV	000001	DMSINS								
HALF	000002	DMSEDI	DMSLDS							

LABEL	COUNT	REFERENCES
HEADER	000026	DMSDLK DMSDSK DMSLST DMSTPE DMSTPF DMSTPG DMSUTL
HEADERTR	000006	DMSTPE DMSTPF DMSTPG
HELPFST	000007	DMSHLS
HEX	000060	DMSCPY DMSDBG DMSDLK DMSDOS DMSDSV DMSEDI DMSFNS DMSOPL DMSPRT DMSSSK DMSTLB DMSTYP
HEXHEX	000010	DMSDBG
HIERR	000009	DMSTPE DMSTPF DMSTPG
HIPHAS	000006	DMSFCH DMSFET
HIPROG	000002	DMSFCH
HLPCTA	000001	DMSHLD
HLPSAVE	000014	DMSHLS
HLPSECT	000013	DMSHLB DMSHLE DMSHLI DMSHLP DMSHLS
HOLD	000013	DMSITI
HOLDFLAG	000015	DMSSCR
HONE	000001	DMSHLS
HTEN	000002	DMSHLB DMSHLP
HTWELVE	000002	DMSHLS
HTWENTY	000001	DMSHLS
HZERO	000002	DMSHLP
H4096	000001	DMSINS
IADT	000005	DMSACC DMSDAS DMSDSK DMSLAD
IC	000008	DMSDBG DMSXPO DMSXSC
IHADEB	000021	DMSFCH DMSLOS DMSMVE DMSSBS DMSSCT DMSSOP DMSSQS DMSSVT
IHADECB	000006	DMSSBD DMSSBS DMSSCT DMSSEB DMSSVT
IHAJFCB	000001	DMSSVT
IJBBOX	000002	DMSETR DMSSTG
IJBCCWT	000001	DMSDOS
IJBFLG04	000001	DMSBOP
IJBFTTAB	000004	DMSDOS DMSFET
IJJHCPL	000002	DMSCVH
IJJHDLST	000009	DMSCVH
IJJHFMT1	000012	DMSCVH
IKQACB	000007	DMSBOP DMSCLS DMSVIP
IKQEXLST	000003	DMSVIP
IKQRPL	000006	DMSVIP
INBUFF	000030	DMSLBM DMSRNE DMSTPE DMSTPF DMSTPG
INCOMM	000009	DMSTPE DMSTPF DMSTPG
INCRNO	000006	DMSEDI DMSXCT
INDEXS	000004	DMSHLP DMSTPE DMSTPF DMSTPG
INDL	000008	DMSHLP
INFILE	000012	DMSTPE DMSTPF DMSTPG
INFV	000010	DMSLBM DMSTPE DMSTPF DMSTPG
INITNO	000021	DMSLBM DMSTPE DMSTPF DMSTPG
INMODE	000023	DMSEDI DMSLBM DMSSCR DMSTPE DMSTPF DMSTPG
INNAME	000012	DMSLBM DMSTPE DMSTPF DMSTPG
INNOIT	000014	DMSLBM DMSTPE DMSTPF DMSTPG
INNORD	000006	DMSTPE DMSTPF DMSTPG

LABEL	COUNT	REFERENCES											
INPUT	000086	DMSARN	DMSBOP	DMSCPY	DMSDBD	DMSDBG	DMSDSL	DMSDSV	DMSEDI	DMSFCH	DMSGRN	DMSITE	DMSMVE
		DMSNCP	DMSOR1	DMSPRV	DMSQRY	DMSRRV	DMSSRV	DMSTPE	DMSTPF	DMSTPG	DMSUTL	DMSXCP	DMSXDS
		DMSXMD											
INPUTSIZ	000002	DMSDBG											
INPUT1	000008	DMSDBG											
INRPTR	000004	DMSDSK	DMSTPE	DMSTPF	DMSTPG								
INSIZE	000012	DMSLBM	DMSSRT	DMSTPE	DMSTPF	DMSTPG							
INSTALID	000005	DMSINI	DMSPRT										
INTINFO	000006	DMSDOS	DMSITP										
INTYPE	000013	DMSDSK	DMSLBM	DMSTPE	DMSTPF	DMSTPG							
INVLD	000003	DMSEDI											
INVLDHDR	000001	DMSEDX											
INWPTR	000004	DMSDSK	DMSTPE	DMSTPF	DMSTPG								
IOAD	000002	DMSEDX											
IOAREA	000002	DMSRDC	DMSTYP										
IOBBCSW	000003	DMSSBS	DMSSEB										
IOBBECBC	000002	DMSSEB											
IOBBECBP	000003	DMSSBS	DMSSEB										
IOBBFLG	000002	DMSSBS	DMSSCT										
IOBCSW	000007	DMSARN	DMSARX	DMSASM	DMSSBS	DMSSCT	DMSUTL						
IOBDCBPT	000008	DMSSOP	DMSTLB										
IOBECB	000005	DMSIFC	DMSSQS										
IOBECBPT	000003	DMSSQS											
IOBEND	000001	DMSSOP											
IOBFLG	000001	DMSSOP											
IOBIN	000034	DMSARN	DMSARX	DMSASM	DMSSBD	DMSSBS	DMSSEB	DMSSOP	DMSSQS	DMSSVU			
IOBIOFLG	000046	DMSARN	DMSARX	DMSASM	DMSSBD	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVU		
IOBNXTAD	000003	DMSSOP											
IOBOUT	000007	DMSSBS	DMSSCT	DMSSQS									
IOBSTART	000008	DMSSOP	DMSSQS										
IOBUFF	000024	DMSTPE	DMSTPF	DMSTPG									
IOBUPD	000004	DMSSQS											
IOCOMM	000007	DMSDIO											
IOID	000005	DMSEDI	DMSEDX										
IOLIST	000043	DMSEDI	DMSEDX										
IOMODE	000003	DMSEDI	DMSEDX										
IONPSW	000012	DMSINI	DMSINS	DMSIOW	DMSITE	DMSXPO	DMSXSC						
IONTABL	000012	DMSABN	DMSHDI	DMSINT	DMSITI								
IOOLD	000002	DMSDIO	DMSITI										
IOOPSW	000029	DMSCIT	DMSDBG	DMSDIO	DMSINI	DMSIOW	DMSITE	DMSITI	DMSXPO	DMSXSC			
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								

LABEL	COUNT	REFERENCES				
ITEM	000071	DMSBRD	DMSEDI	DMSEDX	DMSSCR	DMSUPD
ITSBIT	000013	DMSITS				
JAR	000003	DMSEDI	DMSEDX			
JCSW2	000001	DMSDOS				
JCSW3	000016	DMSOPT	DMSSET			
JCSW4	000005	DMSDOS	DMSOPT	DMSSET		
JFCBIND2	000007	DMSFLD	DMSSEB	DMSSOP	DMSTLB	
JFCBMASK	000027	DMSSOP	DMSSVT	DMSTLB		
JFCBUFNO	000001	DMSFLD				
JFCDSORG	000002	DMSSOP				
JFCKEYLE	000003	DMSFLD	DMSSOP			
JFCLIMCT	000003	DMSFLD	DMSSOP			
JFCLRECL	000001	DMSSVT				
JFCOPTCD	000009	DMSFLD	DMSSOP			
JFIRST	000009	DMSHDS				
JFLAGS	000014	DMSDBG				
JLAST	000010	DMSHDS				
JNUMB	000012	DMSHDS	DMSINT			
JOBDATE	000004	DMSDLK	DMSDOS	DMSSET		
JR1	000008	DMSITE				
JSRO	000004	DMSACM				
JSYM	000002	DMSLSY				
KABUFFSZ	000006	DMSTPE	DMSTPF	DMSTPG		
KAEJECTR	000015	DMSTPE	DMSTPF	DMSTPG		
KALEND	000012	DMSTPE	DMSTPF	DMSTPG		
KAOUTPRT	000003	DMSTPE	DMSTPF	DMSTPG		
KCFID	000003	DMSTPE	DMSTPF	DMSTPG		
KCFROM	000003	DMSTPE	DMSTPF	DMSTPG		
KCFTYPE	000003	DMSTPE	DMSTPF	DMSTPG		
KCFUNC	000006	DMSTPE	DMSTPF	DMSTPG		
KCLEAR	000002	DMSXPO	DMSXSC			
KCOPTION	000003	DMSTPE	DMSTPF	DMSTPG		
KCPARAM	000003	DMSTPE	DMSTPF	DMSTPG		
KCTO	000003	DMSTPE	DMSTPF	DMSTPG		
KC1600B	000003	DMSTPE	DMSTPF	DMSTPG		
KC4096B	000003	DMSTPE	DMSTPF	DMSTPG		
KC800B	000006	DMSTPE	DMSTPF	DMSTPG		
KEEPDEN7	000006	DMSTPE	DMSTPF	DMSTPG		
KEEPTRK7	000006	DMSTPE	DMSTPF	DMSTPG		
KENTER	000002	DMSXSC				
KEYCHANG	000005	DMSSBD	DMSSVU			
KEYCHNG	000007	DMSSBD	DMSSVU			
KEYCOUT	000004	DMSSBD	DMSSVU			
KEYEOF	000001	DMSSVU				
KEYEXTPL	000004	DMSSBD	DMSSVU			
KEYFORM	000002	DMSSVU				

LABEL	COUNT	REFERENCES
KEYLNGTH	000010	DMSSBD DMSSVU
KEYMARK	000002	DMSSVU
KEYMAX	000002	DMSITS
KEYNAME	000007	DMSSBD DMSSVU
KEYOP	000009	DMSSBD DMSSVU
KEYP	000008	DMSITS
KEYPTR1	000003	DMSSVU
KEYPTR2	000002	DMSSVU
KEYS	000003	DMSBTP DMSITS
KEYSECT	000002	DMSSBD DMSSVU
KEYTABLE	000005	DMSSVU
KEYTBLAD	000009	DMSSBD DMSSVU
KEYTBLNO	000016	DMSSBD DMSSVU
KEYTYPE	000002	DMSSVU
KEYXTNT1	000003	DMSSVU
KEYXTNT2	000002	DMSSVU
KF1	000031	DMSERD DMSSBD DMSTPE DMSTPF DMSTPG
KF4096	000021	DMSTPE DMSTPF DMSTPG
KF800	000015	DMSTPE DMSTPF DMSTPG
KH12	000010	DMSDSK DMSTPE DMSTPF DMSTPG
KH2	000011	DMSERD DMSEXT DMSTPE DMSTPF DMSTPG
KH3	000003	DMSTPE DMSTPF DMSTPG
KH5	000006	DMSTPE DMSTPF DMSTPG
KH8	000009	DMSTPE DMSTPF DMSTPG
KLGTPEN	000002	DMSXPO DMSXSC
KNOACT	000003	DMSXSC
KPA1	000002	DMSXPO DMSXSC
KPA2	000004	DMSXPO DMSXSC
KPA3	000002	DMSXPO DMSXSC
KXFLAG	000026	DMSABN DMSACC DMSAUD DMSBWR DMSCIT DMSCRD DMSCWR DMSCWT DMSDIO DMSDSK DMSERD DMSERS
		DMSFNS DMSITI DMSITS DMSRNM DMSTPE DMSTPF DMSTPG
KXWANT	000018	DMSABN DMSACC DMSAUD DMSBWR DMSCIT DMSDIO DMSDSK DMSERD DMSERS DMSFNS DMSITI DMSITS
		DMSRNM DMSTPE DMSTPF DMSTPG
KXWSVC	000006	DMSCRD DMSCWR DMSCWT DMSITS
LABBUFSP	000002	DMSLAB
LABCONV	000001	DMSLAB
LABCRD	000004	DMSLBD DMSQRY DMSTLB
LABDCRD	000002	DMSTLB
LABDEXD	000004	DMSFLD DMSLBD DMSTLB
LABDFID	000002	DMSTLB
LABDFSEQ	000003	DMSTLB
LABDGENN	000002	DMSTLB
LABDGENV	000002	DMSTLB
LABDSEC	000002	DMSTLB
LABDSN	000002	DMSLAB
LABDVID	000007	DMSFLD DMSQRY DMSTLB

LABEL	COUNT	REFERENCES				
LABDVSEQ	000002	DMSTLB				
LABED	000001	DMSLAB				
LABEXD	000004	DMSLBD	DMSQRY	DMSTLB		
LABEXN	000003	DMSLAB				
LABEXT	000002	DMSLAB				
LABFCEPT	000005	DMSFLD	DMSLBD			
LABFDEF	000006	DMSFLD	DMSLBD	DMSQRY		
LABFID	000008	DMSLAB	DMSLBD	DMSQRY	DMSTLB	
LABFILE	000008	DMSFLD	DMSLBD	DMSQRY	DMSTLB	
LABFIRST	000010	DMSABN	DMSFLD	DMSLBD	DMSQRY	DMSTLB
LABFLAG1	000028	DMSFLD	DMSLBD	DMSQRY	DMSTLB	
LABFLAG2	000010	DMSFLD	DMSLBD	DMSQRY	DMSTLB	
LABFNAME	000001	DMSLAB				
LABFSEQ	000005	DMSLBD	DMSQRY	DMSTLB		
LABFSER	000001	DMSLAB				
LABGENN	000004	DMSLBD	DMSQRY	DMSTLB		
LABGENV	000004	DMSLBD	DMSQRY	DMSTLB		
LABIND	000004	DMSLAB				
LABLAST	000002	DMSLAB				
LABLEN	000004	DMSDLK				
LABLUBA	000002	DMSLAB				
LABNEXT	000021	DMSFLD	DMSLBD	DMSQRY	DMSTLB	
LABNUM	000017	DMSABN	DMSFLD	DMSLBD	DMSQRY	DMSTLB
LABOMIT	000001	DMSLAB				
LABOPCOD	000001	DMSLAB				
LABPERM	000002	DMSLBD				
LABREC	000001	DMSLAB				
LABSEC	000004	DMSLBD	DMSQRY	DMSTLB		
LABSECT	000032	DMSFLD	DMSLBD	DMSQRY	DMSTLB	
LABSEQ	000001	DMSLAB				
LABSIZE	000010	DMSFLD	DMSLBD			
LABST	000002	DMSLAB				
LABSTBK	000002	DMSLAB				
LABSW	000002	DMSLAB				
LABTYP	000002	DMSLAB				
LABUCNAM	000002	DMSLAB				
LABVOL	000004	DMSLAB				
LABVOLID	000010	DMSFLD	DMSLBD	DMSQRY	DMSTLB	
LABVSEQ	000006	DMSLAB	DMSLBD	DMSQRY	DMSTLB	
LAB64K	000002	DMSLAB				
LASTCMND	000016	DMSEXT	DMSINT	DMSXCM	DMSXSS	
LASTCYL	000003	DMSDIO				
LASTDMP	000001	DMSDBG				
LASTEXEC	000002	DMSEXT				
LASTHED	000003	DMSDIO				
LASTLINE	000013	DMSDBD				

LABEL	COUNT	REFERENCES						
LASTLMOD	000004	DMSHLL	DMSMOD	DMSSLN	DMSXSG			
LASTLOC	000002	DMSFET	DMSXIN					
LASTREC	000024	DMSBWR	DMSDIO					
LASTTMOD	000008	DMSITS	DMSLSB	DMSMOD	DMSSLN			
LASTUSED	000012	DMSABN	DMSLCK					
LDMSROS	000004	DMSABN	DMSACM	DMSALU				
LDRADDR	000014	DMSLDR	DMSLIO	DMSLOA	DMSOLD			
LDRFLAGS	000020	DMSLDR	DMSLOA	DMSMOD	DMSOLD	DMSSLN		
LDRRTCD	000003	DMSLDR	DMSOLD					
LDRST	000009	DMSLDR	DMSLGT	DMSLIB	DMSLIO	DMSLSB	DMSOLD	
LEFTOVER	000018	DMSTPE	DMSTPF	DMSTPG				
LENOVS	000003	DMSITS	DMSOVR					
LFILEID	000002	DMSDLB	DMSQRY					
LHEADER	000009	DMSTPE	DMSTPF	DMSTPG				
LHH	000001	DMSLAB						
LIBDIR	000007	DMSLBM	DMSPRT					
LIBDIRSZ	000006	DMSLBM	DMSPRT					
LIBDIRX	000005	DMSPRT						
LIBIDENT	000012	DMSLBM	DMSPRT					
LIBSECT	000007	DMSLBM	DMSPRT					
LINE	000126	DMSBTP	DMSDBD	DMSDBG	DMSEDI	DMSEDX	DMSEXE	DMSITE
LINECNT	000013	DMSARN	DMSDLK	DMSHLE	DMSHLP			
LINECT	000011	DMSQRY	DMSSET	DMSTPE	DMSTPF	DMSTPG		
LINELOC	000002	DMSEDX	DMSSCR					
LINENO	000002	DMSEDI						
LINE1	000002	DMSDBD	DMSLIO					
LINE1A	000001	DMSDBD						
LINE1B	000001	DMSDBD						
LINE1C	000001	DMSDBD						
LINKBACK	000012	DMSHLI	DMSHLP					
LINKBEG	000002	DMSHLI						
LINKCHAR	000015	DMSHLP						
LINKDOWN	000017	DMSHLI	DMSHLP					
LINKELEM	000016	DMSHLI	DMSHLP					
LINKFOR	000027	DMSHLI	DMSHLP					
LINKLAST	000007	DMSSAB	DMSSLN	DMSSTG				
LINKMULT	000010	DMSHLP						
LINKNEXT	000001	DMSHLI						
LINKPARM	000006	DMSBTP	DMSHLP					
LINKSIZE	000013	DMSHLI	DMSHLP					
LINKSTAR	000015	DMSHLI	DMSHLP					
LINKSTRT	000009	DMSSLN	DMSSTG	DMSSVT				
LINSEQNO	000003	DMSHLS						
LIOCSCOM	000005	DMSBOP	DMSCLS	DMSVLT				
LLZ	000005	DMSHLP	DMSHLS					
LMCURR	000005	DMSEDI						

LABEL	COUNT	REFERENCES									
LMINCR	000005	DMSEDI									
LMSG	000021	DMSQRY	DMSSET	DMSTMA	DMSTPE	DMSTPF	DMSTPG				
LMSTART	000010	DMSEDI	DMSEDX								
LOAD	000015	DMSBOP	DMSBTB	DMSDOS	DMSFNC	DMSLOS	DMSSPR				
LOADBASE	000008	DMSDLK	DMSTPE	DMSTPF	DMSTPG						
LOADING	000001	DMSTMA									
LOADIT	000010	DMSAMS	DMSBOP	DMSDOS	DMSEDX	DMSHLL	DMSSET	DMSTPE	DMSTPF	DMSTPG	DMSVIB
LOADMACH	000003	DMSTPE	DMSTPF	DMSTPG							
LOADMNME	000006	DMSTPE	DMSTPF	DMSTPG							
LOADMOD	000013	DMSEDX	DMSFNC	DMSHLL	DMSINA	DMSTPE	DMSTPF	DMSTPG			
LOADMPL	000003	DMSTPE	DMSTPF	DMSTPG							
LOADPROC	000009	DMSTPE	DMSTPF	DMSTPG							
LOADSTRT	000004	DMSSET									
LOADSWT	000039	DMSTPE	DMSTPF	DMSTPG							
LOADWR	000003	DMSTPE	DMSTPF	DMSTPG							
LOC	000273	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSAUD	DMSBOP	DMSBWR	DMSCIT	DMSCLS
		DMSCMP	DMSCRD	DMSCVH	DMSDAS	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSEDX	DMSERD	DMSERS	DMSEXC
		DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFET	DMSFLD	DMSFNC	DMSFNS	DMSFOR	DMSFRE	DMSGIO	DMSGLB
		DMSHDI	DMSHDS	DMSHLI	DMSHLS	DMSIFC	DMSINS	DMSINT	DMSITE	DMSITP	DMSITS	DMSLAD	DMSLAF
		DMSLBD	DMSLCK	DMSLDR	DMSLDS	DMSLGT	DMSLIB	DMSLSB	DMSMOD	DMSOLD	DMSOPL	DMSOR1	DMSOVR
		DMSPRE	DMSPRT	DMSPUN	DMSQRY	DMSRNE	DMSROS	DMSSAB	DMSSEB	DMSSET	DMSSFF	DMSSLN	DMSSOP
		DMSSPR	DMSSQS	DMSSTG	DMSSVN	DMSSVT	DMSSVU	DMSSYN	DMSTLB	DMSXFC	DMSZAP		
LOCCNT	000040	DMSACM	DMSBTB	DMSEDX	DMSFET	DMSFRE	DMSINS	DMSINT	DMSLDR	DMSLOA	DMSMOD	DMSOLD	DMSSET
		DMSSLN	DMSSMN	DMSSTG	DMSXSG								
LOCCT	000025	DMSLDR	DMSLSB	DMSOLD									
LOCKTAB	000005	DMSABN	DMSLCK										
LOOP	000043	DMSBOP	DMSCPY	DMSDIO	DMSEXE	DMSEXT	DMSFLD	DMSGLB	DMSITE	DMSLDS	DMSLKD	DMSLSB	DMSLST
		DMSMDP	DMSMVE	DMSOVR	DMSRNE	DMSSLN	DMSSRT	DMSSSK	DMSTIO	DMSTPE	DMSTPF	DMSTPG	DMSXMS
LOOPEY	000003	DMSTPE	DMSTPF	DMSTPG									
LOWSAVE	000007	DMSDBG	DMSSVT										
LSCARWLG	000029	DMSXCT	DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXSU						
LSCFALTF	000011	DMSXSC	DMSXSD	DMSXSS									
LSCFAROW	000008	DMSXSD	DMSXSS										
LSCFCHGD	000027	DMSXCT	DMSXPX	DMSXSD	DMSXSS								
LSCFEOF	000008	DMSXFC	DMSXML	DMSXPX	DMSXSD								
LSCFHIGH	000007	DMSXCT	DMSXSD										
LSCFMASK	000004	DMSXSD											
LSCFMDT	000008	DMSXSD	DMSXSS										
LSCFNPRF	000014	DMSXPO	DMSXSD	DMSXSS									
LSCFNULL	000008	DMSXSD	DMSXSS										
LSCFPROT	000034	DMSXFC	DMSXML	DMSXPO	DMSXSC	DMSXSD	DMSXSS						
LSCFRST	000010	DMSXSD	DMSXSS										
LSCFRSVD	000018	DMSXCT	DMSXPX	DMSXSD	DMSXSE								
LSCFTABS	000004	DMSXSD											
LSCFTOP	000007	DMSXFC	DMSXPX	DMSXSC	DMSXSD								
LSCFWPTR	000027	DMSXBG	DMSXCT	DMSXED	DMSXPX	DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXSU			

LABEL	COUNT	REFERENCES
LSCFWRAP	000019	DMSXPX DMSXSC DMSXSD DMSXSS
LSCLLSCB	000002	DMSXSD
LSCPFXLG	000051	DMSXFC DMSXPX DMSXSC DMSXSD DMSXSS
LSCRAINP	000011	DMSXMD DMSXSC DMSXSD
LSCRBYTE	000003	DMSXBG DMSXCT DMSXSD
LSCRCTL	000005	DMSXSC DMSXSD
LSCRCTLP	000002	DMSXSD
LSCRCURL	000011	DMSXCT DMSXMD DMSXSD DMSXSE DMSXSU
LSCRDSPH	000015	DMSXCT DMSXSC DMSXSD DMSXSE DMSXSS
LSCRDSPV	000007	DMSXCT DMSXSC DMSXSD DMSXSE
LSCREEN	000036	DMSXBG DMSXCT DMSXED DMSXFC DMSXIO DMSXMA DMSXMC DMSXMD DMSXML DMSXPO DMSXPX DMSXSC
		DMSXSD DMSXSE DMSXSS DMSXSU
LSCRFADD	000011	DMSXCT DMSXPX DMSXSD DMSXSS
LSCRFLG1	000073	DMSXCT DMSXFC DMSXML DMSXPX DMSXSC DMSXSD DMSXSS
LSCRFLG2	000076	DMSXCT DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS
LSCRFLNE	000038	DMSXCT DMSXFC DMSXML DMSXPX DMSXSC DMSXSD DMSXSS
LSCRIBUF	000011	DMSXSC DMSXSS
LSCRIMAD	000030	DMSXCT DMSXPX DMSXSC DMSXSD DMSXSS
LSCRINPU	000047	DMSXCT DMSXFC DMSXMD DMSXML DMSXSC DMSXSD DMSXSE DMSXSS DMSXSU
LSCRLGTH	000023	DMSXCT DMSXFC DMSXSC DMSXSD DMSXSS
LSCRLINE	000054	DMSXCT DMSXFC DMSXML DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS
LSCRLTBL	000021	DMSXCT DMSXFC DMSXML DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS
LSCRMASK	000008	DMSXPX DMSXSD DMSXSE DMSXSU
LSCRMSG	000014	DMSXBG DMSXIO DMSXMA DMSXPO DMSXSC DMSXSD
LSCRNBLN	000006	DMSXML DMSXSD
LSCRSIZE	000030	DMSXCT DMSXFC DMSXMC DMSXMD DMSXML DMSXSC DMSXSD DMSXSE DMSXSS DMSXSU
LSCRSTAT	000003	DMSXSD
LSCRTABS	000008	DMSXPX DMSXSD DMSXSE DMSXSU
LSCRULCO	000002	DMSXSD
LSCRWIDT	000045	DMSXCT DMSXFC DMSXIO DMSXMA DMSXMC DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS
LSCSTALG	000006	DMSXSC DMSXSD
LSCZDEPT	000032	DMSXBG DMSXCT DMSXED DMSXPX DMSXSC DMSXSD DMSXSS DMSXSU
LSTFINRD	000005	DMSCIT DMSCRD DMSSVN
LTK	000009	DMSAMS DMSDOS DMSITP DMSSET
LUB	000005	DMSDLK DMSDSV
LUBCLB	000004	DMSDSV
LUBP	000009	DMSDSV
LUBPR	000003	DMSDLK DMSDSV
LUBPT	000025	DMSAMS DMSBOP DMSCLS DMSCVH DMSDAS DMSDLB DMSDOS DMSETR DMSFCH DMSLLU DMSOPL DMSPRV
		DMSRRV DMSSET DMSSRV DMSXCP
LUBRES	000005	DMSDLK DMSDSV
LUBRLB	000005	DMSDLK DMSDSV
LUBSLB	000003	DMSDSV
LUNDEF	000012	DMSLDR DMSOLD
L18	000009	DMSTPE DMSTPF DMSTPG DMSXCT DMSXED DMSXIN DMSXPT
MACDIRC	000011	DMSABN DMSSCT DMSSOP DMSSTG DMSSVT

LABEL	COUNT	REFERENCES											
MACLBSV	000004	DMSGLB											
MACLIBL	000009	DMSGLB	DMSQRY	DMSSCT	DMSSOP	DMSSTG	DMSSVT						
MACRO	000006	DMSEDI	DMSMVG										
MAINAD	000003	DMSEDX											
MAINHIGH	000043	DMSARX	DMSASM	DMSDOS	DMSERD	DMSFCH	DMSFRE	DMSIMA	DMSINS	DMSLDR	DMSLOA	DMSLSB	DMSSET
		DMSSMN	DMSSTG										
MAINLIST	000012	DMSDOS	DMSFCH	DMSSMN	DMSSTG								
MAINSTRT	000008	DMSDOS	DMSFCH	DMSSMN	DMSSTG								
MATCH	000043	DMSASN	DMSDLB	DMSFLD	DMSHLP	DMSLST	DMSTPE	DMSTPF	DMSTPG				
MATCHALL	000003	DMSTPE	DMSTPF	DMSTPG									
MATCHFM	000009	DMSTPE	DMSTPF	DMSTPG									
MATCHFN	000012	DMSTPE	DMSTPF	DMSTPG									
MATCHFT	000012	DMSTPE	DMSTPF	DMSTPG									
MAX	000013	DMSASM	DMSFRE										
MAXADDR	000005	DMSDSV	DMSHLB	DMSHLP									
MAXBUFLG	000113	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXIO
		DMSXMA	DMSXMD	DMSXPT	DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXSU	DMSXUP			
MAXCODE	000001	DMSFRE											
MCKM	000014	DMSINI	DMSINS	DMSITS									
MCKNPSW	000001	DMSINI											
MCKOPSW	000002	DMSFNS											
MDPCALL	000005	DMSABN	DMSMDP	DMSMOD									
MEMBOUND	000008	DMSLDR	DMSOLD										
MENU	000002	DMSHLI											
MESSAGE	000069	DMSTPE	DMSTPF	DMSTPG									
MISFLAGS	000044	DMSABN	DMSACC	DMSAMS	DMSARN	DMSARX	DMSASM	DMSCAT	DMSCIT	DMSCPY	DMSCRD	DMSEDI	DMSEXC
		DMSINS	DMSINT	DMSITI	DMSITS	DMSLBM	DMSLBT	DMSLKD	DMSQRY	DMSSET	DMSSRT	DMSSTG	DMSUPD
		DMSXBG											
MODEL3	000013	DMSBOP	DMSTPE	DMSTPF	DMSTPG								
MODEL5	000010	DMSBOP	DMSTPE	DMSTPF	DMSTPG								
MODEL7	000010	DMSBOP	DMSTPE	DMSTPF	DMSTPG								
MODEL8	000004	DMSBOP	DMSTPE	DMSTPF	DMSTPG								
MODESET	000004	DMSACC	DMSLBT										
MODESETB	000126	DMSTPE	DMSTPF	DMSTPG									
MODESWT	000003	DMSTPE	DMSTPF	DMSTPG									
MODFLGS	000028	DMSABN	DMSACM	DMSINS	DMSLDR	DMSLSB	DMSMDP	DMSMOD	DMSOLD	DMSSET			
MODGNALL	000002	DMSMOD											
MODGNDOS	000003	DMSMOD											
MODLIST	000002	DMSSLN											
MODNAME	000018	DMSIFC	DMSIMA	DMSLKD	DMSTPE	DMSTPF	DMSTPG						
MONREGSV	000001	DMSBOP											
MSG	000041	DMSACC	DMSBTP	DMSDAS	DMSHLS	DMSMOD	DMSSCR	DMSSLN	DMSXBG	DMSXCT	DMSXDC	DMSXED	DMSXFD
		DMSXIN	DMSXMA	DMSXMS	DMSXPX	DMSXRE	DMSXSD	DMSXSE	DMSXST	DMSXTB			
MSGADDR	000005	DMSTPE	DMSTPF	DMSTPG	DMSXER								
MSGFLAGS	000042	DMSCIT	DMSCRD	DMSCWR	DMSEDI	DMSEXT	DMSHLL	DMSINS	DMSINT	DMSQRY	DMSSET	DMSTYP	DMSXIN
		DMSXMA	DMSXSC	DMSXSG	DMSXSU								

LABEL	COUNT	REFERENCES								
MSGNTAB	000001	DMSHLE								
MSG701	000003	DMSTPE	DMSTPF	DMSTPG						
MULTBLK	000012	DMSTPE	DMSTPF	DMSTPG						
MULTPNTR	000003	DMSHLP								
MVCFILID	000006	DMSTPE	DMSTPF	DMSTPG						
MVCLFRNT	000003	DMSTPE	DMSTPF	DMSTPG						
MVCNT	000001	DMSDBG								
MVCNT1	000003	DMSDBD	DMSDBG	DMSITE						
MVCNT2	000001	DMSDBG								
NEED	000007	DMSEXT	DMSLDR	DMSOLD						
NEGITS	000013	DMSCAT	DMSEXC	DMSINT	DMSITS	DMSQRY	DMSSET			
NEWBLKS	000005	DMSSVT								
NEWBUF	000001	DMSHLI								
NEWMODE	000009	DMSEDI	DMSRNM							
NEWNAME	000021	DMSEDI	DMSRNM	DMSUPD	DMSUTL					
NEWTYPE	000005	DMSEDI	DMSRNM							
NEXTO	000001	DMSITI								
NFSWS	000005	DMSHLP								
NICLPT	000012	DMSBOP	DMSCLS	DMSCVH	DMSDAS	DMSDLB	DMSDOS	DMSETR	DMSLLU	DMSXCP
NINEOFF	000003	DMSTPE	DMSTPF	DMSTPG						
NINETK	000024	DMSTPE	DMSTPF	DMSTPG						
NOABBREV	000006	DMSINA	DMSINT	DMSQRY	DMSSET					
NOAUTO	000008	DMSDLK	DMSLDR	DMSLIB	DMSLOA	DMSLSB	DMSOLD			
NOCHARS	000003	DMSHLB	DMSSPR							
NOCOPY	000006	DMSTPE	DMSTPF	DMSTPG						
NODISK	000043	DMSACC	DMSDLK	DMSEXT	DMSFCH	DMSLBT	DMSTPE	DMSTPF	DMSTPG	
NODUP	000007	DMSLDR	DMSLSB	DMSOLD						
NOEOFN	000006	DMSTPE	DMSTPF	DMSTPG						
NOEOT	000009	DMSTPE	DMSTPF	DMSTPG						
NOERASE	000009	DMSARN	DMSARX	DMSASM	DMSLIO	DMSLOA	DMSMOD	DMSUPD	DMSUTL	
NOIMPCP	000008	DMSINT	DMSQRY	DMSSET	DMSXCM					
NOIMPEX	000005	DMSINT	DMSQRY	DMSSET	DMSXCM					
NOINV	000005	DMSLDR	DMSLOA	DMSLSB	DMSOLD					
NOLIBE	000009	DMSLBT	DMSLDR	DMSLIB	DMSLOA	DMSLSB	DMSOLD			
NOLOAD1	000004	DMSSVT	DMSTPE	DMSTPF	DMSTPG					
NOLOAD2	000009	DMSTPE	DMSTPF	DMSTPG						
NOMAP	000007	DMSDLK	DMSLIO	DMSLOA	DMSLSB					
NOMAPFLG	000003	DMSMOD								
NOP	000014	DMSINI								
NOPAGREL	000009	DMSABN	DMSDOS	DMSFRE	DMSINT	DMSQRY	DMSSET	DMSSMN	DMSSTG	
NOPRINT	000049	DMSARN	DMSARX	DMSASM	DMSLKD	DMSLLU	DMSTPE	DMSTPF	DMSTPG	
NORDYMSG	000002	DMSSET								
NORDYTIM	000006	DMSINT	DMSQRY	DMSSET						
NOREP	000006	DMSLDR	DMSLOA	DMSLSB	DMSOLD	DMSUPD				
NOSLCADR	000006	DMSLDR	DMSOLD							
NOSPARSE	000012	DMSTPE	DMSTPF	DMSTPG						

LABEL	COUNT	REFERENCES
NOSTDSYN	000005	DMSINA DMSQRY DMSSYN
NOTACTV	000003	DMSTPE DMSTPF DMSTPG
NOTEOT	000003	DMSTPE DMSTPF DMSTPG
NOTERM	000033	DMSARX DMSASM DMSLKD DMSTPE DMSTPF DMSTPG DMSUPD
NOTEXT	000009	DMSDOS DMSFCH DMSFET
NOTFOUND	000047	DMSARX DMSASM DMSEDI DMSLAB DMSLBR DMSLBT DMSLFS DMSOR1 DMSOR3 DMSQRY DMSROS DMSSBD
		DMSSLN DMSSVT DMSTPE DMSTPF DMSTPG
NOTUSED	000006	DMSTPE DMSTPF DMSTPG
NOTYPING	000022	DMSCIT DMSCRD DMSCWR DMSEDI DMSEXT DMSHLL DMSINT DMSSET DMSTYP DMSXIN DMSXMA DMSXSC
		DMSXSG DMSXSU
NOVMREAD	000003	DMSINS DMSINT DMSSET
NOWORK	000010	DMSOVS DMSTPE DMSTPF DMSTPG
NOWRITE	000003	DMSTPE DMSTPF DMSTPG
NOWTM	000009	DMSTPE DMSTPF DMSTPG
NRMRET	000015	DMSABN DMSITS DMSTLB
NUCAFCHS	000014	DMSABN DMSLOS DMSSTG
NUCAPIO	000002	DMSERR DMSINS
NUCCBLKS	000010	DMSABN DMSLOS DMSSLN DMSSTG
NUCCODE	000004	DMSFRE
NUCCOPYR	000001	DMSINS
NUCFSTLN	000012	DMSCAT DMSCIT DMSCRD
NUCGLOBL	000003	DMSLOS DMSOSR
NUCKEY	000002	DMSFRE DMSSET
NUCLDIRC	000006	DMSSCT DMSSOP DMSSVT
NUCLDLIB	000009	DMSGLB DMSLOS DMSQRY DMSSCT DMSSOP DMSSVT
NUCLODSV	000004	DMSGLB
NUCLSTLN	000004	DMSCAT DMSCRD
NUCNBSTK	000007	DMSCAT DMSCIT DMSCRD
NUCNLSTK	000008	DMSCAT DMSCIT DMSCRD
NUCON	000637	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 DMSCVH DMSCWR DMSCWT DMSDAS DMSDBD DMSDBG DMSDIO DMSDLB DMSDLK DMSDMP
		DMSDOS DMSDSK DMSDSL DMSDSV DMSEDI DMSEDX DMSERD DMSERR DMSERS DMSETR DMSEXC DMSEXE
		DMSEXI DMSEXT DMSFCH DMSFET DMSFLD DMSFNS DMSFOR DMSFRE DMSGIO DMSGLB DMSGND DMSHDI
		DMSHDS DMSHLI DMSHLL DMSHLS DMSIFC DMSIMA DMSINA DMSINI DMSINM DMSINS DMSINT DMSIOW
		DMSITE DMSITI DMSITP DMSITS DMSLAB DMSLAD DMSLAF DMSLBD DMSLBM DMSLBR DMSLBT DMSLCK
		DMSLDP DMSLDS DMSLFS DMSLGT DMSLIB DMSLIO DMSLKD DMSLLU DMSLOA DMSLOS DMSLSB DMSLST
		DMSLSY DMSMDP DMSMOD DMSMVE DMSMVG DMSNCP DMSOLD DMSOPL DMSOPT DMSOR1 DMSOSR DMSOVR
		DMSOVS DMSPIO DMSPNT DMSPRE DMSPRT DMSPRV DMSPUN DMSQRY DMSRDC DMSRNE DMSRNM DMSROS
		DMSRRV DMSSAB DMSSBS DMSSCN DMSSCT DMSSEB DMSSET DMSSFF DMSSLN DMSSMN DMSSOP DMSSPR
		DMSSQS DMSSRT DMSSRV DMSSSK DMSSTG DMSSTT DMSSVN DMSSVT DMSSVU DMSSYN DMSTIO DMSTLA
		DMSTLB DMSTMA DMSTPD DMSTPE DMSTPF DMSTPG DMSTQQ DMSTRK DMSTYP DMSUPD DMSUTL DMSVIB
		DMSVIP DMSVLT DMSVSR DMSXBG DMSXCM DMSXCP DMSXDC DMSXDS DMSXED DMSXFD DMSXGT DMSXHL
		DMSXIN DMSXIO DMSXMA DMSXMD DMSXMS DMSXPO DMSXPT DMSXRE DMSXSC DMSXSD DMSXSE DMSXSG
		DMSXSS DMSXSU DMSXUP DMSZAP DMSZES
NUCOSFLG	000013	DMSABN DMSLOS DMSOSR DMSSLN DMSSTG

LABEL	COUNT	REFERENCES											
NUCOSRLD	000005	DMSLOS	DMSOSR	DMSSLN									
NUCOSRUN	000006	DMSLOS	DMSOSR	DMSSLN									
NUCSCBLK	000007	DMSINT	DMSITS										
NUCSYSDF	000003	DMSLOS	DMSSTG										
NUCTIEIN	000041	DMSLOS	DMSOSR	DMSSFF	DMSUTL								
NUM	000637	DMSABN	DMSACC	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSBOP	DMSFOR	DMSFRE	DMSGND
		DMSSET	DMSSPR	DMSSRT	DMSSSK	DMSSTT	DMSSVT						
NUMBYTE	000005	DMSLDR	DMSLIB	DMSOLD									
NUMFINRD	000031	DMSABN	DMSBTP	DMSCAT	DMSCIT	DMSCRD	DMSEDI	DMSEXT	DMSSVN	DMSXIO	DMSXMD	DMSXSC	DMSXSE
		DMSXSS											
NUMLOC	000002	DMSEDX	DMSSCR										
NUMPNDWR	000016	DMSCIT	DMSCRD	DMSCWR	DMSCWT	DMSITE	DMSSVN						
NXTSYM	000004	DMSLDR	DMSLSY	DMSOLD									
OCTCPDI	000003	DMSBOP	DMSVLT										
OCTDITYP	000001	DMSBOP											
OCTDXBUF	000001	DMSBOP											
OCTGVSIZ	000001	DMSBOP											
OCTMONAD	000005	DMSBOP	DMSCLS										
OCTMONSV	000003	DMSBOP	DMSCLS										
OCTS	000003	DMSBOP	DMSCLS	DMSVLT									
OCTSDWDS	000003	DMSBOP	DMSVLT										
OCTSLEN	000002	DMSBOP											
OCTSPMGT	000001	DMSVLT											
OCTSPPSV	000003	DMSBOP	DMSCLS										
OCTSTADR	000001	DMSBOP											
OCT1FLAG	000005	DMSBOP	DMSVLT										
OFFL	000006	DMSHLP											
OFFLI	000008	DMSHLP											
OLDCOUNT	000001	DMSHLP											
OLDEST	000001	DMSITI											
OLDPSW	000081	DMSABN	DMSDOS	DMSERR	DMSIFC	DMSITS	DMSOVS	DMSSLN	DMSSTG	DMSSVT			
OPENSW	000003	DMSBOP	DMSHLS										
OPSECT	000032	DMSABN	DMSARX	DMSASM	DMSCPY	DMSCRD	DMSCWR	DMSCWT	DMSDBG	DMSEXC	DMSEXI	DMSEXT	DMSINS
		DMSINT	DMSROS	DMSSBD	DMSSBS	DMSSCT	DMSSEB	DMSSOP	DMSSQS	DMSSVN	DMSSVT	DMSSVU	
OPSW	000016	DMSITP											
OPTBYTE	000087	DMSACC	DMSTPE	DMSTPF	DMSTPG								
OPTFLAGS	000036	DMSABN	DMSDOS	DMSFRE	DMSINA	DMSINS	DMSINT	DMSQRY	DMSSET	DMSSMN	DMSSTG	DMSSYN	DMSXCM
OPTNBYTE	000001	DMSSTG											
ORG	000004	DMSDBG											
OSADTDSK	000027	DMSDOS	DMSLAB	DMSLDS	DMSROS	DMSSET							
OSADTFST	000005	DMSABN	DMSALU	DMSROS									
OSADTVTA	000015	DMSACM	DMSLDS	DMSROS									
OSADTVTB	000020	DMSLDS	DMSROS										
OSFST	000016	DMSABN	DMSALU	DMSBOP	DMSDLK	DMSDSV	DMSFCH	DMSMVE	DMSMVG	DMSOPL	DMSROS	DMSRRV	DMSSOP
		DMSSRV	DMSSTT										
OSFSTALT	000009	DMSROS											

LABEL	COUNT	REFERENCES
OSFSTBLK	000005	DMSMVE DMSROS DMSSOP
OSFSTCHR	000014	DMSROS DMSSOP
OSFSTDBK	000002	DMSROS
OSFSTDSK	000010	DMSDLK DMSFCH DMSOPL DMSROS DMSRRV DMSSRV
OSFSTDSN	000006	DMSMVG DMSROS
OSFSTEND	000007	DMSROS
OSFSTEX4	000007	DMSROS
OSFSTFLG	000023	DMSROS DMSSTT
OSFSTFM	000008	DMSBOP DMSMVG 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	000010	DMSROS
OSFSTTYP	000003	DMSROS
OSFSTUMV	000001	DMSROS
OSFSTXNO	000004	DMSROS
OSFSTXTN	000016	DMSDLK DMSDSV DMSFCH DMSOPL DMSROS DMSRRV DMSSRV
OSIOTYPE	000018	DMSARX DMSASM DMSSBS DMSSOP DMSSQS DMSSVT
OSMODLDW	000013	DMSABN DMSINS DMSSET
OSRESET	000009	DMSEXT DMSINT DMSLDR DMSOLD DMSSLN DMSSVT
OSSFLAGS	000062	DMSARN DMSARX DMSASM DMSCIT DMSEXT DMSIFC DMSINT DMSITE DMSLDR DMSLIB DMSLIO DMSOLD
		DMSSLN DMSSMN DMSSTG DMSSVN DMSSVT DMSTLB
OSSMNU	000005	DMSSMN
OSTEMP	000030	DMSDOS DMSSLN DMSSVT DMSSVU
OSWAIT	000006	DMSCIT DMSITE DMSSVN
OUTBUF	000089	DMSLDR DMSLGT DMSLIB DMSLIO DMSLSB DMSOLD DMSPRE DMSRRV DMSSRV
OUTBUFF	000029	DMSLBM DMSRNE
OUTCOMM	000008	DMSLBM DMSTPE DMSTPF DMSTPG
OUTDISK	000006	DMSTPE DMSTPF DMSTPG
OUTFV	000001	DMSLBM
OUTITNO	000025	DMSLBM
OUTMODE	000008	DMSLBM DMSTPE DMSTPF DMSTPG
OUTNAME	000018	DMSLBM DMSPRE DMSTPE DMSTPF DMSTPG
OUTNOIT	000001	DMSLBM
OUTPARM	000001	DMSHLI
OUTPRINT	000007	DMSAMS DMSTPE DMSTPF DMSTPG
OUTPT1	000010	DMSDBG
OUTPUT	000047	DMSBOP DMSDLK DMSDSL DMSGRN DMSLAB DMSLDR DMSLIO DMSMVE DMSOLD DMSOR1 DMSOVS DMSPRE
		DMSQRY DMSTPE DMSTPF DMSTPG DMSUTL DMSXCP
OUTPUT10	000004	DMSQRY DMSTPE DMSTPF DMSTPG

LABEL	COUNT	REFERENCES
OUTSIZE	000010	DMSLBM DMSTPE DMSTPF DMSTPG
OUTSVC	000003	DMSTPE DMSTPF DMSTPG
OUTTERM	000006	DMSTPE DMSTPF DMSTPG
OUTTYPE	000001	DMSLBM
OVAPF	000004	DMSOVR DMSOVS
OVBPF	000005	DMSOVR DMSOVS
OVERLAP	000021	DMSDSK DMSTPE DMSTPF DMSTPG
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
OVSAFT	000011	DMSOVS
OVSECT	000003	DMSITS DMSOVR
OVSHO	000004	DMSCIT DMSOVR DMSOVS
OVSON	000012	DMSCIT DMSOVR DMSOVS
OVSSO	000006	DMSCIT DMSOVR DMSOVS
OVSTAT	000033	DMSCIT DMSITE DMSOVR DMSOVS
PACK	000035	DMSASN DMSBOP DMSBTP DMSCIT DMSCPY DMSDLK DMSEDI DMSFLD DMSLIO DMSRNE DMSTLB DMSTMA DMSXSE
PACKNUM	000006	DMSTPE DMSTPF DMSTPG
PADBUF	000017	DMSEDI DMSEDX
PADCHAR	000007	DMSEDI DMSEDX
PARM	000009	DMSDLB DMSGRN DMSHLI DMSOSR
PARMLIST	000015	DMSGRN DMSHLP DMSLDR DMSLIO DMSOLD
PARMPUT	000001	DMSHLP
PBUFF	000023	DMSEXT DMSHLP DMSTIO
PBUFFCT	000003	DMSHLP
PBUFFSZ	000007	DMSHLP DMSTIO
PCT	000009	DMSTPE DMSTPF DMSTPG
PCTVSAM	000002	DMSFCH DMSSTG
PDSBLKSI	000009	DMSSVT
PDSDIR	000016	DMSSVT
PDSDIRIT	000008	DMSSVT
PDSDIRSZ	000020	DMSSTG DMSSVT
PDSENTSZ	000040	DMSSVT
PDSFLG1	000006	DMSSVT
PDSFNEW	000007	DMSSVT
PDSHDRSZ	000002	DMSSVT

LABEL	COUNT	REFERENCES
PDSIDENT	000025	DMSSVT
PDSLEN	000004	DMSSTG DMSSVT
PDSSECT	000005	DMSSTG DMSSVT
PDSTEMPF	000003	DMSSVT
PENDLOG	000001	DMSSTG
PENDREAD	000022	DMSCIT DMSCRD DMSCWR DMSCWT DMSITE DMSSVN
PENDWRIT	000011	DMSCIT DMSCWR DMSSVN
PERASE	000003	DMSTPE DMSTPF DMSTPG
PERFORM	000005	DMSTPE DMSTPF DMSTPG
PERFORM1	000003	DMSTPE DMSTPF DMSTPG
PERF003	000003	DMSTPE DMSTPF DMSTPG
PERF004	000303	DMSTPE DMSTPF DMSTPG
PERF005	000003	DMSTPE DMSTPF DMSTPG
PERF010	000003	DMSTPE DMSTPF DMSTPG
PERF015	000003	DMSTPE DMSTPF DMSTPG
PERF020	000003	DMSTPE DMSTPF DMSTPG
PERF030	000006	DMSTPE DMSTPF DMSTPG
PERF040	000012	DMSTPE DMSTPF DMSTPG
PERF110	000003	DMSTPE DMSTPF DMSTPG
PERF120	000003	DMSTPE DMSTPF DMSTPG
PERF130	000003	DMSTPE DMSTPF DMSTPG
PERF140	000003	DMSTPE DMSTPF DMSTPG
PERF150	000003	DMSTPE DMSTPF DMSTPG
PERF210	000003	DMSTPE DMSTPF DMSTPG
PERF220	000003	DMSTPE DMSTPF DMSTPG
PERMSW	000004	DMSHLS
PERSU	000004	DMSHLP
PGEND	000001	DMSSTG
PGMNPSW	000008	DMSABN DMSINS DMSITP
PGMOPSW	000017	DMSABN DMSDBG DMSITP DMSSAB
PGMSECT	000006	DMSITP DMSSAB DMSSLN DMSSTG DMSSVT
PIBADR	000005	DMSCVH DMSDOS
PIBFLG	000001	DMSDOS
PIBLUBNO	000002	DMSCVH DMSDOS
PIBPT	000020	DMSAMS DMSBOP DMSCLS DMSCVH DMSDOS DMSITP DMSSET DMSVLT
PIB2PTR	000001	DMSDOS
PICADDR	000004	DMSITP DMSSTG
PIE	000002	DMSITP
PIK	000002	DMSDOS
PLIST	000134	DMSBOP DMSBRD DMSBWR DMSCLS DMSDIO DMSDLK DMSDMP DMSDSV DMSEDI DMSEDX DMSEXC DMSEXE
		DMSINT DMSLBM DMSLOS DMSMDP DMSMVE DMSOSR DMSRNE DMSSOP DMSSVT DMSSVU DMSTIO DMSTLB
		DMSUPD DMSXRE
PLISTSAV	000018	DMSLDR DMSLIO DMSOLD
PLONE	000002	DMSHLP DMSHLS
PLSTFV	000001	DMSEDI
PLSTITEM	000006	DMSEDI DMSEDX

LABEL	COUNT	REFERENCES											
RECFTYPE	000001	DMSXMA											
RECLGHDR	000002	DMSXMA											
RECLRECD	000002	DMSXMA											
RECMCBFL	000002	DMSXMA											
RECMCBUF	000005	DMSXMA											
RECPLIST	000002	DMSXMA											
RECS	000009	DMSEDX	DMSMVG										
RECSAVE	000003	DMSXMA											
RECSAVRG	000002	DMSXMA											
RECSAV13	000002	DMSXMA											
RECZMAPT	000001	DMSXMA											
REDERRID	000005	DMSCWR	DMSINT	DMSQRY	DMSSET								
REFCMD	000004	DMSLDR	DMSOLD										
REFLG1	000008	DMSLDR	DMSOLD										
REFLG2	000004	DMSLDR	DMSOLD										
REFLIB	000006	DMSLDR	DMSOLD										
REFUND	000004	DMSLDR	DMSOLD										
REGSAV	000025	DMSEDI	DMSINS	DMSUPD	DMSVSR								
REGSAVE	000006	DMSDLK	DMSHLI										
REGSAVE2	000007	DMSHLB	DMSHLI										
REGSAVX	000007	DMSEDI											
REGSAV0	000048	DMSACF	DMSACM	DMSALU	DMSAUD	DMSERD	DMSLAD	DMSLFS					
REGSAV1	000022	DMSACF	DMSERD	DMSERS	DMSRNM								
REGSAV3	000052	DMSBRD	DMSBWR	DMSERD	DMSFNS	DMSMOD	DMSPNT	DMSSTT					
REG13SAV	000003	DMSLDR	DMSOLD										
RELPAGES	000021	DMSABN	DMSAMS	DMSARN	DMSARX	DMSASM	DMSCPY	DMSEDI	DMSINT	DMSLBM	DMSLBT	DMSLKD	DMSSRT
		DMSSTG	DMSUPD	DMSXBG									
RELPHSE	000002	DMSFCH											
REPCNT	000010	DMSEDI	DMSEDX										
REPL	000017	DMSEDI	DMSLGT	DMSUPD									
REQADR	000002	DMSXDC											
REQDES	000002	DMSXDC	DMSXHL										
REQLGMIN	000002	DMSXDC	DMSXHL										
REQLNAME	000001	DMSXDC											
REQNAME	000004	DMSXDC	DMSXHL										
REQNBOPR	000003	DMSXDC	DMSXHL										
REQPARM1	000001	DMSXDC											
RESET	000160	DMSACC	DMSAMS	DMSARN	DMSARX	DMSASM	DMSBOP	DMSBTB	DMSBTP	DMSBWR	DMSCLS	DMSCPY	DMSDAS
		DMSDLB	DMSDLK	DMSDOS	DMSDSL	DMSDSV	DMSEDI	DMSFLD	DMSFOR	DMSHLI	DMSHLL	DMSIFC	DMSITE
		DMSITP	DMSLAB	DMSLBM	DMSLBT	DMSLDR	DMSLDS	DMSLSB	DMSMVE	DMSOLD	DMSOPT	DMSPRV	DMSRRV
		DMSSAB	DMSSCT	DMSSET	DMSSOP	DMSSPR	DMSSRT	DMSSRV	DMSSVU	DMSTLB	DMSTMA	DMSTPE	DMSTPF
		DMSTPG	DMSUPD	DMSVIB	DMSVIP	DMSXBG	DMSXCM	DMSXCT	DMSXED	DMSXFD	DMSXGT	DMSXHL	DMSXIN
		DMSXMA	DMSXPO	DMSXPT	DMSXSC	DMSXTB	DMSZAP						
RESETM7	000003	DMSTPE	DMSTPF	DMSTPG									
RESET7	000009	DMSTPE	DMSTPF	DMSTPG									
RETCODE	000034	DMSEXE	DMSEXT	DMSHLE	DMSHLI	DMSHLS	DMSLBT	DMSUPD					

LABEL	COUNT	REFERENCES
RETREG	000009	DMSDIO DMSLDR DMSLST DMSOLD
RETRYBIT	000002	DMSSAB
RETSAV	000006	DMSDBG DMSVIP
RETT	000005	DMSLSB
RETURN	000372	DMSARN DMSARX DMSASM DMSBRD DMSBTP DMSCMP DMSDBG DMSDIO DMSDLB DMSDOS DMSEDI DMSERD DMSEXE DMSFET DMSFLD DMSFNC DMSFRE DMSGLB DMSGRN DMSHDI DMSHDS DMSIMA DMSIOW DMSITE DMSITI DMSLBM DMSLIB DMSLIO DMSLKD DMSLLU DMSLOA DMSLOS DMSOR1 DMSOR2 DMSOR3 DMSOVR DMSPNT DMSRDC DMSRNE DMSSMN DMSSOP DMSSTG DMSTIO DMSTLB DMSTPD DMSTPE DMSTPF DMSTPG DMSTQQ DMSTYP DMSUPD DMSXSC
REW	000023	DMSBOP DMSCLS DMSIFC DMSTLB DMSTPE DMSTPF DMSTPG
REWIND	000007	DMSCLS DMSTPE DMSTPF DMSTPG
RPIX	000001	DMSLGT
RFPRS	000001	DMSOVS
RGPRS	000007	DMSOVS DMSSET
RGPR8	000001	DMSOVS
RITEM	000007	DMSLBT DMSLGT DMSLIB
RLDCONST	000008	DMSLDR DMSOLD
RLENG	000002	DMSLGT DMSLIB
RMARGIN	000005	DMSHLP
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
RPLOPT1	000004	DMSVIP
RPLOPT2	000001	DMSVIP
RPLRLEN	000001	DMSVIP
RPLRTNCD	000006	DMSVIP
RPLST	000002	DMSVIP
RPLSTRID	000001	DMSVIP
RPLUPD	000001	DMSVIP
RPLVLERR	000001	DMSVIP
RRDPT	000001	DMSLIB
RSTNPSW	000002	DMSDBG
RUN	000005	DMSCLS DMSGRN DMSSEB

LABEL	COUNT	REFERENCES											
RWCCW	000003	DMSDIO											
RWCNT	000021	DMSACC	DMSACF	DMSACM	DMSAUD	DMSERD	DMSERS	DMSFNS	DMSMOD				
RWCNT2	000002	DMSERD											
RWFSTRG	000014	DMSAUD	DMSBRD	DMSBWR	DMSERD	DMSFNS							
RWMFD	000009	DMSACM	DMSAUD										
RWRPT	000002	DMSLIB											
R0	005297	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCAT	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD
		DMSCVH	DMSCWR	DMSCWT	DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSDSK	DMSDSL
		DMSEDC	DMSEDI	DMSEDX	DMSERD	DMSERR	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH
		DMSFET	DMSFLD	DMSFNS	DMSFOR	DMSGIO	DMSGLB	DMSGND	DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLI
		DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI	DMSINM	DMSINS	DMSINT	DMSIOW	DMSITE
		DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBM	DMSLBR	DMSLBT	DMSLCK	DMSLDR
		DMSLDS	DMSLFS	DMSLGT	DMSLIB	DMSLIO	DMSLKD	DMSLLU	DMSLOA	DMSLOS	DMSLSB	DMSLST	DMSLSY
		DMSMDP	DMSMOD	DMSMVE	DMSMVG	DMSNCP	DMSOLD	DMSOPL	DMSOPT	DMSOR1	DMSOSR	DMSOVR	DMSOVS
		DMSPNT	DMSPRE	DMSPRT	DMSPRV	DMSPUN	DMSQRY	DMSRDC	DMSREA	DMSRNE	DMSRNM	DMSROS	DMSRRV
		DMSSAB	DMSSBD	DMSSBS	DMSSCN	DMSSCR	DMSSCT	DMSSEB	DMSSFF	DMSSMN	DMSSOP	DMSSPR	DMSSQS
		DMSSRT	DMSSRV	DMSSSK	DMSSTG	DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSSYN	DMSTIO	DMSTLA	DMSTLB
		DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIB	DMSVIP	DMSVLT	DMSXBG	DMSXCG
		DMSXCM	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL
		DMSXIN	DMSXIO	DMSXMA	DMSXMS	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXRE	DMSXSC
		DMSXSD	DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES			
R1	011674	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCAT	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD
		DMSCVH	DMSCWR	DMSCWT	DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSDSK	DMSDSL
		DMSEDC	DMSEDI	DMSEDX	DMSERD	DMSERR	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH
		DMSFET	DMSFLD	DMSFNS	DMSFOR	DMSGIO	DMSGLB	DMSGND	DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLE
		DMSHLI	DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI	DMSINM	DMSINS	DMSINT	DMSIOW
		DMSITE	DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBM	DMSLBR	DMSLBT	DMSLCK
		DMSLDR	DMSLDS	DMSLFS	DMSLGT	DMSLIB	DMSLIO	DMSLKD	DMSLLU	DMSLOA	DMSLOS	DMSLSB	DMSLST
		DMSLSY	DMSMDP	DMSMOD	DMSMVE	DMSMVG	DMSNCP	DMSOLD	DMSOPL	DMSOPT	DMSOR1	DMSOR2	DMSOR3
		DMSOSR	DMSOVR	DMSOVS	DMSPIO	DMSPNT	DMSPRE	DMSPRT	DMSPRV	DMSPUN	DMSQRY	DMSRDC	DMSREA
		DMSRNE	DMSRNM	DMSROS	DMSRRV	DMSSAB	DMSSBD	DMSSBS	DMSSCN	DMSSCR	DMSSCT	DMSSEB	DMSSFF
		DMSSMN	DMSSOP	DMSSPR	DMSSQS	DMSSRT	DMSSRV	DMSSSK	DMSSTG	DMSSTT	DMSSVN	DMSSVT	DMSSVU
		DMSSYN	DMSTIO	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIB	DMSVIP
		DMSVLT	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC
		DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT
		DMSXPX	DMSXRE	DMSXSC	DMSXSD	DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES
R10	002632	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD	DMSBOP
		DMSBRD	DMSBTP	DMSBWR	DMSCIO	DMSCLS	DMSCPF	DMSCPY	DMSCVH	DMSCWR	DMSCWT	DMSDAS	DMSDBD
		DMSDBG	DMSDIO	DMSDLB	DMSDOS	DMSDSK	DMSDSL	DMSEDC	DMSEDI	DMSEDX	DMSERD	DMSERR	DMSERS
		DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFLD	DMSFNS	DMSFOR	DMSGIO	DMSGRN	DMSHDI
		DMSHDS	DMSHLB	DMSHLI	DMSHLL	DMSHLP	DMSHLS	DMSIMA	DMSINI	DMSINM	DMSINS	DMSINT	DMSIOW
		DMSITE	DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLBD	DMSLBM	DMSLBT	DMSLDR	DMSLDS	DMSLFS
		DMSLGT	DMSLIO	DMSLKD	DMSLLU	DMSLOS	DMSLSB	DMSLST	DMSMOD	DMSMVE	DMSMVG	DMSNCP	DMSOLD
		DMSOPT	DMSPIO	DMSPRE	DMSPRT	DMSPRV	DMSPUN	DMSQRY	DMSRDC	DMSRNE	DMSRNM	DMSROS	DMSRRV

LABEL	COUNT	REFERENCES											
		DMSSAB	DMSSBD	DMSSCR	DMSSFF	DMSSMN	DMSSOP	DMSSPR	DMSSQS	DMSSRV	DMSSTG	DMSSTT	DMSSVN
		DMSSVT	DMSSVU	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIP	DMSVLT
		DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD
		DMSXGT	DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX
		DMSXRE	DMSXSC	DMSXSD	DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP		
R11	001187	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD	DMSBOP
		DMSBRD	DMSBTP	DMSBWR	DMSCIO	DMSCLS	DMSCPY	DMSCRD	DMSCWR	DMSCWT	DMSDAS	DMSDBD	DMSDIO
		DMSDLB	DMSDOS	DMSDSK	DMSERD	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSFCH	DMSFLD	DMSFNS
		DMSFOR	DMSGLB	DMSGND	DMSGRN	DMSHLB	DMSHLP	DMSINI	DMSINS	DMSINT	DMSIOW	DMSITE	DMSITI
		DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBT	DMSLCK	DMSLDR	DMSLDS	DMSLFS	DMSLIB
		DMSLIO	DMSLKD	DMSLLU	DMSLOS	DMSLSB	DMSLST	DMSMOD	DMSMVG	DMSNCP	DMSOLD	DMSOPT	DMSPIO
		DMSPNT	DMSPRE	DMSPRT	DMSPUN	DMSQRY	DMSRDC	DMSRNM	DMSROS	DMSRRV	DMSSAB	DMSSBD	DMSSBS
		DMSSCR	DMSSCT	DMSSEB	DMSSFF	DMSSOP	DMSSPR	DMSSQS	DMSSTG	DMSSVT	DMSSYN	DMSTIO	DMSTLB
		DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSUPD	DMSUTL	DMSVIP	DMSVLT	DMSXBG	DMSXCG	DMSXCM	DMSXCN
		DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXIO
		DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXRE	DMSXSC	DMSXSD	DMSXSE
		DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP							
R12	001422	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCAT	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD
		DMSCVH	DMSCWR	DMSCWT	DMSDAS	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSDSK	DMSDSL	DMSEDX	DMSERD
		DMSERR	DMSERS	DMSEXC	DMSEXE	DMSEXI	DMSFCH	DMSFET	DMSFLD	DMSFNS	DMSFOR	DMSGLB	DMSGND
		DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLI	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINI	DMSINS	DMSINT
		DMSITE	DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBM	DMSLBR	DMSLBT	DMSLDR
		DMSLDS	DMSLFS	DMSLGT	DMSLIB	DMSLKD	DMSLLU	DMSLOA	DMSLOS	DMSLSB	DMSLST	DMSMOD	DMSMVE
		DMSMVG	DMSNCP	DMSOLD	DMSOPL	DMSOPT	DMSOR1	DMSOR2	DMSOR3	DMSOSR	DMSOVR	DMSOVS	DMSPIO
		DMSPNT	DMSPRE	DMSPRT	DMSPRV	DMSPUN	DMSQRY	DMSREA	DMSRNE	DMSRNM	DMSROS	DMSRRV	DMSSAB
		DMSSBD	DMSSBS	DMSSCN	DMSSCR	DMSSCT	DMSSFF	DMSSMN	DMSSOP	DMSSPR	DMSSQS	DMSSRT	DMSSRV
		DMSSSK	DMSSTG	DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSSYN	DMSTIO	DMSTLA	DMSTLB	DMSTMA	DMSTPD
		DMSTPE	DMSTRK	DMSUPD	DMSUTL	DMSVIB	DMSVIP	DMSVLT	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCP
		DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXIO	DMSXMA
		DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXRE	DMSXSC	DMSXSD	DMSXSE	DMSXSS
		DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES							
R13	002445	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD	DMSBOP
		DMSBRD	DMSBTP	DMSBWR	DMSCAT	DMSCIO	DMSCIT	DMSCLS	DMSCPY	DMSCRD	DMSCVH	DMSCWR	DMSDAS
		DMSDBG	DMSDIO	DMSDLB	DMSDOS	DMSDSK	DMSEDC	DMSEDI	DMSEDX	DMSERD	DMSERR	DMSERS	DMSEXC
		DMSEXE	DMSEXI	DMSFCH	DMSFLD	DMSFNS	DMSFOR	DMSGIO	DMSGLB	DMSGND	DMSGRN	DMSHDI	DMSHDS
		DMSHLB	DMSHLD	DMSHLE	DMSHLI	DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSINI	DMSINS	DMSINT	DMSITE
		DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLAF	DMSLBM	DMSLBR	DMSLBT	DMSLCK	DMSLDR	DMSLDS
		DMSLFS	DMSLGT	DMSLIB	DMSLIO	DMSLOS	DMSLSB	DMSLST	DMSMOD	DMSMVE	DMSMVG	DMSNCP	DMSOLD
		DMSOVS	DMSPIO	DMSPNT	DMSPRT	DMSPUN	DMSQRY	DMSREA	DMSRNE	DMSRNM	DMSSAB	DMSSBS	DMSSCR
		DMSSCT	DMSSEB	DMSSFF	DMSSMN	DMSSOP	DMSSQS	DMSSTG	DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSTIO
		DMSTLB	DMSTMA	DMSTPE	DMSTRK	DMSUPD	DMSUTL	DMSVIP	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCP
		DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXIO	DMSXMA
		DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXRE	DMSXSC	DMSXSD	DMSXSE	DMSXSG
		DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP							
R14	006293	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD

LABEL	COUNT	REFERENCES											
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCAT	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD
		DMSCVH	DMSCWR	DMSCWT	DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDOS	DMSDSK	DMSDSL	DMSEDC
		DMSEDI	DMSEDX	DMSERD	DMSERR	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFET
		DMSFLD	DMSFNS	DMSFOR	DMSGIO	DMSGLB	DMSGND	DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLE	DMSHLI
		DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI	DMSINM	DMSINS	DMSINT	DMSIOW	DMSITE
		DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBM	DMSLBR	DMSLBT	DMSLCK	DMSLDR
		DMSLDS	DMSLFS	DMSLGT	DMSLIB	DMSLIO	DMSLKD	DMSLLU	DMSLOA	DMSLOS	DMSLSB	DMSLST	DMSLSY
		DMSMDP	DMSMOD	DMSMVE	DMSMVG	DMSNCP	DMSOLD	DMSOPL	DMSOPT	DMSOR3	DMSOSR	DMSOVR	DMSOVS
		DMSPIO	DMSPNT	DMSPRE	DMSPRT	DMSPRV	DMSPUN	DMSQRY	DMSRDC	DMSREA	DMSRNE	DMSRNM	DMSROS
		DMSRRV	DMSSAB	DMSSBD	DMSSBS	DMSSCN	DMSSCR	DMSSCT	DMSSEB	DMSSFF	DMSSMN	DMSSOP	DMSSPR
		DMSSQS	DMSSRT	DMSSRV	DMSSSK	DMSSTG	DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSSYN	DMSTIO	DMSTLA
		DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIB	DMSVIP	DMSVLT	DMSXBG
		DMSXCG	DMSXCM	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT
		DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXRE
		DMSXSC	DMSXSD	DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES		
R15	012390	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCAT	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD
		DMSCVH	DMSCWR	DMSCWT	DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDOS	DMSDSK	DMSDSL	DMSEDC
		DMSEDI	DMSEDX	DMSERD	DMSERR	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFET
		DMSFLD	DMSFNS	DMSFOR	DMSGIO	DMSGLB	DMSGND	DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLE	DMSHLI
		DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI	DMSINM	DMSINS	DMSINT	DMSIOW	DMSITE
		DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBM	DMSLBR	DMSLBT	DMSLCK	DMSLDR
		DMSLDS	DMSLFS	DMSLGT	DMSLIB	DMSLIO	DMSLKD	DMSLLU	DMSLOA	DMSLOS	DMSLSB	DMSLST	DMSLSY
		DMSMDP	DMSMOD	DMSMVE	DMSMVG	DMSNCP	DMSOLD	DMSOPL	DMSOPT	DMSOR1	DMSOSR	DMSOVR	DMSOVS
		DMSPIO	DMSPNT	DMSPRE	DMSPRT	DMSPRV	DMSPUN	DMSQRY	DMSRDC	DMSREA	DMSRNE	DMSRNM	DMSROS
		DMSRRV	DMSSAB	DMSSBD	DMSSBS	DMSSCN	DMSSCR	DMSSCT	DMSSEB	DMSSFF	DMSSMN	DMSSOP	DMSSPR
		DMSSQS	DMSSRT	DMSSRV	DMSSSK	DMSSTG	DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSSYN	DMSTIO	DMSTLA
		DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIB	DMSVIP	DMSVLT	DMSXBG
		DMSXCG	DMSXCM	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT
		DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXRE
		DMSXSC	DMSXSD	DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES		
R2	007231	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCAT	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD
		DMSCVH	DMSCWR	DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSDSK	DMSDSL	DMSEDC
		DMSEDI	DMSEDX	DMSERD	DMSERR	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFET
		DMSFLD	DMSFNS	DMSFOR	DMSGIO	DMSGLB	DMSGND	DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLE	DMSHLI
		DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI	DMSINM	DMSINS	DMSINT	DMSIOW	DMSITE
		DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBM	DMSLBR	DMSLBT	DMSLCK	DMSLDR	DMSLDS
		DMSLFS	DMSLGT	DMSLIO	DMSLKD	DMSLLU	DMSLOA	DMSLOS	DMSLSB	DMSLST	DMSMDP	DMSMOD	DMSMVE
		DMSMVG	DMSNCP	DMSOLD	DMSOPL	DMSOPT	DMSOR1	DMSOSR	DMSPIO	DMSPNT	DMSPRE	DMSPRT	DMSPRV
		DMSPUN	DMSQRY	DMSRDC	DMSREA	DMSRNE	DMSRNM	DMSROS	DMSRRV	DMSSAB	DMSSBD	DMSSBS	DMSSCN
		DMSSCR	DMSSCT	DMSSEB	DMSSFF	DMSSMN	DMSSOP	DMSSPR	DMSSQS	DMSSRT	DMSSRV	DMSSSK	DMSSTG
		DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSSYN	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD
		DMSUTL	DMSVIB	DMSVIP	DMSVLT	DMSXBG	DMSXCG	DMSXCM	DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED
		DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS
		DMSXPO	DMSXPT	DMSXPX	DMSXRE	DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP

LABEL	COUNT	REFERENCES											
		DMSZES											
R3	006929	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCAT	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD
		DMSCVH	DMSCWR	DMSDAS	DMSDBD	DMSDBG	DMSDLB	DMSDMP	DMSDOS	DMSDSK	DMSDSL	DMSEDC	DMSEDI
		DMSEDX	DMSERD	DMSERR	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFET	DMSFLD
		DMSFOR	DMSGIO	DMSGLB	DMSGND	DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLE	DMSHLI	DMSHLL	DMSHLP
		DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI	DMSINM	DMSINS	DMSINT	DMSITE	DMSITI	DMSITP	DMSITS
		DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBM	DMSLBR	DMSLBT	DMSLCK	DMSLDR	DMSLDS	DMSLFS	DMSLGT
		DMSLIO	DMSLKD	DMSLLU	DMSLOS	DMSLSB	DMSLST	DMSMDP	DMSMOD	DMSMVE	DMSMVG	DMSNCP	DMSOLD
		DMSOPL	DMSOSR	DMSOVR	DMSOVS	DMSPIO	DMSPNT	DMSPRE	DMSPRT	DMSPRV	DMSPUN	DMSQRY	DMSRDC
		DMSREA	DMSRNE	DMSRNM	DMSROS	DMSRRV	DMSSAB	DMSSBD	DMSSBS	DMSSCN	DMSSCR	DMSSCT	DMSSEB
		DMSSFF	DMSSMN	DMSSOP	DMSSPR	DMSSQS	DMSSRT	DMSSRV	DMSSSK	DMSSTG	DMSSTT	DMSSVN	DMSSVT
		DMSSVU	DMSSYN	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIB	DMSVIP
		DMSVLT	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXED	DMSXER	DMSXFC	DMSXFD
		DMSXGT	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXRE
		DMSXSC	DMSXSD	DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES		
R4	005382	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCAT	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD
		DMSCVH	DMSCWR	DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSDSK	DMSDSL	DMSEDC
		DMSEDI	DMSEDX	DMSERD	DMSERR	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFET
		DMSFLD	DMSFOR	DMSGIO	DMSGLB	DMSGND	DMSGRN	DMSHDI	DMSHDS	DMSHLE	DMSHLI	DMSHLL	DMSHLP
		DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI	DMSINM	DMSINS	DMSINT	DMSIOW	DMSITI	DMSITP	DMSITS
		DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBM	DMSLBR	DMSLBT	DMSLCK	DMSLDR	DMSLDS	DMSLFS	DMSLGT
		DMSLIO	DMSLKD	DMSLLU	DMSLSB	DMSLST	DMSMDP	DMSMOD	DMSMVE	DMSMVG	DMSNCP	DMSOLD	DMSOPL
		DMSOSR	DMSOVR	DMSOVS	DMSPIO	DMSPNT	DMSPRE	DMSPRT	DMSPUN	DMSQRY	DMSRDC	DMSREA	DMSRNE
		DMSRNM	DMSROS	DMSRRV	DMSSAB	DMSSBD	DMSSBS	DMSSCN	DMSSCR	DMSSCT	DMSSEB	DMSSFF	DMSSMN
		DMSSOP	DMSSPR	DMSSQS	DMSSRT	DMSSRV	DMSSSK	DMSSTG	DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSSYN
		DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIP	DMSVLT	DMSXBG	DMSXCG
		DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXIN	DMSXMA	DMSXMC	DMSXML
		DMSXMS	DMSXPO	DMSXPX	DMSXRE	DMSXSC	DMSXSD	DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP
		DMSZAP	DMSZES										
R5	005345	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD	DMSCVH
		DMSCWR	DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSDSK	DMSDSL	DMSEDC	DMSEDI
		DMSEDX	DMSERD	DMSERR	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFET	DMSFLD
		DMSFNS	DMSFOR	DMSGIO	DMSGLB	DMSGND	DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLI	DMSHLL	DMSHLP
		DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI	DMSINM	DMSINS	DMSINT	DMSIOW	DMSITI	DMSITP	DMSITS
		DMSLAB	DMSLAD	DMSLAF	DMSLBD	DMSLBM	DMSLBR	DMSLBT	DMSLCK	DMSLDR	DMSLDS	DMSLFS	DMSLGT
		DMSLIB	DMSLKD	DMSLLU	DMSLOS	DMSLSB	DMSLST	DMSMOD	DMSMVE	DMSMVG	DMSNCP	DMSOLD	DMSOPL
		DMSOR1	DMSOSR	DMSOVR	DMSOVS	DMSPIO	DMSPNT	DMSPRE	DMSPRT	DMSPUN	DMSQRY	DMSRDC	DMSREA
		DMSRNE	DMSRNM	DMSROS	DMSRRV	DMSSAB	DMSSBD	DMSSBS	DMSSCN	DMSSCR	DMSSCT	DMSSFF	DMSSMN
		DMSSOP	DMSSPR	DMSSQS	DMSSRT	DMSSRV	DMSSSK	DMSSTG	DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSSYN
		DMSTIO	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIB	DMSVIP	DMSVLT
		DMSXCG	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL
		DMSXIN	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXRE	DMSXSC	DMSXSD
		DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES				

LABEL	COUNT	REFERENCES											
R6	004824	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTP	DMSBWR	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD	DMSCVH	DMSCWR
		DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSDSK	DMSEDC	DMSEDI	DMSEDX	DMSERD
		DMSERR	DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXT	DMSFCH	DMSFET	DMSFLD	DMSFNS	DMSFOR	DMSGND
		DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLI	DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI
		DMSINS	DMSINT	DMSIOW	DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLBD	DMSLBM	DMSLBR	DMSLBT
		DMSLCK	DMSLDR	DMSLDS	DMSLFS	DMSLGT	DMSLKD	DMSLLU	DMSLOA	DMSLOS	DMSLSB	DMSLST	DMSMOD
		DMSMVE	DMSMVG	DMSNCP	DMSOLD	DMSOPL	DMSOR1	DMSOSR	DMSOVR	DMSOVS	DMSPIO	DMSPNT	DMSPRE
		DMSPRT	DMSPUN	DMSQRY	DMSRDC	DMSREA	DMSRNE	DMSRNM	DMSROS	DMSRRV	DMSSAB	DMSSBD	DMSSBS
		DMSSCN	DMSSCR	DMSSCT	DMSSFF	DMSSMN	DMSSOP	DMSSPR	DMSSQS	DMSSRT	DMSSRV	DMSSSK	DMSSTG
		DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSSYN	DMSTIO	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP
		DMSUPD	DMSUTL	DMSVIP	DMSVLT	DMSXBG	DMSXCG	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXED	DMSXER
		DMSXFC	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT
		DMSXPX	DMSXRE	DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES	
R7	004550	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTP	DMSBWR	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCVH	DMSCWR	DMSDAS
		DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDMP	DMSDOS	DMSDSK	DMSEDC	DMSEDI	DMSEDX	DMSERD	DMSERR
		DMSERS	DMSETR	DMSEXC	DMSEXE	DMSEXT	DMSFCH	DMSFET	DMSFLD	DMSFNS	DMSFOR	DMSGLB	DMSGND
		DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLE	DMSHLI	DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINA
		DMSINI	DMSINS	DMSINT	DMSIOW	DMSITE	DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLBD	DMSLBM
		DMSLBR	DMSLBT	DMSLCK	DMSLDR	DMSLDS	DMSLFS	DMSLGT	DMSLIB	DMSLKD	DMSLLU	DMSLOS	DMSLSB
		DMSLST	DMSMOD	DMSMVE	DMSMVG	DMSOLD	DMSOPL	DMSOSR	DMSOVR	DMSOVS	DMSPIO	DMSPRE	DMSPRT
		DMSPUN	DMSQRY	DMSRDC	DMSREA	DMSRNE	DMSRNM	DMSROS	DMSRRV	DMSSAB	DMSSBD	DMSSCN	DMSSCR
		DMSSCT	DMSSFF	DMSSMN	DMSSOP	DMSSPR	DMSSQS	DMSSRT	DMSSRV	DMSSTG	DMSSTT	DMSSVT	DMSSVU
		DMSSYN	DMSTIO	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIP	DMSVLT
		DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT
		DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXRE	DMSXSC
		DMSXSD	DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES			
R8	004304	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTB	DMSBTP	DMSBWR	DMSCIO	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD	DMSCVH
		DMSCWR	DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDOS	DMSDSK	DMSDSL	DMSEDC	DMSEDI	DMSEDX
		DMSERD	DMSERR	DMSERS	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFLD	DMSFNS	DMSFOR	DMSGLB
		DMSGRN	DMSHDI	DMSHDS	DMSHLB	DMSHLI	DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI
		DMSINM	DMSINS	DMSIOW	DMSITE	DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLBD	DMSLBM	DMSLBT
		DMSLCK	DMSLDR	DMSLDS	DMSLFS	DMSLGT	DMSLLU	DMSLOS	DMSLSB	DMSLST	DMSMOD	DMSMVE	DMSMVG
		DMSNCP	DMSOLD	DMSOPL	DMSOSR	DMSOVR	DMSOVS	DMSPIO	DMSPRE	DMSPRT	DMSPUN	DMSQRY	DMSRDC
		DMSRNM	DMSROS	DMSRRV	DMSSAB	DMSSBD	DMSSBS	DMSSCN	DMSSCT	DMSSEB	DMSSFF	DMSSMN	DMSSOP
		DMSSPR	DMSSRV	DMSSSK	DMSSTG	DMSSTT	DMSSVN	DMSSVT	DMSSVU	DMSSYN	DMSTLA	DMSTLB	DMSTMA
		DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL	DMSVIP	DMSVLT	DMSXBG	DMSXCG	DMSXCM	DMSXCN
		DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXIO
		DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXSC	DMSXSD	DMSXSE	DMSXSG
		DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES						
R9	003534	DMSABN	DMSACC	DMSACF	DMSACM	DMSALU	DMSAMS	DMSARE	DMSARN	DMSARX	DMSASM	DMSASN	DMSAUD
		DMSBOP	DMSBRD	DMSBTP	DMSBWR	DMSCIT	DMSCLS	DMSCPF	DMSCPY	DMSCRD	DMSCVH	DMSCWR	DMSCWT
		DMSDAS	DMSDBD	DMSDBG	DMSDIO	DMSDLB	DMSDOS	DMSDSK	DMSEDC	DMSEDI	DMSEDX	DMSERD	DMSERR
		DMSERS	DMSEXC	DMSEXE	DMSEXI	DMSEXT	DMSFCH	DMSFLD	DMSFNS	DMSFOR	DMSGIO	DMSGND	DMSGRN

LABEL	COUNT	REFERENCES											
		DMSHDI	DMSHDS	DMSHLB	DMSHLE	DMSHLI	DMSHLL	DMSHLP	DMSHLS	DMSIFC	DMSIMA	DMSINA	DMSINI
		DMSINS	DMSINT	DMSIOW	DMSITI	DMSITP	DMSITS	DMSLAB	DMSLAD	DMSLBD	DMSLBM	DMSLBR	DMSLBT
		DMSLCK	DMSLDR	DMSLDS	DMSLFS	DMSLGT	DMSLKD	DMSLOS	DMSLSB	DMSLST	DMSMOD	DMSMVE	DMSMVG
		DMSNCP	DMSOLD	DMSOPL	DMSOSR	DMSPIO	DMSPRE	DMSPRT	DMSPUN	DMSQRY	DMSRDC	DMSRNM	DMSROS
		DMSRRV	DMSSAB	DMSSBD	DMSSCR	DMSSCT	DMSSFF	DMSSMN	DMSSOP	DMSSPR	DMSSRV	DMSSSK	DMSSTG
		DMSSTT	DMSSVT	DMSSVU	DMSTLA	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTRK	DMSTYP	DMSUPD	DMSUTL
		DMSVIP	DMSVLT	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCP	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER
		DMSXFC	DMSXFD	DMSXGT	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT
		DMSXPX	DMSXSC	DMSXSD	DMSXSE	DMSXSG	DMSXSS	DMSXST	DMSXSU	DMSXUP	DMSZAP	DMSZES	
SAMELEN	000012	DMSBWR	DMSERD										
SAVBWPTR	000761	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT
		DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXPX	DMSXSC	DMSXSD
		DMSXSE	DMSXSS	DMSXST	DMSXSU	DMSXUP							
SAVBYTE1	000185	DMSXCG	DMSXCT	DMSXDC	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXIN	DMSXMA	DMSXMC	DMSXMD
		DMSXPO	DMSXPT	DMSXPX	DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXSU	DMSXUP			
SAVBYTE2	000052	DMSXCG	DMSXDC	DMSXER	DMSXFD	DMSXIN	DMSXPO	DMSXPX	DMSXSC	DMSXSD	DMSXSS	DMSXSU	DMSXUP
SAVBYTE3	000017	DMSXED	DMSXIN	DMSXSC	DMSXSD	DMSXUP							
SAVBYTE4	000059	DMSXFD	DMSXIN	DMSXPX	DMSXSD	DMSXUP							
SAVCNT	000005	DMSEDI	DMSSCR										
SAVCWD	000022	DMSEDI											
SAVDWRD1	000160	DMSXDC	DMSXFD	DMSXHL	DMSXIN	DMSXIO	DMSXPO	DMSXPX	DMSXSC	DMSXSD	DMSXSE	DMSXSU	DMSXUP
SAVDWRD2	000050	DMSXDC	DMSXFD	DMSXIN	DMSXPO	DMSXSC	DMSXSD	DMSXSE	DMSXUP				
SAVEADT	000003	DMSDIO	DMSFNS										
SAVEAR	000010	DMSEDC	DMSSCR										
SAVEAREA	000022	DMSARN	DMSARX	DMSASM	DMSCPY	DMSDOS	DMSGRN	DMSHLE	DMSHLP	DMSIFC	DMSSRT	DMSUTL	
SAVEFM	000030	DMSTPE	DMSTPF	DMSTPG									
SAVEFN	000035	DMSBOP	DMSMVE	DMSTPE	DMSTPF	DMSTPG							
SAVEFORM	000015	DMSTPE	DMSTPF	DMSTPG									
SAVEFT	000003	DMSTPE	DMSTPF	DMSTPG									
SAVELNTH	000012	DMSTPE	DMSTPF	DMSTPG									
SAVEMODE	000006	DMSTPE	DMSTPF	DMSTPG									
SAVEREG	000174	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT
		DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXPX	DMSXSC	DMSXSD
		DMSXSE	DMSXSS	DMSXST	DMSXSU	DMSXUP							
SAVERR	000006	DMSTPE	DMSTPF	DMSTPG									
SAVER0	000020	DMSDSV	DMSIFC	DMSREA	DMSVIP								
SAVER1	000079	DMSIFC	DMSREA	DMSSOP	DMSTPE	DMSTPF	DMSTPG	DMSVIP					
SAVER14	000073	DMSEXE	DMSIFC	DMSREA	DMSSCT	DMSSEB	DMSSPR	DMSSVT	DMSTMA	DMSTPE	DMSTPF	DMSTPG	DMSVIP
SAVER15	000021	DMSIFC	DMSREA	DMSSOP	DMSSPR								
SAVEXT	000003	DMSINS	DMSTTE										
SAVE1	000021	DMSBOP	DMSDBD	DMSDBG	DMSDSL	DMSRRV	DMSSRV						
SAVE10R	000004	DMSDSK	DMSTPE	DMSTPF	DMSTPG								
SAVE2	000022	DMSBOP	DMSDBG	DMSIFC									
SAVE5	000015	DMSTPE	DMSTPF	DMSTPG									
SAVLSAVB	000168	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT
		DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXPX	DMSXSC	DMSXSD

LABEL	COUNT	REFERENCES											
		DMSXSE	DMSXSS	DMSXST	DMSXSU	DMSXUP							
SAVREG0	000363	DMSXCG	DMSXCM	DMSXCN	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXHL
		DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXPX	DMSXSC	DMSXSD	DMSXSE
		DMSXSS	DMSXST	DMSXSU	DMSXUP								
SAVREG1	000021	DMSXCN	DMSXDC	DMSXER	DMSXFD	DMSXIN	DMSXIO	DMSXPX	DMSXSD	DMSXSS	DMSXSU		
SAVREG10	000002	DMSXSU											
SAVREG12	000005	DMSXCT	DMSXED	DMSXMA	DMSXSC	DMSXSS							
SAVREG14	000006	DMSXER											
SAVREG15	000013	DMSXDS	DMSXFD	DMSXIN	DMSXSD	DMSXSU	DMSXUP						
SAVREG2	000009	DMSXFC	DMSXMA	DMSXSC	DMSXSD	DMSXST	DMSXSU						
SAVREG3	000002	DMSXSD	DMSXST										
SAVREG5	000001	DMSXSC											
SAVREG6	000002	DMSXSD											
SAVREG7	000001	DMSXSD											
SAVREG8	000002	DMSXFD											
SAVWORD1	000103	DMSXCG	DMSXCN	DMSXCT	DMSXDC	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXIN	DMSXMC	DMSXMD	DMSXSC
		DMSXSD	DMSXSE	DMSXSS	DMSXUP								
SAVWORD2	000031	DMSXCT	DMSXFD	DMSXIN	DMSXMD	DMSXSC	DMSXSD	DMSXSE	DMSXUP				
SAV67	000012	DMSLDR	DMSOLD										
SBA	000011	DMSXSC	DMSXSD										
SCAN	000044	DMSASN	DMSCRD	DMSDLK	DMSFLD	DMSLBM	DMSLBT	DMSLST	DMSSLN	DMSSYN			
SCANNING	000003	DMSASN	DMSFLD										
SCANNULL	000015	DMSTPE	DMSTPF	DMSTPG									
SCAN2	000011	DMSDLB	DMSLBM	DMSPRT	DMSPUN	DMSTPE	DMSTPF	DMSTPG					
SCAW	000003	DMSITE											
SCBENTR	000001	DMSITS											
SCBFWPTR	000006	DMSINT	DMSITS										
SCBLOCK	000007	DMSINT	DMSITS	DMSXIN	DMSXMA	DMSXMS							
SCBLOCKD	000003	DMSINT	DMSITS										
SCBNAME	000004	DMSITS											
SCBPSW	000001	DMSITS											
SCBPTR	000015	DMSITP	DMSSAB	DMSSLN	DMSSTG	DMSSVT							
SCBSAV12	000004	DMSSAB											
SCBWKWRD	000004	DMSITS	DMSXIN	DMSXMA	DMSXMS								
SCBWORK	000008	DMSSAB	DMSSTG										
SCLNO	000002	DMSSCR											
SCNSWT	000036	DMSTPE	DMSTPF	DMSTPG									
SCRBUFAD	000002	DMSEDX	DMSSCR										
SCRFLGS	000036	DMSEDI	DMSSCR										
SCRFLG2	000019	DMSEDI	DMSSCR										
SDISK	000004	DMSALU	DMSINI										
SEARCH	000054	DMSINI	DMSQRY										
SEBSAV	000013	DMSSBD	DMSSEB	DMSSVT									
SECTNUM	000005	DMSACM	DMSDIO	DMSFNS	DMSFOR	DMSITI							
SEEK	000040	DMSINI											
SEEKADR	000011	DMSACM	DMSDIO	DMSFNS	DMSFOR	DMSITI							

LABEL	COUNT	REFERENCES											
SEGORELO	000006	DMSFRE	DMSINS	DMSITS									
SENCCW	000003	DMSDIO	DMSSPR										
SENSB	000007	DMSACM	DMSDIO	DMSFNS	DMSFOR	DMSITI							
SENSE	000016	DMSBOP	DMSCLS	DMSPRV	DMSRRV	DMSSRV	DMSTLB	DMSXSC					
SEQNAME	000004	DMSEDI	DMSEDX										
SEPSAV	000002	DMSEDI											
SERTSEQ	000003	DMSEDI											
SERTSW	000003	DMSEDI											
SETD7TRK	000003	DMSTPE	DMSTPF	DMSTPG									
SETLIB	000002	DMSLIB											
SETRTCH	000003	DMSTPE	DMSTPF	DMSTPG									
SETSEC	000003	DMSINI											
SETUP	000025	DMSABN	DMSDSV	DMSEXT	DMSLIB	DMSSAB	DMSSCR	DMSSVU	DMSSYN	DMSTLB	DMSTPE	DMSTPF	DMSTPG
		DMSUTL											
SF	000033	DMSDLK	DMSFCH	DMSNCP	DMSUTL	DMSXBG	DMSXPO	DMSXSC	DMSXSD	DMSXSS			
SFSTADAT	000009	DMSTPE	DMSTPF	DMSTPG									
SFSTAIC	000003	DMSTPE	DMSTPF	DMSTPG									
SFSTDAT	000003	DMSTPE	DMSTPF	DMSTPG									
SFSTFOP	000003	DMSTPE	DMSTPF	DMSTPG									
SFSTFV	000006	DMSTPE	DMSTPF	DMSTPG									
SFSTIC	000006	DMSTPE	DMSTPF	DMSTPG									
SFSTRP	000003	DMSTPE	DMSTPF	DMSTPG									
SFSTWP	000003	DMSTPE	DMSTPF	DMSTPG									
SFSTYR	000003	DMSTPE	DMSTPF	DMSTPG									
SIGNAL	000057	DMSACM	DMSEDI	DMSERS									
SILI	000339	DMSDBD	DMSDBG	DMSINI	DMSINS	DMSITE	DMSTIO						
SIZE	000022	DMSFRE											
SKEY	000003	DMSFRE											
SKIP	000014	DMSEXE	DMSEXT	DMSINI	DMSROS	DMSSRT	DMSXCP						
SKPSWT	000030	DMSTPE	DMSTPF	DMSTPG									
SOB1	000002	DMSOPT	DMSSET										
SPARES	000033	DMSEDI	DMSEDX	DMSEXE	DMSUPD								
SPEC	000201	DMSLDR	DMSLGT	DMSLIB	DMSOLD	DMSTLB							
SPECLF	000002	DMSINS	DMSINT										
SPIESAV	000002	DMSINT											
SSAVE	000063	DMSABN	DMSDBG	DMSDLB	DMSDOS	DMSERR	DMSFLD	DMSFRE	DMSIFC	DMSITP	DMSITS	DMSLDR	DMSOVS
		DMSSLN	DMSSMN	DMSSOP	DMSSTG	DMSSVN	DMSSVT	DMSSVU	DMSTLB	DMSXMA			
SSAVENXT	000004	DMSITS											
SSAVEPRV	000009	DMSITS	DMSTLB										
SSAVESZ	000006	DMSITS											
STACK	000011	DMSEDI	DMSEXE	DMSHLI	DMSHLL	DMSHLS	DMSXCM	DMSXTB					
STACKAT	000002	DMSEDI											
STACKATL	000005	DMSEDI											
STAEBIT	000003	DMSSAB											
STAESAV	000002	DMSINT											
STAIBIT	000002	DMSSAB											

LABEL	COUNT	REFERENCES											
STARS	000001	DMSINT											
STARS4	000002	DMSHLS											
START	000028	DMSEXE	DMSFET	DMSFNC	DMSFOR	DMSGRN	DMSITS	DMSLDR	DMSLSB	DMSOVS	DMSSRV	DMSTYP	
STATCOPY	000004	DMSARX	DMSASM	DMSHLS									
STATEFST	000042	DMSALU	DMSBRD	DMSDSK	DMSERS	DMSFNS	DMSINT	DMSRNM	DMSSTT	DMSTPE	DMSTPF	DMSTPG	
STATEID	000002	DMSHLI	DMSHLS										
STATERO	000003	DMSBRD	DMSSOP	DMSSTT									
STATER1	000006	DMSBRD	DMSDSK	DMSERS									
STATFM	000010	DMSRNM	DMSTPE	DMSTPF	DMSTPG								
STATFST2	000016	DMSALU	DMSBRD	DMSFNS	DMSRNM	DMSSTT							
STATLST	000013	DMSDLB	DMSFLD	DMSFOR	DMSRNM	DMSTPE	DMSTPF	DMSTPG					
STCKCNT	000003	DMSHLP	DMSHLS										
STCKLEN	000005	DMSHLI	DMSHLS										
STDEVTAB	000003	DMSTPE	DMSTPF	DMSTPG									
STDWCCMD	000002	DMSXSC											
STFILE	000003	DMSTPE	DMSTPF	DMSTPG									
STIMEXIT	000010	DMSITE	DMSSTG	DMSSVN	DMSSVT								
STOPAT	000002	DMSDBG											
STORAGE	000005	DMSHLI	DMSMOD	DMSSLN									
STORMODE	000039	DMSTPE	DMSTPF	DMSTPG									
STRTADDR	000036	DMSFET	DMSHLL	DMSITS	DMSLDR	DMSLOA	DMSLSB	DMSMOD	DMSOLD	DMSSET	DMSSLN	DMSXSG	
STRTNO	000010	DMSEDI	DMSRNE	DMSXCT									
SUBACT	000009	DMSEDX	DMSHLL	DMSINT	DMSLOA	DMSSLN	DMSXCM	DMSXSG	DMSXSS				
SUBFLAG	000034	DMSABN	DMSEDX	DMSEXT	DMSFNS	DMSHLL	DMSINT	DMSLOA	DMSMOD	DMSSLN	DMSXCM	DMSXSG	DMSXSS
SUBINIT	000001	DMSFNS											
SUBR	000027	DMSHLP	DMSLBM	DMSXTB									
SUBREJ	000003	DMSEDX	DMSINT										
SUBRTN	000001	DMSINT											
SUBSECT	000004	DMSABN	DMSINM	DMSINT									
SUOFFSW	000001	DMSHLP											
SUONSW	000001	DMSHLP											
SUSAVE	000002	DMSHLP											
SUSAVE2	000004	DMSHLP											
SUT1A	000001	DMSHLP											
SUT2	000001	DMSHLP											
SVC$202	000016	DMSEXE	DMSEXT										
SVCAB	000011	DMSFRE											
SVCOPSW	000035	DMSBRD	DMSBWR	DMSDOS	DMSITS								
SVCOUNT	000003	DMSOVS											
SVCSECT	000023	DMSCIT	DMSFRE	DMSHDS	DMSINT	DMSITE	DMSLAD	DMSLFS	DMSOVR	DMSOVS	DMSSLN	DMSXMA	
SVC12SAV	000004	DMSDOS											
SVEAIA	000002	DMSBOP	DMSCLS										
SVEARA	000011	DMSBOP	DMSCLS	DMSDOS	DMSITP	DMSVLT							
SVEA0008	000001	DMSVLT											
SVEA0908	000002	DMSBOP	DMSCLS										
SVEPSW	000007	DMSDOS	DMSITP										

LABEL	COUNT	REFERENCES											
SVEPSW2	000008	DMSDOS	DMSITP										
SVEROF	000004	DMSDOS											
SVER00	000015	DMSDOS	DMSITP										
SVER09	000009	DMSDOS	DMSITP										
SVLAD	000006	DMSLAD											
SVLADW	000003	DMSLAD											
SVLFS	000006	DMSLFS											
SWITCH	000044	DMSBOP	DMSCLS	DMSEDI	DMSHLB	DMSHLI	DMSHLP	DMSHLS	DMSOR1				
SWTCH	000013	DMSACM	DMSTLB										
SWTCHSAV	000002	DMSINT											
SYMTABLE	000003	DMSDBG											
SYMTAPA	000027	DMSTPE	DMSTPF	DMSTPG									
SYMTBG	000004	DMSDBG											
SYNABBR	000006	DMSXDC	DMSXIN	DMSXSE									
SYNARGS	000006	DMSXDC	DMSXIN										
SYNARG3	000002	DMSXDC	DMSXIN										
SYNBUFF	000009	DMSXDC	DMSXIN										
SYNFWPTR	000012	DMSXBG	DMSXDC	DMSXIN	DMSXSE								
SYNLSYND	000005	DMSXBG	DMSXDC	DMSXIN									
SYNMAXAR	000001	DMSXDC											
SYNNAME	000007	DMSXDC	DMSXIN	DMSXSE									
SYNNARG	000007	DMSXDC	DMSXIN										
SYNNBTYP	000001	DMSXDC											
SYNOBUFF	000003	DMSXDC	DMSXIN	DMSXSE									
SYNOSYNL	000003	DMSXDC	DMSXIN	DMSXSE									
SYNSUB	000010	DMSXBG	DMSXDC	DMSXIN	DMSXSE								
SYNSYNL	000004	DMSXDC	DMSXIN										
SYSADDR	000003	DMSINI											
SYSCODE	000006	DMSFRE	DMSSET										
SYSCOM	000016	DMSBOP	DMSDOS	DMSETR	DMSFET	DMSQRY	DMSSET	DMSSTG	DMSSYN				
SYSLINE	000003	DMSDLK	DMSQRY	DMSSET									
SYSLOAD	000010	DMSACM	DMSINS	DMSLDR	DMSLSB	DMSOLD	DMSSET						
SYSNAME	000006	DMSBTP	DMSINS										
SYSNAMES	000045	DMSAMS	DMSBOP	DMSBTP	DMSDOS	DMSEDX	DMSEXC	DMSHLL	DMSINS	DMSINT	DMSITS	DMSQRY	DMSSET
		DMSTLA	DMSVIB	DMSXSG									
SYSNCNT	000001	DMSQRY											
SYSNEND	000015	DMSAMS	DMSBOP	DMSBTP	DMSDOS	DMSEDX	DMSEXC	DMSHLL	DMSINS	DMSINT	DMSITS	DMSQRY	DMSSET
		DMSTLA	DMSVIB	DMSXSG									
SYSREF	000004	DMSINS	DMSLOA	DMSSET									
SYSTEMID	000005	DMSINI	DMSINS										
SYSUT1	000028	DMSARX	DMSASM	DMSDLK	DMSLDR	DMSLKD	DMSOLD	DMSUTL					
S17	000003	DMSTPE	DMSTPF	DMSTPG									
S17A	000003	DMSTPE	DMSTPF	DMSTPG									
S202	000041	DMSDSK	DMSLBD	DMSQRY	DMSSOP	DMSTLB	DMSTMA	DMSTPD	DMSTPE	DMSTPF	DMSTPG		
TAB	000005	DMSEDI	DMSHLI	DMSTPE	DMSTPF	DMSTPG							
TABBGN	000002	DMSHLI	DMSHLP										

LABEL	COUNT	REFERENCES								
TABLIN	000016	DMSEDI	DMSSCR							
TABN	000003	DMSTPE	DMSTPF	DMSTPG						
TABS	000025	DMSEDI	DMSEDX							
TAIEIAD	000002	DMSCIT								
TAIEMSGL	000001	DMSCIT								
TAIERSAV	000002	DMSCIT								
TAPE	000034	DMSCLS	DMSLLU	DMSSEB	DMSTIO	DMSTMA	DMSTPE	DMSTPF	DMSTPG	DMSXCP
TAPEBUF	000003	DMSTPE	DMSTPF	DMSTPG						
TAPEBUFF	000004	DMSSEB	DMSSOP							
TAPECCU	000015	DMSTPE	DMSTPF	DMSTPG						
TAPECOUT	000003	DMSSEB	DMSSOP							
TAPEDEV	000005	DMSSBS	DMSSEB	DMSSOP						
TAPELIST	000009	DMSSBS	DMSSEB	DMSSOP						
TAPEMASK	000005	DMSSBS	DMSSEB	DMSSOP						
TAPEOF	000006	DMSTPE	DMSTPF	DMSTPG						
TAPEOPER	000018	DMSSBS	DMSSEB	DMSSOP						
TAPESIZE	000003	DMSSEB	DMSSOP							
TAPE05	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE06	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE07	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE08	000006	DMSTPE	DMSTPF	DMSTPG						
TAPE1	000002	DMSASN								
TAPE10	000090	DMSTPE	DMSTPF	DMSTPG						
TAPE100	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE110	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE120	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE13	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE130	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE140	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE15	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE150	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE151	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE152	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE153	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE154	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE155	000006	DMSTPE	DMSTPF	DMSTPG						
TAPE156	000006	DMSTPE	DMSTPF	DMSTPG						
TAPE157	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE160	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE161	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE163	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE165	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE167	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE170	000006	DMSTPE	DMSTPF	DMSTPG						
TAPE20	000003	DMSTPE	DMSTPF	DMSTPG						
TAPE200	000003	DMSTPE	DMSTPF	DMSTPG						

LABEL	COUNT	REFERENCES
TAPE201	000003	DMSTPE DMSTPF DMSTPG
TAPE23	000003	DMSTPE DMSTPF DMSTPG
TAPE30	000003	DMSTPE DMSTPF DMSTPG
TAPE300	000009	DMSTPE DMSTPF DMSTPG
TAPE301	000003	DMSTPE DMSTPF DMSTPG
TAPE302	000003	DMSTPE DMSTPF DMSTPG
TAPE303	000012	DMSTPE DMSTPF DMSTPG
TAPE4	000002	DMSASN
TAPE40	000003	DMSTPE DMSTPF DMSTPG
TAPE45	000003	DMSTPE DMSTPF DMSTPG
TAPE50	000012	DMSTPE DMSTPF DMSTPG
TAPE60	000003	DMSTPE DMSTPF DMSTPG
TAPE61	000003	DMSTPE DMSTPF DMSTPG
TAPE62	000003	DMSTPE DMSTPF DMSTPG
TAPE65	000003	DMSTPE DMSTPF DMSTPG
TAPE70	000003	DMSTPE DMSTPF DMSTPG
TAPE80	000003	DMSTPE DMSTPF DMSTPG
TAPE90	000003	DMSTPE DMSTPF DMSTPG
TAP1	000006	DMSIFC DMSTPE DMSTPF DMSTPG
TAKEADDR	000010	DMSCIT DMSITE DMSITI DMSSTG DMSSVT
TAKEDEF	000001	DMSSVT
TAKEEXIT	000002	DMSCIT DMSSVT
TAKEEXTS	000001	DMSCIT
TAKEFREQ	000006	DMSCIT DMSITE DMSITI
TAKEIOL	000003	DMSCIT DMSITI
TAKEIOWS	000002	DMSCIT
TAKELNK	000006	DMSCIT DMSITE DMSITI DMSSVT
TAKERTNA	000002	DMSCIT
TAKESTAT	000005	DMSCIT DMSITE DMSITI
TAKETAIE	000002	DMSCIT
TAKETSOF	000002	DMSCIT
TBENT	000030	DMSACM DMSBTB DMSFET DMSGND DMSINS DMSLDR DMSLOA DMSMDP DMSMOD DMSOLD DMSOSR DMSSET DMSSLN
TBLCT	000019	DMSLDR DMSLIB DMSOLD
TBLEND	000003	DMSDBD DMSDBG DMSITE
TBLLNGTH	000005	DMSSBD DMSSVU
TBLREF	000020	DMSLDR DMSLIB DMSOLD
TBUF	000005	DMSHLE DMSHLP DMSHLS
TBUFLGZ	000005	DMSHLE DMSHLS
TCBABPTR	000008	DMSBAB DMSDOS DMSITP
TCBADR	000015	DMSAMS DMSBAB DMSDOS DMSITP DMSSET DMSVSR
TCBFLAGS	000006	DMSDOS DMSVSR
TCBPCPTR	000008	DMSBAB DMSDOS DMSITP
TCBSAVE	000020	DMSAMS DMSBAB DMSDOS DMSITP DMSSET
TCODE	000001	DMSFRE
TEMOPT	000004	DMSOPT

LABEL	COUNT	REFERENCES
TEMPBYTE	000002	DMSSVT
TEMPFILE	000003	DMSTPE DMSTPF DMSTPG
TEMPST	000008	DMSLDR DMSOLD
TEMPTAB	000004	DMSEDI
TESTDDEN	000003	DMSTPE DMSTPF DMSTPG
TESTMOD2	000003	DMSTPE DMSTPF DMSTPG
TESTMOD3	000006	DMSTPE DMSTPF DMSTPG
TESTOC	000003	DMSTPE DMSTPF DMSTPG
TEST2420	000003	DMSTPE DMSTPF DMSTPG
TEST3420	000003	DMSTPE DMSTPF DMSTPG
TEST6250	000003	DMSTPE DMSTPF DMSTPG
TEST8809	000003	DMSTPE DMSTPF DMSTPG
TEST9TRK	000003	DMSTPE DMSTPF DMSTPG
TIC	000075	DMSINI
TID	000001	DMSDOS
TIMBUF	000015	DMSEXT DMSINM DMSSVT
TIMCCW	000005	DMSITE DMSQRY DMSSET
TIMCHAR	000023	DMSINS DMSINT DMSIOW DMSITE DMSQRY DMSSET DMSSMN DMSSTG DMSSVN DMSSVT
TIMER	000018	DMSINS DMSINT DMSIOW DMSITE DMSSET DMSSVN DMSSVT
TIMINIT	000012	DMSINS DMSINT DMSIOW DMSITE DMSSET DMSSVN
TIN	000004	DMSEDI DMSEDX
TLBBLKCT	000001	DMSTLB
TLBBLOK	000010	DMSBOP DMSCLS DMSSEB DMSSOP DMSTLB DMSTMA DMSTPD
TLBCALL	000036	DMSBOP DMSCLS DMSSEB DMSSOP DMSTLB
TLBCLIN	000004	DMSCLS DMSSOP DMSTLB
TLBCLOUT	000002	DMSCLS DMSSOP
TLBCMAC	000006	DMSTLB
TLBCMS	000003	DMSTLB DMSTMA DMSTPD
TLBDOS	000015	DMSBOP DMSCLS DMSTLB
TLBDTFPT	000004	DMSBOP DMSCLS DMSTLB
TLBDWSZ	000016	DMSBOP DMSCLS DMSSEB DMSSOP
TLBEOV	000003	DMSCLS DMSSEB DMSTLB
TLBFCBPT	000006	DMSSEB DMSSOP DMSTLB
TLBLABID	000007	DMSTLB DMSTMA DMSTPD
TLBLABT	000023	DMSBOP DMSCLS DMSSEB DMSSOP DMSTLB DMSTMA DMSTPD
TLBMODE	000007	DMSBOP DMSCLS DMSSEB DMSSOP DMSTLB
TLBMSPC	000005	DMSTLB
TLBNAME	000005	DMSBOP DMSCLS DMSSEB DMSSOP
TLBNSL	000003	DMSTLB DMSTMA DMSTPD
TLBNSLMD	000003	DMSTLB DMSTMA DMSTPD
TLBNSLNM	000007	DMSSEB DMSSOP DMSTLB DMSTMA DMSTPD
TLBOPIN	000007	DMSBOP DMSSOP DMSTLB DMSTMA DMSTPD
TLBOPOUT	000005	DMSBOP DMSSOP DMSTLB
TLBOS	000015	DMSSEB DMSSOP DMSTLB
TLBSL	000005	DMSBOP DMSCLS DMSTLB DMSTMA DMSTPD
TLBSUL	000004	DMSTLB

LABEL	COUNT	PEFERENCES						
TLBTAPID	000014	DMSBOP	DMSCLS	DMSSEB	DMSSOP	DMSTLB	DMSTMA	DMSTPD
TLBTYPE	000020	DMSBOP	DMSCLS	DMSSEB	DMSSOP	DMSTLB		
TMARK	000003	DMSTPE	DMSTPF	DMSTPG				
TMPLOC	000008	DMSLDR	DMSLSB	DMSOLD				
TOOBIG	000003	DMSDIO						
TOTLIBS	000003	DMSGLB	DMSSMN					
TOUT	000004	DMSEDI						
TPCONTL	000024	DMSTPE	DMSTPF	DMSTPG				
TPDUMP	000006	DMSTPE	DMSTPF	DMSTPG				
TPDUMP10	000012	DMSTPE	DMSTPF	DMSTPG				
TPDVOLMV	000003	DMSTPE	DMSTPF	DMSTPG				
TPDVOL1	000003	DMSTPE	DMSTPF	DMSTPG				
TPEDIS	000051	DMSTPE	DMSTPF	DMSTPG				
TPEENA	000051	DMSTPE	DMSTPF	DMSTPG				
TPEFLG	000090	DMSTPE	DMSTPF	DMSTPG				
TPEND	000003	DMSTPE	DMSTPF	DMSTPG				
TPFACB	000004	DMSSOP						
TPFERT	000006	DMSITS						
TPFNS	000009	DMSITS						
TPFR01	000002	DMSITS						
TPFSVO	000005	DMSDOS	DMSITS	DMSOVS				
TPFUSR	000012	DMSDBG	DMSITP	DMSITS	DMSLDR			
TPINIT	000003	DMSTPE	DMSTPF	DMSTPG				
TPLD0	000015	DMSTPE	DMSTPF	DMSTPG				
TPLD1	000009	DMSTPE	DMSTPF	DMSTPG				
TPLOAD	000006	DMSTPE	DMSTPF	DMSTPG				
TPMODEST	000003	DMSTPE	DMSTPF	DMSTPG				
TPSBSR	000003	DMSTPE	DMSTPF	DMSTPG				
TPSCAN	000006	DMSTPE	DMSTPF	DMSTPG				
TPSKIP	000003	DMSTPE	DMSTPF	DMSTPG				
TPSLOOP	000003	DMSTPE	DMSTPF	DMSTPG				
TPSRSET	000021	DMSTPE	DMSTPF	DMSTPG				
TPVOLEND	000003	DMSTPE	DMSTPF	DMSTPG				
TPVOLHDR	000003	DMSTPE	DMSTPF	DMSTPG				
TPVOLREW	000006	DMSTPE	DMSTPF	DMSTPG				
TPWVOLCK	000003	DMSTPE	DMSTPF	DMSTPG				
TPWVOL1	000006	DMSTPE	DMSTPF	DMSTPG				
TRACK7	000004	DMSASN	DMSBOP					
TRACK9	000003	DMSTPE	DMSTPF	DMSTPG				
TRANTAB	000007	DMSHLP	DMSVLT					
TRKLSAVE	000002	DMSTQQ						
TRNCNUM	000006	DMSEDI						
TRNCODE	000001	DMSFRE						
TRSWS	000004	DMSHLP						
TETCHE	000003	DMSTPE	DMSTPF	DMSTPG				
TETCHET	000012	DMSTPE	DMSTPF	DMSTPG				

LABEL	COUNT	REFERENCES											
TRTCHO	000003	DMSTPE	DMSTPF	DMSTPG									
TRTCHOC	000012	DMSTPE	DMSTPF	DMSTPG									
TRTCHOT	000003	DMSTPE	DMSTPF	DMSTPG									
TRTVOLID	000007	DMSFLD	DMSLBD	DMSTPE	DMSTPF	DMSTPG							
TRUNCOL	000021	DMSEDI	DMSEDX	DMSEXE	DMSSCR								
TSCB	000001	DMSHLP											
TSCBLENG	000001	DMSHLP											
TSOATCNL	000018	DMSCIT	DMSCRD	DMSITE	DMSITI	DMSITS	DMSSEB	DMSSVN					
TSOFLAGS	000018	DMSCIT	DMSCRD	DMSITE	DMSITI	DMSITS	DMSSEB	DMSSVN					
TST34208	000009	DMSTPE	DMSTPF	DMSTPG									
TSYM	000005	DMSDBG											
TTRANS	000003	DMSTPE	DMSTPF	DMSTPG									
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							
TYPCHAIN	000129	DMSXTB											
TYPCHDEL	000004	DMSXTB											
TYPE	000154	DMSACC	DMSACF	DMSACM	DMSAUD	DMSBOP	DMSBRD	DMSBTB	DMSBWR	DMSCAT	DMSCLS	DMSCMP	DMSCPY
		DMSCRD	DMSDIO	DMSDLK	DMSDMP	DMSDSK	DMSDSV	DMSEDI	DMSEDX	DMSERD	DMSEXC	DMSEXE	DMSEXT
		DMSFLD	DMSFNS	DMSFOR	DMSFRE	DMSIFC	DMSINA	DMSINS	DMSINT	DMSITE	DMSITP	DMSITS	DMSLAD
		DMSLAF	DMSLBD	DMSLFS	DMSLGT	DMSLIB	DMSLIO	DMSLOA	DMSLOS	DMSLSB	DMSLST	DMSOPL	DMSOR1
		DMSOVR	DMSOVS	DMSRNE	DMSROS	DMSSAB	DMSSCR	DMSSEB	DMSSET	DMSSOP	DMSSVT	DMSSYN	DMSTLB
		DMSTRK	DMSUPD	DMSVIB	DMSVIP	DMSXBG	DMSXED	DMSXMA	DMSXSD	DMSXSE	DMSXTB	DMSXUP	DMSZAP
TYPEAD	000001	DMSLIO											
TYPFLAG	000039	DMSDBG	DMSDOS	DMSITP	DMSITS	DMSLDR	DMSOVS	DMSSOP					
TYPFLG	000002	DMSEDI											
TYPLIGNE	000038	DMSXTB											
TYPLIN	000045	DMSEXE	DMSEXT	DMSFNC	DMSLBT	DMSLIO	DMSTYP						
TYPLIST	000007	DMSITE											
TYPNUM	000059	DMSXTB											
TYPSCR	000009	DMSEDX	DMSSCR										
TYPTRGT	000024	DMSXTB											
TYPTRGTC	000004	DMSXTB											
TYP2401	000004	DMSASN	DMSTPE	DMSTPF	DMSTPG								
TYP2415	000001	DMSASN											
TYP2420	000004	DMSASN	DMSTPE	DMSTPF	DMSTPG								
TYP3420	000007	DMSASN	DMSTPE	DMSTPF	DMSTPG								
TYP8809	000010	DMSASN	DMSTIO	DMSTMA	DMSTPD	DMSTPE	DMSTPF	DMSTPG					
UE	000001	DMSCIT											
UFDBUSY	000059	DMSABN	DMSACC	DMSACF	DMSACM	DMSAUD	DMSBTP	DMSBWR	DMSCIT	DMSDIO	DMSDOS	DMSDSK	DMSERD
		DMSERS	DMSFNS	DMSITE	DMSITI	DMSITP	DMSITS	DMSRNM	DMSTPE	DMSTPF	DMSTPG		
UND	000021	DMSBWR	DMSLOS	DMSROS	DMSSBS	DMSSEB	DMSSOP	DMSSQS					
UNDL	000006	DMSHLP											

LABEL	COUNT	REFERENCES
UNPACK	000013	DMSCPY DMSEXT DMSLIO
UNRES	000005	DMSLDR DMSLOA DMSOLD
UPBIT	000006	DMSACM DMSAUD DMSDSK
UPSI	000004	DMSSET
UP1BLK	000006	DMSTPE DMSTPF DMSTPG
USARCODE	000002	DMSFRE
USAVE	000003	DMSITS
USAVEPTR	000029	DMSITS DMSSLN DMSSOP DMSSTG DMSSVT DMSTLB
USAVESZ	000005	DMSITS
USERCODE	000004	DMSFRE DMSSET
USERKEY	000012	DMSABN DMSFRE DMSSET
USRREGSV	000005	DMSBOP DMSCLS
USRRGLEN	000003	DMSBOP DMSCLS DMSVLT
UTILFLAG	000022	DMSEDI DMSSCR
VAR	000036	DMSEXE DMSOR1 DMSROS DMSSBD DMSSBS DMSSEB DMSSOP DMSSQS DMSSVT DMSSVU DMSTPD
VBLOCK	000011	DMSDSK DMSERD DMSTPE DMSTPF DMSTPG
VBOX	000001	DMSHLB
VCADTLKP	000033	DMSACC DMSACM DMSALU DMSARE DMSASN DMSBOP DMSDIO DMSDLB DMSDSL DMSERD DMSERS DMSEXT DMSINS DMSLAB DMSLDS DMSLLU DMSLST DMSQRY DMSRNM DMSSET DMSSVT DMSSVU DMSUPD DMSXCP DMSZES
VCADTLKW	000008	DMSAMS DMSARN DMSEXT DMSRNE DMSSRT DMSUPD DMSXRE
VCADTNXT	000010	DMSABN DMSACC DMSALU DMSARE DMSLDS DMSLST DMSQRY DMSROS
VCFSTLKP	000009	DMSACC DMSDSK DMSEDX DMSERD DMSERS DMSPNT DMSTPE DMSTPF DMSTPG
VCFSTLKW	000011	DMSERD DMSRNM DMSSTT DMSTPE DMSTPF DMSTPG
VERCOL1	000009	DMSEDI DMSEDX DMSSCR
VERCOL2	000004	DMSEDI DMSEDX
VERLEN	000007	DMSEDI DMSEDX DMSSCR
VFORMAT	000024	DMSERD DMSTPE DMSTPF DMSTPG
VIPINIT	000009	DMSCLS DMSDOS DMSEXT DMSINT DMSSTG DMSVIP DMSVSR
VIPSOP	000008	DMSBOP DMSCLS DMSVIP
VIPTCLOS	000004	DMSCLS DMSVIP
VMSIZE	000051	DMSAMS DMSBOP DMSBRD DMSBWR DMSDBG DMSDOS DMSFRE DMSHDI DMSHDS DMSINS DMSLDR DMSOVS DMSPRE DMSSET DMSSSK DMSSVT DMSVIB DMSZES
VOLERR	000003	DMSTPE DMSTPF DMSTPG
VRULE	000005	DMSHLP
VRULEREM	000006	DMSHLB DMSHLP
VSAMFLG1	000051	DMSABN DMSAMS DMSBAB DMSBOP DMSCLS DMSDLB DMSDOS DMSEXT DMSFCH DMSINT DMSITP DMSSTG DMSVIB DMSVIP DMSVSR
VSAMOPEN	000003	DMSBOP DMSDOS
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
VSJOBCAT	000003	DMSDLB
VSMINSTL	000005	DMSFCH DMSFET
VSTRANGE	000001	DMSITI
WAIT	000030	DMSCIT DMSINI DMSINS DMSITE

LABEL	COUNT	REFERENCES											
WAITLIST	000002	DMSDBG	DMSSVT										
WAITLST	000003	DMSCRD	DMSCWR	DMSCWT									
WAITRD	000009	DMSDBG	DMSEXE	DMSFNC	DMSFOR								
WAITSAVE	000007	DMSCIT	DMSDBG	DMSIOW									
WORKFILE	000004	DMSOLD											
WRBIT	000017	DMSACC	DMSBWR	DMSDSK	DMSERD	DMSTPE	DMSTPF	DMSTPG					
WRBUF	000008	DMSFNC	DMSSRT	DMSTPE	DMSTPF	DMSTPG	DMSXCP						
WRBUFF	000009	DMSTPE	DMSTPF	DMSTPG									
WRDATA	000038	DMSINI											
WRFILE	000024	DMSTPE	DMSTPF	DMSTPG									
WRFV	000019	DMSCPY	DMSTPE	DMSTPF	DMSTPG								
WRITE	000051	DMSBOP	DMSCLS	DMSDIO	DMSDLK	DMSDSL	DMSEXE	DMSINI	DMSQRY	DMSSBS	DMSTLB	DMSTPE	DMSTPF
		DMSTPG											
WRITEOUT	000009	DMSTPE	DMSTPF	DMSTPG									
WRITERTN	000003	DMSTPE	DMSTPF	DMSTPG									
WRITE1	000007	DMSINI											
WRITNO	000027	DMSTPE	DMSTPF	DMSTPG									
WRMODE	000019	DMSSRT	DMSTPE	DMSTPF	DMSTPG								
WRNAME	000009	DMSTPE	DMSTPF	DMSTPG									
WRNOIT	000018	DMSTPE	DMSTPF	DMSTPG									
WRSIZE	000013	DMSSRT	DMSTPE	DMSTPF	DMSTPG								
WRTKF	000003	DMSDIO											
WRTYPE	000052	DMSDSK	DMSEDI	DMSEDX									
WTM	000028	DMSBOP	DMSCLS	DMSSEB	DMSTLB	DMSTPE	DMSTPF	DMSTPG					
WTM2	000003	DMSTPE	DMSTPF	DMSTPG									
WTRDCNT	000002	DMSDBG											
WVOL1	000006	DMSTPE	DMSTPF	DMSTPG									
XAREA	000001	DMSEDI											
XCOUNT	000002	DMSOVS											
XFF	000066	DMSAUD	DMSCIT	DMSDOS	DMSDSK	DMSERD	DMSFLD	DMSFOR	DMSITS	DMSLBD	DMSLCK	DMSSRV	DMSTMA
		DMSTPD	DMSTPE	DMSTPF	DMSTPG	DMSXED	DMSXER	DMSXIN	DMSXMS				
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	000002	DMSINI											
YEXTS	000001	DMSHLS											
YSRCH	000002	DMSHLS											
YYDDD	000003	DMSINS											
ZDEABPNG	000011	DMSXPX	DMSXSD	DMSXSU									
ZDEACURD	000054	DMSXCG	DMSXCT	DMSXFC	DMSXFD	DMSXIN	DMSXMD	DMSXML	DMSXMS	DMSXPX	DMSXSD	DMSXSE	DMSXUP

LABEL	COUNT	REFERENCES											
ZDEACURL	000053	DMSXCG	DMSXCM	DMSXDS	DMSXFC	DMSXFD	DMSXGT	DMSXIN	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXSD
		DMSXSE	DMSXSU	DMSXUP									
ZDEARBCH	000009	DMSXCG	DMSXFD	DMSXIN	DMSXSE	DMSXSU							
ZDEARBON	000007	DMSXCG	DMSXFD	DMSXSE	DMSXSU								
ZDEATCNT	000012	DMSXIN	DMSXMD	DMSXSC	DMSXSE	DMSXSU							
ZDEATSID	000010	DMSXCT	DMSXMD	DMSXSC	DMSXSE	DMSXSU							
ZDEATSMD	000008	DMSXCT	DMSXMD	DMSXSC	DMSXSE	DMSXSU							
ZDEBWPTR	000024	DMSXBG	DMSXCT	DMSXED									
ZDECANON	000009	DMSXCG	DMSXFC	DMSXFD	DMSXIN	DMSXSD	DMSXSE						
ZDECASMU	000007	DMSXIO	DMSXPO	DMSXPX	DMSXSD	DMSXSE							
ZDECASRI	000007	DMSXFD	DMSXIN	DMSXMS	DMSXSE								
ZDECESCA	000002	DMSXMD	DMSXSE										
ZDECFILL	000009	DMSXCG	DMSXER	DMSXFC	DMSXIN	DMSXSE							
ZDECGCNT	000064	DMSXCG	DMSXCT	DMSXER	DMSXFD	DMSXGT	DMSXIN	DMSXMD	DMSXMS	DMSXPO	DMSXPX	DMSXSC	DMSXSD
		DMSXSE	DMSXSU										
ZDECLNSC	000006	DMSXPX	DMSXSC	DMSXSD	DMSXSU								
ZDECLPON	000004	DMSXFC	DMSXIN	DMSXSE									
ZDECLTGT	000014	DMSXCG	DMSXDC	DMSXFD	DMSXMC								
ZDECLTG1	000001	DMSXFD											
ZDECLTG2	000001	DMSXFD											
ZDECMSON	000006	DMSXDC	DMSXIN	DMSXSE									
ZDECNFCT	000001	DMSXPX											
ZDECSCOL	000019	DMSXCT	DMSXFC	DMSXIN	DMSXMD	DMSXPX	DMSXSC	DMSXSE					
ZDECSERR	000005	DMSXPX	DMSXSC										
ZDECSINP	000012	DMSXPX	DMSXSC	DMSXSD									
ZDECSLIN	000027	DMSXCT	DMSXFC	DMSXIN	DMSXMD	DMSXPX	DMSXSC	DMSXSD	DMSXSE	DMSXSS			
ZDECSRST	000015	DMSXCT	DMSXED	DMSXMD	DMSXSC	DMSXSD	DMSXSE	DMSXSS					
ZDECSSCX	000033	DMSXCT	DMSXFC	DMSXPX	DMSXSC	DMSXSE	DMSXSS	DMSXSU					
ZDECSSCY	000028	DMSXCT	DMSXFC	DMSXPX	DMSXSC	DMSXSE	DMSXSS	DMSXSU					
ZDECSSET	000012	DMSXCT	DMSXMD	DMSXSC	DMSXSD	DMSXSS	DMSXSU						
ZDECTLON	000006	DMSXFD	DMSXIN	DMSXUP									
ZDECURCL	000071	DMSXCG	DMSXCT	DMSXFC	DMSXFD	DMSXIN	DMSXMC	DMSXSD					
ZDECURLN	000104	DMSXCG	DMSXCT	DMSXDC	DMSXFC	DMSXFD	DMSXIN	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPX	DMSXSC
		DMSXSD	DMSXSE	DMSXSS	DMSXSU	DMSXUP							
ZDEDELPT	000013	DMSXED	DMSXSU	DMSXUP									
ZDEENDRG	000030	DMSXCG	DMSXFC	DMSXFD	DMSXIN	DMSXML	DMSXMS	DMSXSC	DMSXSD	DMSXSU	DMSXUP		
ZDEEQBFL	000012	DMSXCT	DMSXDC	DMSXSE	DMSXSU								
ZDEEQBUF	000003	DMSXDC	DMSXSE	DMSXSU									
ZDEESCON	000004	DMSXIN	DMSXMD	DMSXSE									
ZDEFCURL	000005	DMSXIN	DMSXSE	DMSXSU									
ZDEFINPU	000004	DMSXIN	DMSXSE	DMSXSU									
ZDEFLABL	000002	DMSXPX											
ZDEFLAG1	000154	DMSXCG	DMSXCM	DMSXCT	DMSXFC	DMSXFD	DMSXGT	DMSXIN	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXPX
		DMSXSC	DMSXSD	DMSXSU	DMSXUP								
ZDEFLAG2	000056	DMSXCG	DMSXCT	DMSXDC	DMSXFC	DMSXFD	DMSXIN	DMSXIO	DMSXMC	DMSXML	DMSXPX	DMSXSD	DMSXSE
		DMSXSS	DMSXSU	DMSXUP									

LABEL	COUNT	REFERENCES											
ZDEFLAG3	000059	DMSXCG	DMSXDC	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXIN	DMSXMD	DMSXPO	DMSXSD	DMSXSE	DMSXSU
		DMSXUP											
ZDEFLAG4	000048	DMSXCG	DMSXCT	DMSXDC	DMSXDS	DMSXER	DMSXFD	DMSXIN	DMSXIO	DMSXSD	DMSXSE	DMSXSU	
ZDEFLAG5	000083	DMSXFC	DMSXIN	DMSXMD	DMSXPX	DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXSU			
ZDEFLDCL	000003	DMSXSC	DMSXSD										
ZDEFLDLN	000011	DMSXPX	DMSXSC	DMSXSD	DMSXSS								
ZDEFLEOF	000080	DMSXCG	DMSXCM	DMSXCT	DMSXFC	DMSXFD	DMSXGT	DMSXIN	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXPX
		DMSXSE	DMSXSU	DMSXUP									
ZDEFLLEF	000008	DMSXFC	DMSXFD										
ZDEFLNBL	000006	DMSXPX											
ZDEFLPRX	000010	DMSXCT	DMSXIN	DMSXPX	DMSXSU								
ZDEFLRIG	000028	DMSXCG	DMSXFC	DMSXFD	DMSXMD								
ZDEFLSIZ	000047	DMSXBG	DMSXCG	DMSXCT	DMSXED	DMSXFC	DMSXFD	DMSXIN	DMSXMD	DMSXML	DMSXPT	DMSXPX	DMSXRE
		DMSXSD	DMSXSE	DMSXSU	DMSXUP								
ZDEFLTOF	000069	DMSXCG	DMSXCM	DMSXCT	DMSXFC	DMSXFD	DMSXIN	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXPX	DMSXSE
		DMSXSU											
ZDEFMASK	000004	DMSXSE	DMSXSU										
ZDEFMODE	000019	DMSXCT	DMSXED	DMSXER	DMSXIN	DMSXPT	DMSXSE	DMSXUP					
ZDEFNAME	000039	DMSXCT	DMSXDS	DMSXED	DMSXER	DMSXGT	DMSXIN	DMSXPT	DMSXSE	DMSXSU	DMSXUP		
ZDEFSINP	000008	DMSXIN	DMSXMD	DMSXSC	DMSXSD								
ZDEFTABS	000004	DMSXSE	DMSXSU										
ZDEFTYPE	000016	DMSXCT	DMSXED	DMSXER	DMSXGT	DMSXIN	DMSXPT	DMSXSE	DMSXUP				
ZDEFWPTR	000030	DMSXBG	DMSXCT	DMSXED	DMSXSE								
ZDEHEXON	000004	DMSXDC	DMSXFD	DMSXSE									
ZDEIMGON	000008	DMSXFC	DMSXIN	DMSXSD	DMSXSE								
ZDEINCPL	000004	DMSXPX	DMSXSD	DMSXSU									
ZDEINHLD	000007	DMSXMD	DMSXSC	DMSXSD									
ZDEINVCM	000001	DMSXPX											
ZDELLABL	000002	DMSXPX											
ZDELLNBL	000003	DMSXPX											
ZDELLPRX	000020	DMSXCT	DMSXIN	DMSXPX	DMSXSU								
ZDELNDON	000005	DMSXIN	DMSXPO	DMSXSE	DMSXSU								
ZDELNEND	000004	DMSXIN	DMSXPO	DMSXSE	DMSXSU								
ZDELNINV	000002	DMSXPX	DMSXSC										
ZDELRECL	000084	DMSXCG	DMSXCT	DMSXDS	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXIN	DMSXMC	DMSXMS	DMSXPT	DMSXPX
		DMSXRE	DMSXSD	DMSXSE	DMSXSU	DMSXUP							
ZDELSCPT	000053	DMSXBG	DMSXCT	DMSXED	DMSXFC	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPX
		DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXSU							
ZDELSTCG	000002	DMSXMD	DMSXSC										
ZDELZDEB	000006	DMSXBG	DMSXCT	DMSXED									
ZDELZDED	000008	DMSXBG	DMSXCT	DMSXED									
ZDEMCPNG	000010	DMSXPX	DMSXSD	DMSXSU									
ZDEMCRON	000005	DMSXDC	DMSXSE										
ZDEMRGUP	000004	DMSXIN	DMSXUP										
ZDEMSBFL	000002	DMSXIO	DMSXSE										
ZDEMSBUF	000004	DMSXIO	DMSXSE	DMSXSU									

LABEL	COUNT	REFERENCES
ZDEMSGMD	000018	DMSXCT DMSXDS DMSXIN DMSXIO DMSXSE DMSXSU
ZDEMSKLN	000012	DMSXCT DMSXIN DMSXMD DMSXPX DMSXSE
ZDENBTBC	000015	DMSXCG DMSXCT DMSXFC DMSXSC DMSXSD DMSXSE DMSXSS
ZDENBVRC	000006	DMSXMC DMSXSD DMSXSE
ZDENSPAN	000009	DMSXCT DMSXFD DMSXIN DMSXSE
ZDENULON	000004	DMSXSC DMSXSD DMSXSE
ZDENUMON	000006	DMSXPX DMSXSD DMSXSE DMSXSS
ZDEOSDSN	000003	DMSXIN
ZDEPCKON	000008	DMSXER DMSXFD DMSXIN DMSXSE
ZDEPFCOD	000007	DMSXMD DMSXSC
ZDEPFKEY	000011	DMSXMD DMSXSC
ZDEPFKPT	000013	DMSXBG DMSXED DMSXIN DMSXMD DMSXSC DMSXSE DMSXSS DMSXSU
ZDEPRFEX	000008	DMSXPX DMSXSC DMSXSS
ZDEPRFON	000029	DMSXFC DMSXIN DMSXMD DMSXSC DMSXSD DMSXSE DMSXSS
ZDEPRFRG	000032	DMSXFC DMSXIN DMSXMD DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS
ZDEPRFW1	000006	DMSXSC
ZDEPRSPT	000014	DMSXBG DMSXCT DMSXED
ZDEQMBUF	000019	DMSXBG DMSXCT DMSXED DMSXSU
ZDEQMPTR	000008	DMSXBG DMSXCT DMSXED DMSXSU
ZDERDALL	000005	DMSXCT DMSXSD
ZDERDINP	000003	DMSXCT DMSXSD
ZDERDNCH	000005	DMSXCT DMSXSD
ZDERDNUM	000005	DMSXCT DMSXSD
ZDERECFM	000026	DMSXER DMSXFD DMSXIN DMSXPT DMSXSE DMSXUP
ZDESBCOM	000015	DMSXCT DMSXDC DMSXED DMSXIN DMSXMD DMSXPX DMSXSC DMSXSD DMSXSU
ZDESC	000211	DMSXBG DMSXCG DMSXCM DMSXCN DMSXCT DMSXDC DMSXDS DMSXED DMSXER DMSXFC DMSXFD DMSXGT
		DMSXHL DMSXIN DMSXIO DMSXMA DMSXMC DMSXMD DMSXML DMSXMS DMSXPO DMSXPT DMSXPX DMSXRE
		DMSXSC DMSXSD DMSXSE DMSXSS DMSXST DMSXSU DMSXUP
ZDESCABV	000022	DMSXCT DMSXFC DMSXFD DMSXIN DMSXML DMSXSC DMSXSE
ZDESCBLW	000021	DMSXCT DMSXFC DMSXFD DMSXIN DMSXML DMSXSC DMSXSE
ZDESCHGD	000011	DMSXMD DMSXPX DMSXSC DMSXSD DMSXSS
ZDESCLON	000007	DMSXIN DMSXPX DMSXSD DMSXSE
ZDESERCH	000011	DMSXFD DMSXIN DMSXSE DMSXUP
ZDESERIN	000011	DMSXCT DMSXFD DMSXIN DMSXSE DMSXUP
ZDESERLG	000012	DMSXFD DMSXIN DMSXSE DMSXUP
ZDESERST	000005	DMSXFD DMSXIN DMSXSE
ZDESFLG1	000011	DMSXCT DMSXSD
ZDESFLG2	000062	DMSXCT DMSXDC DMSXED DMSXFC DMSXFD DMSXIN DMSXIO DMSXMA DMSXMD DMSXML DMSXPO DMSXPX
		DMSXSC DMSXSD DMSXSE DMSXSS DMSXSU
ZDESFLG3	000043	DMSXCT DMSXED DMSXMD DMSXSC DMSXSD DMSXSE DMSXSS DMSXSU
ZDESFLG4	000048	DMSXCT DMSXPX DMSXSC DMSXSD DMSXSS DMSXSU
ZDESHFLG	000003	DMSXFC DMSXML
ZDESHMSG	000005	DMSXER DMSXSE
ZDESIDCD	000004	DMSXIN DMSXUP
ZDESIDON	000002	DMSXIN DMSXUP
ZDESPABN	000006	DMSXFD DMSXIN DMSXSE

LABEL	COUNT	REFERENCES
ZDESPNON	000006	DMSXFD DMSXSE
ZDESQ8ON	000019	DMSXIN DMSXSE DMSXUP
ZDESRPNG	000005	DMSXUP
ZDESTMON	000005	DMSXFD DMSXIN DMSXSE
ZDESTYLN	000014	DMSXCG DMSXCT DMSXDC DMSXFD DMSXML DMSXSU
ZDESYNON	000005	DMSXDC DMSXIN DMSXPX DMSXSE
ZDETABCL	000019	DMSXCG DMSXCT DMSXFC DMSXIN DMSXMD DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS
ZDETAYON	000006	DMSXCG DMSXDC DMSXSE DMSXSU
ZDETBSON	000006	DMSXPX DMSXSD DMSXSE
ZDETFLIN	000004	DMSXIN DMSXSD DMSXSE
ZDETODSP	000015	DMSXSC DMSXSS
ZDETOPPT	000036	DMSXBG DMSXCG DMSXCT DMSXED DMSXFC DMSXFD DMSXIN DMSXML DMSXMS DMSXPX DMSXRE DMSXSE DMSXSU DMSXUP
ZDETOPRG	000021	DMSXCG DMSXFC DMSXFD DMSXIN DMSXML DMSXMS DMSXSC DMSXSD DMSXSU
ZDETRUNC	000071	DMSXCG DMSXCM DMSXDS DMSXER DMSXFC DMSXFD DMSXGT DMSXIN DMSXPO DMSXPX DMSXSD DMSXSE DMSXSU DMSXUP
ZDEUPDON	000020	DMSXCG DMSXED DMSXFC DMSXFD DMSXIN DMSXMD DMSXSD DMSXSE
ZDEUPINC	000004	DMSXIN DMSXUP
ZDEVERCL	000008	DMSXCT DMSXMC DMSXSD DMSXSE DMSXSU
ZDEVERC1	000022	DMSXCT DMSXFC DMSXIN DMSXPX DMSXSC DMSXSD DMSXSS DMSXSU
ZDEVERC2	000006	DMSXIN DMSXPX
ZDEVERON	000013	DMSXCG DMSXCT DMSXIN DMSXIO DMSXMC DMSXSE DMSXSU
ZDEVERTR	000025	DMSXCT DMSXFC DMSXMC DMSXSC DMSXSD DMSXSS DMSXSU
ZDEVRBON	000008	DMSXFD DMSXSE
ZDEWIDTH	000033	DMSXCG DMSXCN DMSXDS DMSXER DMSXFC DMSXIN DMSXSE DMSXST
ZDEWRMSG	000014	DMSXIO DMSXMA DMSXPO DMSXSC DMSXSD DMSXSU
ZDEWRPON	000006	DMSXDC DMSXFD DMSXML DMSXSE
ZDEZDEPT	000043	DMSXBG DMSXCT DMSXED DMSXFC DMSXPX DMSXSC DMSXSU
ZDEZONEL	000038	DMSXCG DMSXCT DMSXFC DMSXFD DMSXIN DMSXMC DMSXSC DMSXSD
ZDEZONER	000033	DMSXCG DMSXDS DMSXFC DMSXFD DMSXIN DMSXMC DMSXSC DMSXSD
ZDE2INPT	000019	DMSXFC DMSXIN DMSXSC DMSXSD DMSXSE DMSXSU
ZFOABUFF	000034	DMSXED DMSXFD DMSXGT DMSXIN DMSXMA DMSXPT DMSXSU DMSXUP
ZFOAITNO	000020	DMSXFD DMSXGT DMSXIN DMSXMA DMSXPT DMSXUP
ZFOALARM	000012	DMSXCT DMSXER DMSXIO DMSXMD DMSXPO DMSXSC DMSXSS
ZFOANOIT	000011	DMSXFD DMSXGT DMSXIN DMSXMA DMSXPT DMSXUP
ZFOAPLON	000008	DMSXBG DMSXSE
ZFOATERM	000005	DMSXBG DMSXPO DMSXSC
ZFOATSID	000002	DMSXSU
ZFOBLKCR	000006	DMSXIN DMSXST
ZFOBLKPT	000008	DMSXBG DMSXIN DMSXPT DMSXST
ZFOBUFIM	000015	DMSXBG DMSXPX DMSXSC DMSXSD DMSXSS
ZFOBUFIO	000019	DMSXBG DMSXPO DMSXSC DMSXSD
ZFOCLRLG	000040	DMSXCM DMSXCT DMSXED DMSXFC DMSXHL DMSXMC DMSXMD DMSXMS DMSXPO DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS DMSXSU
ZFOCLRSC	000017	DMSXBG DMSXCM DMSXCT DMSXIO DMSXSC DMSXSD DMSXSS
ZFOCMBLK	000023	DMSXCT DMSXPX DMSXSD DMSXSS

LABEL	COUNT	REFERENCES
ZFOCMCNT	000029	DMSXCT DMSXPX DMSXSD DMSXSS
ZFOCMPNG	000033	DMSXCT DMSXPX DMSXSD DMSXSS
ZFOCSDPT	000004	DMSXSS
ZFOCSRAD	000025	DMSXCT DMSXSC DMSXSS
ZFOCSRFL	000017	DMSXBG DMSXCT DMSXED DMSXMD DMSXSC DMSXSS
ZFOCSRSZ	000016	DMSXBG DMSXSE DMSXSS
ZFOCSSTK	000007	DMSXBG DMSXSS
ZFOCTLSZ	000001	DMSXSD
ZFOCURFL	000015	DMSXBG DMSXCT DMSXED DMSXIN DMSXMA DMSXMS DMSXSS
ZFOCURSV	000010	DMSXCM DMSXMA
ZFOC2741	000003	DMSXBG DMSXFC DMSXSE
ZFOC3215	000002	DMSXBG DMSXSE
ZFOC3270	000049	DMSXBG DMSXCG DMSXCT DMSXED DMSXFC DMSXIN DMSXIO DMSXMC DMSXMD DMSXML DMSXPO DMSXSC DMSXSE DMSXSU
ZFOC3278	000006	DMSXBG DMSXSC DMSXSD DMSXSE
ZFOEDGON	000038	DMSXBG DMSXCG DMSXCT DMSXDC DMSXIN DMSXIO DMSXMD DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS DMSXSU
ZFOERCOD	000004	DMSXBG DMSXPO DMSXSC
ZFOFKCOD	000006	DMSXBG DMSXSC DMSXSD
ZFOFLAG	000008	DMSXFD DMSXGT DMSXIN DMSXMA DMSXPT DMSXUP
ZFOFLAG1	000083	DMSXBG DMSXCG DMSXCT DMSXED DMSXER DMSXFC DMSXIN DMSXIO DMSXMA DMSXMC DMSXMD DMSXML DMSXPO DMSXSC DMSXSD DMSXSE DMSXSU
ZFOFLAG2	000071	DMSXBG DMSXCM DMSXCT DMSXED DMSXER DMSXFC DMSXHL DMSXIO DMSXMC DMSXMD DMSXMS DMSXPO DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS DMSXSU
ZFOFLAG3	000110	DMSXBG DMSXCG DMSXCT DMSXDC DMSXED DMSXFC DMSXFD DMSXIN DMSXIO DMSXMA DMSXMD DMSXML DMSXPX DMSXSC DMSXSD DMSXSE DMSXSS DMSXSU
ZFOFLAG4	000015	DMSXBG DMSXCM DMSXDC DMSXSE
ZFOFMODE	000026	DMSXCT DMSXFD DMSXGT DMSXPT DMSXSE DMSXSU DMSXUP
ZFOFNAME	000041	DMSXCM DMSXCT DMSXED DMSXFD DMSXGT DMSXIN DMSXPT DMSXSE DMSXSU DMSXUP
ZFOFREPT	000028	DMSXCG DMSXCN DMSXED DMSXFC DMSXMD DMSXPT DMSXPX DMSXSS DMSXST DMSXUP
ZFOFTYPE	000027	DMSXCM DMSXCT DMSXFD DMSXGT DMSXIN DMSXPT DMSXSE DMSXSU DMSXUP
ZFOHVCED	000004	DMSXBG DMSXSE
ZFOHVCP1	000002	DMSXBG DMSXSE
ZFOHVCP3	000001	DMSXSE
ZFOIMPCM	000009	DMSXCM DMSXDC DMSXSE
ZFOINVCM	000015	DMSXCT DMSXPX DMSXSD DMSXSS
ZFOIOCMP	000006	DMSXBG DMSXSC DMSXSD DMSXSE
ZFOIOTBL	000017	DMSXBG DMSXCT DMSXIO DMSXPO DMSXSC DMSXSD DMSXSE
ZFOKPLIN	000009	DMSXPX DMSXSC DMSXSD DMSXSS
ZFOKYCOD	000004	DMSXSC
ZFOLADDR	000069	DMSXCG DMSXCN DMSXED DMSXFC DMSXFD DMSXIN DMSXML DMSXMS DMSXPT DMSXPX DMSXRE DMSXSD DMSXST DMSXUP
ZFOLBUFF	000033	DMSXFD DMSXGT DMSXIN DMSXMA DMSXPT DMSXUP
ZFOLBWPT	000048	DMSXCG DMSXCN DMSXED DMSXFC DMSXIN DMSXML DMSXMS DMSXSD DMSXUP
ZFOLDSCR	000023	DMSXED DMSXFC DMSXIN DMSXST DMSXUP
ZFOLFLAG	000110	DMSXCG DMSXCT DMSXFC DMSXIN DMSXMD DMSXPT DMSXPX DMSXSD DMSXSE DMSXSS DMSXST DMSXUP

LABEL	COUNT	REFERENCES											
ZFOLFWPT	000107	DMSXBG	DMSXCG	DMSXCN	DMSXCT	DMSXED	DMSXFC	DMSXFD	DMSXIN	DMSXMD	DMSXMS	DMSXPX	DMSXRE
		DMSXSD	DMSXSE	DMSXST	DMSXUP								
ZFOLGOP1	000072	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXED	DMSXHL	DMSXMC	DMSXMD	DMSXML	DMSXSE	DMSXSS	DMSXUP
ZFOLGOP2	000028	DMSXCG	DMSXCT	DMSXDC	DMSXHL	DMSXSE	DMSXUP						
ZFOLGOP3	000009	DMSXCT	DMSXSE	DMSXUP									
ZFOLGOP4	000002	DMSXUP											
ZFOLGOP5	000006	DMSXFD	DMSXUP										
ZFOLGOP7	000003	DMSXDS											
ZFOLGOP8	000011	DMSXDS	DMSXFD	DMSXIN	DMSXSE								
ZFOLGSCB	000009	DMSXBG	DMSXPO	DMSXSC									
ZFOLNAME	000024	DMSXBG	DMSXCN	DMSXCT	DMSXED	DMSXFC	DMSXFD	DMSXPX	DMSXSE				
ZFOLNCHG	000024	DMSXCG	DMSXFC	DMSXSD	DMSXUP								
ZFOLNCUR	000007	DMSXFC	DMSXPX										
ZFOLNDEL	000013	DMSXUP											
ZFOLNDSP	000022	DMSXCG	DMSXCT	DMSXFC	DMSXPX	DMSXSD	DMSXSE						
ZFOLNNEW	000018	DMSXFC	DMSXUP										
ZFOLRBUF	000037	DMSXCT	DMSXED	DMSXIN	DMSXMA	DMSXMD	DMSXSS	DMSXSU	DMSXUP				
ZFOLSCPT	000028	DMSXBG	DMSXCT	DMSXED	DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXSU				
ZFOLSTSV	000155	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT
		DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXPX	DMSXSC	DMSXSD
		DMSXSE	DMSXSS	DMSXST	DMSXSU	DMSXUP							
ZFOLZFOB	000001	DMSXBG											
ZFOLZFOD	000003	DMSXBG											
ZFOMCRNG	000007	DMSXCT	DMSXED	DMSXER	DMSXMA	DMSXSU							
ZFOMOVUP	000034	DMSXCG	DMSXCT	DMSXFC	DMSXFD	DMSXML	DMSXSC	DMSXSU					
ZFOMSGCT	000023	DMSXBG	DMSXCT	DMSXIN	DMSXIO	DMSXMA	DMSXMD	DMSXSC					
ZFONC	000162	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT
		DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXMS	DMSXPO	DMSXPT	DMSXPX	DMSXSC
		DMSXSD	DMSXSE	DMSXSS	DMSXST	DMSXSU	DMSXUP						
ZFONCOLS	000065	DMSXBG	DMSXCT	DMSXIO	DMSXPO	DMSXPX	DMSXSC	DMSXSD	DMSXSE	DMSXSS			
ZFONDSPC	000006	DMSXBG	DMSXSE										
ZFONFILE	000010	DMSXBG	DMSXED	DMSXER	DMSXIN	DMSXSD	DMSXSU						
ZFONOBRD	000002	DMSXIN	DMSXMA										
ZFONROWS	000008	DMSXBG	DMSXCT	DMSXSC	DMSXSE	DMSXSS							
ZFOOPABS	000084	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXED	DMSXGT	DMSXHL	DMSXMD	DMSXPT	DMSXSE	DMSXSU	
ZFOOPFL1	000044	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXED	DMSXGT	DMSXHL	DMSXMD	DMSXSC	DMSXSE	DMSXSU	
ZFOOPFL2	000050	DMSXCG	DMSXCT	DMSXGT	DMSXHL	DMSXMD	DMSXPT	DMSXSC	DMSXSE	DMSXSU			
ZFOOPFL3	000016	DMSXCT	DMSXGT	DMSXMD	DMSXPT	DMSXSC	DMSXSE	DMSXSU					
ZFOOPFL4	000004	DMSXGT	DMSXPT										
ZFOOPNB1	000073	DMSXBG	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXDS	DMSXIN	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPT
		DMSXSC	DMSXSE	DMSXSS	DMSXSU	DMSXUP							
ZFOOPNB2	000033	DMSXCG	DMSXCM	DMSXCT	DMSXSE								
ZFOOPNB3	000020	DMSXCG	DMSXCM	DMSXCT	DMSXGT	DMSXIN	DMSXSE	DMSXUP					
ZFOOPNB4	000007	DMSXCG	DMSXGT										
ZFOOPNB5	000013	DMSXCG	DMSXGT	DMSXIN	DMSXUP								
ZFOOPNB6	000008	DMSXCG											

LABEL	COUNT	REFERENCES											
ZFOOPNB7	000007	DMSXCG	DMSXFD										
ZFOOPNB8	000009	DMSXCG	DMSXUP										
ZFOOPNEG	000005	DMSXFD											
ZFOOPNSP	000007	DMSXDC	DMSXGT	DMSXHL	DMSXPT	DMSXSE							
ZFOOPNUM	000006	DMSXCT	DMSXDC	DMSXGT									
ZFOOPSTR	000017	DMSXDC	DMSXGT	DMSXMD	DMSXSC	DMSXSE	DMSXSU						
ZFOOPST1	000255	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXED	DMSXFD	DMSXGT	DMSXHL	DMSXMC	DMSXMD	DMSXML	DMSXSC
		DMSXSE	DMSXSS	DMSXSU	DMSXUP								
ZFOOPST2	000088	DMSXCG	DMSXCT	DMSXDC	DMSXFD	DMSXGT	DMSXHL	DMSXMD	DMSXPT	DMSXSC	DMSXSE	DMSXSU	DMSXUP
ZFOOPST3	000040	DMSXCG	DMSXCT	DMSXFD	DMSXGT	DMSXMD	DMSXPT	DMSXSC	DMSXSE	DMSXSU	DMSXUP		
ZFOOPST4	000012	DMSXCG	DMSXPT	DMSXPX	DMSXSC	DMSXUP							
ZFOOPST5	000002	DMSXUP											
ZFOOPST6	000002	DMSXUP											
ZFOOPST7	000021	DMSXSE											
ZFOOPST8	000021	DMSXFC	DMSXGT	DMSXHL	DMSXPT	DMSXSC	DMSXSE						
ZFOOPTGT	000003	DMSXCG	DMSXCT	DMSXDC									
ZFOPLIST	000121	DMSXBG	DMSXCM	DMSXCT	DMSXDC	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXMA	DMSXMD	DMSXPT	DMSXSC
		DMSXSD	DMSXSE	DMSXSS	DMSXSU	DMSXUP							
ZFOPRFER	000003	DMSXIN	DMSXSU										
ZFOPRFIN	000004	DMSXMD	DMSXPX	DMSXSD									
ZFOPRFPT	000005	DMSXBG	DMSXPX	DMSXSE									
ZFOPROFL	000005	DMSXED	DMSXIN	DMSXMA	DMSXSU								
ZFOPRVTB	000009	DMSXDC	DMSXSE										
ZFORDBUF	000070	DMSXCT	DMSXDS	DMSXED	DMSXIN	DMSXIO	DMSXMA	DMSXMD	DMSXSC	DMSXSS	DMSXSU	DMSXUP	
ZFORDSVR	000003	DMSXSC	DMSXSU										
ZFORECFM	000015	DMSXFD	DMSXGT	DMSXIN	DMSXMA	DMSXPT	DMSXUP						
ZFORECPT	000004	DMSXMA											
ZFORETMC	000003	DMSXIN											
ZFORMTUB	000007	DMSXBG	DMSXPO	DMSXSD									
ZFORTBYT	000106	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXFC	DMSXGT	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXSE
		DMSXSS	DMSXSU	DMSXUP									
ZFORTCOD	000075	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXED	DMSXFD	DMSXGT	DMSXHL	DMSXIN	DMSXMA	DMSXMD	DMSXML
		DMSXPT	DMSXSC	DMSXSE	DMSXSU								
ZFOSAVNB	000008	DMSXBG	DMSXCT	DMSXED	DMSXMA	DMSXSC	DMSXSS						
ZFOSAVSV	000010	DMSXBG	DMSXCT	DMSXED	DMSXMA	DMSXSC	DMSXSS	DMSXSU					
ZFOSAV01	000004	DMSXIN	DMSXMA	DMSXSU									
ZFOSAV02	000001	DMSXMA											
ZFOSCRRD	000006	DMSXCT	DMSXIO	DMSXPX	DMSXSC	DMSXSD							
ZFOSINDX	000002	DMSXSC	DMSXSD										
ZFOSOSIM	000016	DMSXBG	DMSXPX	DMSXSC	DMSXSD	DMSXSE	DMSXSS						
ZFOSPCVC	000004	DMSXSU											
ZFOSUBCM	000065	DMSXCG	DMSXCT	DMSXDC	DMSXED	DMSXHL	DMSXIN	DMSXMA	DMSXMC	DMSXML	DMSXSE		
ZFOSUSED	000003	DMSXCT	DMSXSC										
ZFOSVCED	000002	DMSXSE											
ZFOSVC04	000001	DMSXSE											
ZFOSVEND	000001	DMSXBG											

LABEL	COUNT	REFERENCES											
ZFOSYNBF	000004	DMSXDC	DMSXSE										
ZFOSYNBL	000001	DMSXDC											
ZFOSYNPT	000007	DMSXBG	DMSXDC	DMSXIN	DMSXSE								
ZFOTABS1	000003	DMSXFC	DMSXSC										
ZFOTWRMD	000015	DMSXBG	DMSXCT	DMSXIN	DMSXSC	DMSXSE	DMSXSU						
ZFOTXTON	000009	DMSXBG	DMSXSC	DMSXSD	DMSXSE								
ZFOUFLDS	000011	DMSXBG	DMSXSD	DMSXSE									
ZFOWKBUF	000152	DMSXBG	DMSXDC	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXIN	DMSXMA	DMSXMC	DMSXPO	DMSXSC	DMSXSD
		DMSXSE	DMSXSS	DMSXSU	DMSXUP								
ZFOWKTBL	000016	DMSXCG	DMSXFD	DMSXIN	DMSXSC	DMSXSD							
ZFOXER	000221	DMSXBG	DMSXCG	DMSXCM	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT	DMSXIN
		DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXSC	DMSXSD	DMSXSE	DMSXSS	DMSXSU
		DMSXUP											
ZFOXSUFL	000155	DMSXBG	DMSXCG	DMSXCM	DMSXCN	DMSXCT	DMSXDC	DMSXDS	DMSXED	DMSXER	DMSXFC	DMSXFD	DMSXGT
		DMSXHL	DMSXIN	DMSXIO	DMSXMA	DMSXMC	DMSXMD	DMSXML	DMSXPO	DMSXPT	DMSXPX	DMSXSC	DMSXSD
		DMSXSE	DMSXSS	DMSXST	DMSXSU	DMSXUP							
ZFOZMAPT	000010	DMSXBG	DMSXCT	DMSXMA									
ZMACST	000010	DMSXBG	DMSXCT	DMSXMA									
ZMAEXELG	000002	DMSXMA											
ZMAEXEPT	000004	DMSXMA											
ZMAFNAME	000003	DMSXCT	DMSXMA										
ZMAFWPTR	000012	DMSXBG	DMSXCT	DMSXMA									
ZMALZMAD	000002	DMSXMA											
ZMAMCLST	000003	DMSXMA											
ZMANBMST	000003	DMSXMA											
ZONE1	000011	DMSEDI	DMSEDX										
ZONE2	000016	DMSEDI	DMSEDX										
ZPACKBL	000002	DMSXFD	DMSXIN										
ZPAFLAG1	000023	DMSXFD	DMSXIN										
ZPALZPAD	000008	DMSXFD	DMSXIN										
ZPAPKBUF	000016	DMSXFD	DMSXIN										
ZPAPKBXE	000007	DMSXFD	DMSXIN										
ZPAPKBX2	000003	DMSXFD											
ZPAPKBX3	000003	DMSXFD											
ZPAPKBX4	000002	DMSXFD											
ZPAPKCC	000003	DMSXFD											
ZPAPKDAF	000005	DMSXFD	DMSXIN										
ZPAPKELF	000003	DMSXFD	DMSXIN										
ZPAPKERF	000002	DMSXFD	DMSXIN										
ZPAPKFFF	000006	DMSXFD	DMSXIN										
ZPAPKFIL	000002	DMSXIN											
ZPAPKSCF	000004	DMSXFD	DMSXIN										

CMS Diagnostic Aids

This section contains the following information:

- A list of devices Supported by a CMS Virtual Machine
- DMSFREX Error Codes
- Abend Codes

Figure 29 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, 3310, 3330, 3340, 3350, 3370, 3380	190	DSK0	System disk (read-only)
2314, 3310, 3330, 3340, 3350, 3370, 3380	191[2]	DSK1	Primary disk (user files)
2314, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSK2	Disk (user files)
2314, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSK3	Disk (user files)
2314, 3310, 3330, 3340, 3350, 3370, 3380	192	DSK4	Disk (user files)
2314, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSK5	Disk (user files)
2314, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSK6	Disk (user files)
2314, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSK7	Disk (user files)
2314, 3310, 3330, 3340, 3350, 3370, 3380	19E	DSK8	Disk (user files)
2314, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSK9	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKH	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKI	Disk (user files)

[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 29. Devices Supported by a CMS Virtual Machine (Part 1 of 2)

Virtual IBM Device	Virtual Address[1]	Symbolic Name	Device Type
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKJ	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKK	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKL	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKM	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKN	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKO	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKP	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKQ	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKR	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKT	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKU	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKV	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKW	Disk (user files)
2314, 2319, 3310, 3330, 3340, 3350, 3370, 3380	ccu	DSKX	Disk (user files)
1403, 3203, 3211, 1443, 3262, 3289E, 3800	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.

Figure 29. Devices Supported by a CMS Virtual Machine (Part 2 of 2)

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 30 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 VSE for which there is no OS equivalent. An internal error occurred in a VSE/VSAM routine.	Refer to <u>VSE/VSAM Messeges and Codes</u>, 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 30. 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 VSE/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 30. 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.
160	DMSXSU	Occurs when XEDIT cannot allocate a save area to a called routine.	None. Abend recovery takes place when the next command is entered.
174	DMSVIB	The OS interface to VSE/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 VSE/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.
500	DMSTLB	A block count error was detected when reading a SL tape. User replied 'cancel' to message 425R or the user's program contained a block count error routine that returned a code of 0 under OS simulation.	Find out what caused the block count error. Then reload CMS and rerun the job.
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.

Figure 30. CMS Abend Codes (Part 3 of 4)

Abend Code	Module Name	Cause of Abend	Action
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.
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 30. 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 VM/SP.

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.
BBOX	DSECT of boundary box; contains beginning and ending addresses of background communication region.
BGCOM	DSECT of background communication region.
BGTCB	Task Control Block.
*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 <u>VM/SP CMS Command and Macro Reference</u>.
*CORG	Sets the origin for CSECT.
*DBGSECT	Generates a CSECT or DSECT for DEBUG environment variables.
DESTYP	Used by the XEDIT module DMSXIN to determine filetype default settings. The DESTYP block is defined in DMSXTF.
*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.
DIB	Disk Information Blocks.
*DIOSECT	Generates a CSECT or DSECT for all I/O information.
DISPW	Generates the calling sequence for the display terminal interface. Refer to <u>VM/SP System Programmer's Guide</u>.
DMSABN	ABEND the virtual machine. Refer to <u>VM/SP System Programmer's Guide</u>.
*DMSCCB	DSECT describes field of DOS command control block (CCB). Refer to <u>VM/SP Data Areas and Control Block Logic</u>.
*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 <u>VM/SP System Logic and Problem Determination Guide--Volume 2</u>.
DMSFREE	Gets free storage. Refer to <u>VM/SP System Programmer's Guide</u>.
*DMSFRES	Calls system free storage service routines.
DMSFRET	Releases free storage. Refer to <u>VM/SP System Programmer's Guide</u>.
*DMSFREX	Calls system free storage service routines.
*DMSFRT	Generates a DSECT for free storage management work area.
*DMSFRX	Submacro called by DMSFRET.

CMS Macro	Function
DMSFST	Sets up a file status table for a given file. Refer to VM/SP System Programmer's Guide.
DMSKEY	Sets nucleus protection on or off. Refer to VM/SP 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.
DOSAVE	DSECT, describes fields in the logical transient area (LTA).
DOSCB	DOS simulation control block used for simulation of the CMS file control block (FCB).
DOSCON	Creates CMS/DOS control blocks for DMSNUC.
DTFSD	DTFSD DSECT.
DTFX	DTF extension DSECT.
*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/SP CMS Command and Macro Reference.
*FSCBD	DSECT that describes fields in CMS PLIST for related commands.
FSCLOSE	Closes a file. Refer to VM/SP CMS Command and Macro Reference.
*FSENTR	Used by CMS file system routines at entry.
FSERASE	Erases a file. Refer to VM/SP CMS Command and Macro Reference.
FSOPEN	Opens a file. Refer to VM/SP CMS Command and Macro Reference.
*FSPOINT	Executes the CMS POINT function.
FSREAD	Reads a record from a file. Refer to VM/SP CMS Command and Macro Reference.
FSSTATE	Checks for an existing file. Refer to VM/SP 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/SP 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/SP CMS Command and Macro Reference.
HNDINT	Handles interrupt on devices. Refer to VM/SP CMS Command and Macro Reference.
HNDSVC	Handles SVCs. Refer to VM/SP CMS Command and Macro Reference.
IJJHCPL	Common VTOC handler input PLIST.
IJJHDLST	Common VTOC handler descriptor list DSECT.
IJJHMFT1	Format 1 VTOC label DSECT.
*IO	Contains PLISTs needed to access CMS I/O routines.
*IOSECT	Defines miscellaneous I/O variables.

CMS Macro	Function
*KEYSECT	Contains variables necessary for storage key handling.
*KXCHK	Checks to see if HX has been entered by the user.
LABREC	DLBL/EXTENT record.
*LDM	Loads double multiple (for floating point registers).
*LDRST	CMS Loader work area.
LINEDIT	Types a line to the terminal. Refer to _VM/SP CMS Command and Macro Reference_.
LOCKTAB	LOCK/UNLOCK resource table.
LPLDCT	LABEL macro PLIST.
LSCREEN	Used by XEDIT modules to describe the layout of a logical screen on the physical screen. LSCREEN is built by module DMSXSD.
*NUCON	Generates a DSECT CMS nucleus constant area.
OCTS	OPEN/CLOSE transient SVA PLIST.
*OVSECT	DMSOVS work area.
*OSFST	Defines an OS file status table for OS ACCESS.
*PDSSECT	DSECT used for processing MACLIB files.
*PGMSECT	Defines work area for DMSITP.
PIBTAB	DSECT, program information block.
PIB2TAB	DSECT, program information block extension.
PRINTL	Prints a line on the printer. Refer to _VM/SP CMS Command and Macro Reference_.
PRSCB	Used by the XEDIT subcommands PRESERVE and RESTORE. It is built by module DMSXCT.
PUNCHC	Punches a card. Refer to _VM/SP CMS Command and Macro Reference_.
RDCARD	Reads a card from the reader. Refer to _VM/SP CMS Command and Macro Reference_.
RDTAPE	Reads a record from tape. Refer to _VM/SP CMS Command and Macro Reference_.
RDTERM	Reads a record from the terminal. Refer to _VM/SP CMS Command and Macro Reference_.
RECSAVE	Used by XEDIT modules to describe the address list for nested macro calls. It is built by DMSXMA.
REGEQU	Generates symbolic register equates. Refer to _VM/SP CMS Command and Macro Reference_.
*RELPAGES	Sets the release pages flag.
REQDES	Used by XEDIT modules to describe all XEDIT subcommands and their operands and syntax. The REQDES block is defined in DMSXTB.
SAVEREG	Used by XEDIT modules to save register contents during subroutine calls.
*STDM	Storage for multiple floating-point registers.
STRINIT	Initializes storage. Refer to _VM/SP CMS Command and Macro Reference_.
*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.
SYNSUB	Used by XEDIT modules to describe the synonyms defined for XEDIT subcommands. A SYNSUB block is built dynamically by DMSXDC each time a synonym is defined.
SYSCOM	DSECT of system communication region.
*SYSLOAD	Puts in a specified register the address of a specified routine in NUCON.
*SYSNAMES	Saves system names table loaded via CMS routines.

CMS Macro	Function
TAPECTL	Positions a tape. Refer to _VM/SP CMS Command and Macro Reference_.
*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 _VM/SP CMS Command and Macro Reference_.
WAITT	Waits until all pending I/O to the terminal has completed. Refer to _VM/SP CMS Command and Macro Reference_.
WRTAPE	Writes a record to tape. Refer to _VM/SP CMS Command and Macro Reference_.
WRTERM	Writes a record to the terminal. Refer to _VM/SP CMS Command and Macro Reference_.
ZDESC	Used by XEDIT modules to describe file characteristics.
ZFONC	Used by XEDIT modules as a common work area. It is built by DMSXBG only once in an editing session.
ZMACST	Used by XEDIT modules to describe an XEDIT macro in storage. A ZMACST block is built dynamically by DMSXMA each time a macro is invoked.
ZPACK	Used by XEDIT modules when a file is being packed or unpacked. It is built by DMSXIN or DMSXFD.

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).
ABTAB	DSECT of abnormal termination option table.
FICL	DSECT, CMS/DOS first in class table.
NICL	DSECT, CMS/DOS number in class table.
PUBOWNER	DSECT, physical unit block ownership table.
ANCHTAB	DSECT, DOS/VS anchor table.
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.
EXCPW	DSECT, work area for DMSXCP routine.
LUBTAB	DSECT for CMS/DOS logical unit block.

Appendix C: CMS/DOS Support Modules

The modules listed below (by phase) make up the CMSBAM segment. The phases and modules (except DMSLBR) retain their VSE identifiers.

Phase	Modules				
$IJBLKMD	IJBLKMD				
$IJBLBSL	IJBLBSL				
$IJGXCP	IJGXCP				
$IJGXDI	IJGXDI				
$IJGXSDF	IJGXSDF				
$IJGXSDU	IJGXSDU				
$IJGXSDV	IJGXSDV				
$IJGXSDW	IJGXSDW				
$IJBLKMD	IJBLKMD				
$IJGXSFI	IJGXSFI				
$IJGXSRI	IJGXSRI				
$IJGXSSR	IJGXSSR				
$IJGXSVI	IJGXSVI				
$IJJGTOP	IJJGDACX	IJJGDAI1	IJJGDAI2	IJJGDAMO	IJJGDAMS
	IJJGDAMX	IJJGDAO1	IJJGDAO2	IJJGDAO3	IJJGDAO4
	IJJGDAO5	IJJGDARL	IJJGDART	IJJGDAVC	IJJGMFBA
	IJJGMIOI	IJJGMLLM	IJJGMMBF	IJJGMSOO	IJJGMS10
	IJJGMTOP	IJJGSDBH	IJJGSDBS	IJJGSDCD	IJJGSDCI
	IJJGSDCI	IJJGSDCW	IJJGSDFP	IJJGSDGC	IJJGSDI1
	IJJGSDI2	IJJGSDI3	IJJGSDI4	IJJGSDI5	IJJGSDLP
	IJJGSDMC	IJJGSDMF	IJJGSDMN	IJJGSDMO	IJJGSDNV
	IJJGSDO1	IJJGSDO2	IJJGSDO4	IJJGSDO5	IJJGSDO6
	IJJGSDO7	IJJGSDRL	IJJGSDSF	IJJGSDUL	IJJGSDVH
	IJJGSDW1	IJJGSDW2	IJJGSDW3	IJJGSDW4	IJJGSDXT
	IJJGVDOO	IJJGVD10	IJJGVMOO	IJJGVM10	
$IJJHCVH	IJJHCCVO	IJJHCVHO	IJJHOPNO	IJJHRDSO	IJJHSRNO
	IJJHWDSO				
DMSLBR	DMSLBR				

Index

IBM Virtual Machine/System Product:
System Logic and Problem Determination
Guide Volume 2(CMS) LY20-0893-0

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

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.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

	Yes	*No*
• Does the publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

• What is your occupation? ______________________________

• How do you use this publication:

As an introduction to the subject?	☐	As an instructor in class?	☐
For advanced knowledge of the subject?	☐	As a student in class?	☐
To learn about operating procedures?	☐	As a reference manual?	☐

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Reader's Comment Form

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department G60
P. O. Box 6
Endicott, New York 13760

Fold | Fold

If you would like a reply, *please print:*

Your Name ______________________

Company Name ______________ *Department* ________

Street Address ______________________

City ______________________

State ______________ *Zip Code* ________

IBM Branch Office serving you ______________________

IBM®

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

Cut or Fold Along Line

LY20-0893-0

IBM VM/SP System Logic and Prob. Determination [illegible] Vol 2 (CMS) Printed in U.S.A. LY20-0893-0

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