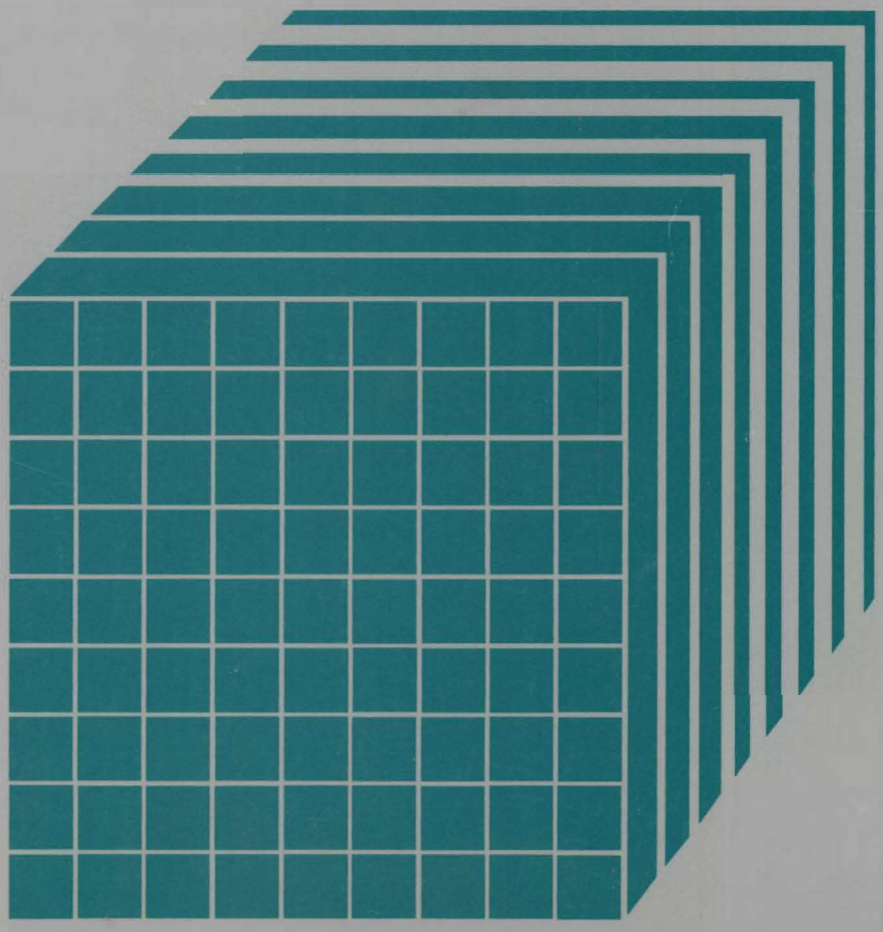
IBM

Virtual Machine/ System Product

CP for System Programming

Release 5

SC24-5285-0



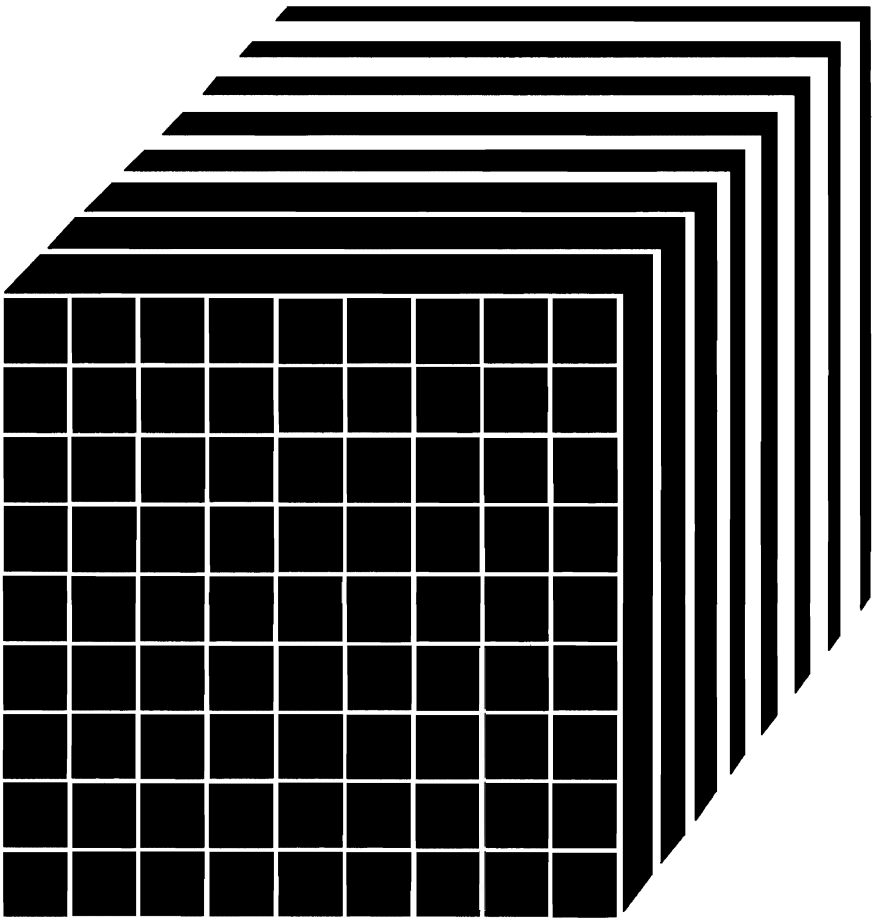
IBM

Virtual Machine/
System Product

CP for System Programming

Release 5

SC24-5285-0



First Edition (December 1986)

This edition, SC24-5285-0, applies to Release 5 of IBM Virtual Machine/System Product (VM/SP) unless otherwise indicated in new editions or Technical Newsletters. It contains material formerly included in the *VM/SP System Programmer's Guide*. Changes are continually made to the information contained herein; before using this publication in connection with the operation of IBM systems, consult the *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Summary of Changes

A cumulative **Summary of Changes** begins on page 235.

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

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

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

Preface

This is a reference for system programmers, system analysts, and others who implement and extend the functions of the Control Program (CP) of IBM's Virtual Machine/System Product (VM/SP). It assumes some experience with programming concepts and techniques.

This CP reference is one of a set of reference manuals for VM/SP system programmers. It consists mostly of material extracted from the *VM/SP System Programmer's Guide* for Release 4. Some material has been transferred from the *VM/SP Operator's Guide* for Release 4. Changes for Release 5 of VM/SP are marked by a vertical bar in the left margin and outlined in the "Summary of Changes" on page 235.

Other information formerly contained in the *VM/SP System Programmer's Guide* is now in *VM/SP CMS for System Programming*, *VM System Facilities for Programming*, and *VM Diagnosis Guide*. The order numbers for these and related publications can be found in the Bibliography.

This reference consists of three parts and an appendix:

"Part 1: Control Program (CP) Features" (chapters 1—11) describes the functions of CP and provides guidance in using some CP features.

"Part 2: Performance" (chapters 12—14) describes options available in VM/SP to analyze and improve the performance of virtual machines and operating systems.

"Part 3: Hardware Considerations" (chapters 15—21) is a reference for dealing with processor features and real peripheral devices.

Appendix, "VM/SP Monitor Tape Format and Content" on page 217 describes the format and contents of data records for classes and codes of MONITOR CALL.

After the appendix, there are four sections that can help you use this reference more easily:

"Summary of Changes" on page 235 is a cumulative summary of the changes to the Control Program of the Virtual Machine/System Product, as described here (Release 5) and in the *VM/SP System Programmer's Guide* (Release 4 and earlier).

"Glossary of Terms and Abbreviations" on page 243 defines technical terms and abbreviations, and acronyms and device numbers that are used for IBM products.

"Bibliography" on page 249 lists related publications.

"Index" on page 255 lists topics alphabetically and points to the pages where they are discussed. It includes cross-references to the principal topics of the other volumes of the system reference sublibrary.

Contents

Part 1: Control Program (CP) Features 1

Chapter 1. Introduction to the VM/SP Control Program (CP) . . . 3

Chapter 2. Storage Protection 5

Chapter 3. Virtual Storage Preservation 7
Using the VMSAVE Option 7
Setting Priority 8
Defining VMSAVE Areas 8
Specifying Overlapping Areas 9
Saving Other Systems 10

Chapter 4. I/O Management 11
Virtual Machine I/O 11
Dedicated Channels 12
Spool Files 13
Recovering Spool Files 14
Warm Start 14
Checkpoint Start 14
Force Start 15

Chapter 5. Alternate Nucleus Support 17
Backup Directories and Override Files 18

Chapter 6. Changing Commands and Privilege Classes 19
Command Privilege Classes and Types 19
Tailoring the Class Structure 21
Changing Privilege Classes and User Access 21
Planning the Command Authorization for the System 23
Determining Functions to be Done by Users 23
Assigning Commands to Kinds of Users 24
Associating Privilege Classes with Commands and Users 26
Further Considerations 27
How to Assign Privilege Classes to Commands and DIAGNOSE Codes . 28
Creating a Class Override File 28
Class Override File Example 30
Verifying and Activating the Override File 31
Reverting to the IBM-Defined User Classes 32
How Users Can Find Which Commands They Can Issue 33
Changing a User's Privilege Classes 33
Directory Entry Example 34
Defining Privilege Classes for a Virtual Machine 35
How to Change the Privilege Class of Certain Internal CP Functions . . 36

Chapter 7. Interrupt Handling 39

Device Support Facilities 121
Format/Allocate Service Program 122
CP Disk Formats 122
Format for CP-Owned CKD Devices 123
Format for CP-Owned FBA Devices 125
Format/Allocate Program Input 126
Format/Allocate Program Card Input 126
Format/Allocate Console Input 133
DASD Dump Restore Service Program (DDR) 138
Invoking DDR Under CMS 138
Invoking DDR as a Stand-alone Program 139
DDR 8809 Data Streaming Support 140
DDR Control Statements 141
INPUT/OUTPUT Control Statement 141
SYSPRINT Control Statement 145
Function Statements 145
Restrictions 150
PRINT/TYPE Function Statement 153
Responses 156

Chapter 16. VM/SP Use of the IBM 3850 MSS 161
VM/SP Access to the MASS Storage Control 161
Asynchronous MSS Mount Processing 162
VM/SP Processing of MSS Cylinder Faults 162
Backup and Recovery of MSS Volumes 163

Chapter 17. Timers in a Virtual Machine 165
Interval Timer 165
Virtual Interval Timer Assist 165
Processor Timers 166
TOD Clock 167
Clock Comparator 167
Pseudo Timer 167
Pseudo Timer Start I/O 168
Pseudo Timer DIAGNOSE 168

Chapter 18. CP in Attached Processor and Multiprocessor Modes 169
Multiprocessor Environment 169
Attached Processor Environment 170
Advantages of the AP/MP Environment 170
Facilitating an AP/MP Environment 170
Prefixing 170
Identifying a Processor Address 172
Signaling with the SIGNAL Macro 172
Time-of-Day (TOD) Clock Synchronization Check 175
Fetching and Storing 175
Locks and Serialization of Functions 176
Locking Hierarchy 177
Locking Structure 177
LOCK Macro 179
Affinity 181
How to Set Affinity 181
Shared Segments in an AP/MP Environment 181

SWTCHVM Macro 181
Configuring I/O Devices for an MP System 182

Chapter 19. Print Buffers and Forms Control 183
Adding New Print Buffer Images 186
UCS Buffer Images for the 1403 Printer 186
UCSB Buffer Images for the 3211 Printer 188
FOB Buffer Images for the 3289 Model 4 Printer 192
UCC Buffer Images for the 3203 Printer 193
PIB Buffer Images for the 3262 Model I and II Printers 196
Forms Control Buffer 197
Extended FCB Macro Instruction 200

Chapter 20. IBM 3800 Printing Subsystem 203
Using the 3800 Printer as a Dedicated Device 204
Using the 3800 Printer as Virtual Spooling Devices 205
Defining a Virtual 3800 Printer 205
Using the SPOOL and CHANGE Commands 205
Using the SETPRT Command 206
Using the 3800 Printer as a Real Spooling Device 206
Specifying Printer Options 207
The GENIMAGE Command 207
Maintaining the 3800 Image Library 207
The GENIMAGE Service Program 208
Responses from GENIMAGE Command 209
The IMAGELIB Service Routine 210
The IMAGEMOD Command 210
Recovering from I/O Errors 212

Chapter 21. 3088 Multichannel Control Unit 213
System Programmer Considerations 213
RDEVICE MACRO 213
RCTLUNIT MACRO 213
Special Directory Control Statement 214
Virtual 3088 Support 214
Command Use and 3088 Support 214
Channel Command Words 215
Diagnostic Aids 215
Online Testing 215
Messages and MNOTES To Support 3088 Devices 215

Appendix. VM/SP Monitor Tape Format and Content 217
Header Record 217
Data Records 218
Class Zero - Codes for Tape Header, Trailer, and Data Suspension Records 218
Class Zero - PERFORM 219
Class One - RESPONSE 226
Class Two - SCHEDULE 227
Class Four - USER 229
Class Five - INSTSIM 230
Class Six - DASTAP 231
Class Seven - SEEKS 232
Class Eight - SYSPROF -- Additional data for system profile class 233

Summary of Changes 235

Glossary of Terms and Abbreviations 243

Bibliography 249

Index 255

Figures

1. 2K Storage Protection Key 6
2. Relationships of Privilege Classes, Types, and Administrative Functions 20
3. Different System Users and Their Responsibilities 24
4. DIAGNOSE Instructions That Can Be Respecified on an OVERRIDE Control Stat. 30
5. Relationships of Scheduler Lists and Queue Levels 71
6. Storage Layout in a Virtual=Real Machine 84
7. Definitions of storage levels and segment tables. 92
8. MONITOR CALL Classes 111
9. Format of 3330 Cylinders for Use by CP 124
10. 3330, 3340, 3350 or 3380 Cylinder 0 Format 124
11. Using the Format Program Label Function 134
12. Using the Format Program Allocate Function 134
13. Using the Format Program Allocate Overlap Function 135
14. Using the Format Program Label Function for FBA Devices 135
15. Using the Format Program Allocate Overlap Function for FBA Devices 136
16. Using the Format Program Allocate Overlap Function for FBA Devices 137
17. Annotated Sample of Output from the TYPE and PRINT Functions of the DDR Program 159
18. Formats of Pseudo Timer Information 168
19. Storage Layout in a Virtual=Real Machine 171
20. Sample of the Correct Way to Set a Flag in an AP/MP Environment 176
21. Hierarchy of VM/SP Locks 177
22. UCSB Associative Field Chart 191
23. VM-Supplied Character Arrangement Tables for the 3800 Model 1 and Model 3 Printers 209
24. VM-Supplied Character Arrangement Tables for the 3800 Model 3 Only 209
25. Prefixes of TEXT Deck Names, Output by the GENIMAGE Command 209
26. CP commands and 3088 Support 214
27. VM/SP System Programming Manuals for VM/SP Release 5. . . . 236

Part 1: Control Program (CP) Features

Part 1 contains the following information:

- Introduction to the VM/SP Control Program
- Storage Protection
- Virtual Machine I/O Management
- Alternate Nucleus Support
- CP Coding Conventions
- CP Command Classes
- Interrupt Handling
- Accounting Information
- Generating Saved Systems
- Security Measures

Chapter 1. Introduction to the VM/SP Control Program (CP)

The VM/SP Control Program, CP, manages the resources of a single computer so that multiple users each have the impression of controlling a dedicated computer system. These apparent multiple computer systems are called "virtual" machines and each is the functional equivalent of an IBM System/370.

Each virtual machine is managed at two levels of operation: its own and the system's. The work to be done by the virtual machine is scheduled and controlled by some System/360 or System/370 operating system running on the virtual machine. Furthermore, the CP commands allow you to control the virtual machine from the terminal, much as an operator controls a real machine. You can stop virtual machine execution at any time by using the 3066 terminal's attention key or the 3270 terminal's ENTER or PA1 key. To restart execution, enter the CP BEGIN command. You can also simulate external, attention, and device ready interrupts on the virtual machine and inspect and change virtual storage, virtual machine registers, and status words such as the PSW and the CSW. You can use extensive trace facilities for the virtual machine, a single-instruction mode, and commands that invoke the spooling and disk sharing functions of CP.

CP uses multiprogramming techniques to manage the concurrent execution of many virtual machines:

CP overlaps the idle time of one virtual machine with the execution of another by scheduling each virtual machine's access to the processor and "dispatching" each virtual machine for its turn.

CP mimics the response of a System/370 to the actions of the virtual machine.

A virtual machine's configuration matches the components of a real IBM System/370:

- A virtual operator's console
- Virtual storage
- A virtual processor
- Virtual I/O devices.

CP makes these components appear real to the operating system controlling the work flow of the virtual machine.

A virtual machine is configured by describing it in the VM/SP directory. When a user logs on, a virtual machine is created based on information

stored in the user's entry in the directory. The entry for each userid includes:

- A list of the virtual I/O devices associated with the particular virtual machine
- The command privilege class
- Accounting data
- Normal and maximum virtual storage sizes
- Dispatching priority
- Optional virtual machine characteristics (such as extended control (EC) mode).

VM/SP performs some functions differently when running in attached processor or multiprocessor mode. For more information on attached processor and multiprocessor support see Chapter 18, "CP in Attached Processor and Multiprocessor Modes" on page 169.

When instructions in the Control Program are being executed, the real computer is in the supervisor state; when running virtual machines, the real computer is in the problem state. Therefore, privileged instructions cannot be executed by the virtual machine.

A problem program executes on the virtual machine in a manner identical to its execution on a real System/370 processor, as long as the problem program does not violate the CP restrictions. (See *VM/SP Planning Guide and Reference*.) If the virtual machine's virtual PSW indicates that it is functioning in supervisor mode, the privileged instruction is simulated according to its type. If the virtual machine is in problem mode, a program interrupt is reflected to the virtual machine for handling by its operating system. User interrupts, including those caused by trying privileged operations, are handled either by the control program or, if the virtual machine assist feature or VM/370 Extended Control-Program Support is enabled and supports that instruction, by the processor. Only those interrupts that the user program would expect from a real machine are reflected to it.

Chapter 2. Storage Protection

VM/SP provides both fetch and store protection for real storage. The contents of real storage are protected from destruction or misuse caused by erroneous or unauthorized storing or fetching by the program. When a store access is prohibited because of protection, the contents of the protected location remain unchanged. On fetching, the protected information is not loaded into an addressable register, moved to another storage location, or provided to an I/O device.

When a processor access is prohibited because of protection, the operation is suppressed or terminated, and a program interruption for a protection exception takes place. When a channel access is prohibited, a protection-check condition is indicated in the channel status word (CSW) stored as a result of the operation.

When the access to storage is inhibited by the processor, and protection applies, the protection key of the processor occupies bit positions 8-11 of the PSW. When the reference is made by a channel, and protection applies, the protection key associated with the I/O operation is used as the comparand. The protection key for an I/O operation is specified in bit positions 0-3 of the channel-address word (CAW) and is recorded in bit positions 0-3 of the channel status word stored as a result of the I/O operation.

To use fetch protection, a virtual machine must execute the Set Storage Key (SSK) instruction referring to the data areas to be protected, with the fetch-protect bit in the key set on. VM/SP subsequently:

- Checks for a fetch protect violation in handling privileged and nonprivileged instructions
- Saves and restores the fetch protect bit (in the virtual storage key) when writing and recovering virtual machine pages from the paging device
- Checks for a fetch protection violation on a write CCW (except for spooling or console devices).

The CMS nucleus presents a special case for storage protection. The nucleus resides in a shared segment, which must be protected and still shared among many CMS users. To protect the CMS nucleus in the shared segment, user programs and disk-resident CMS commands run with a different key from the nucleus code.

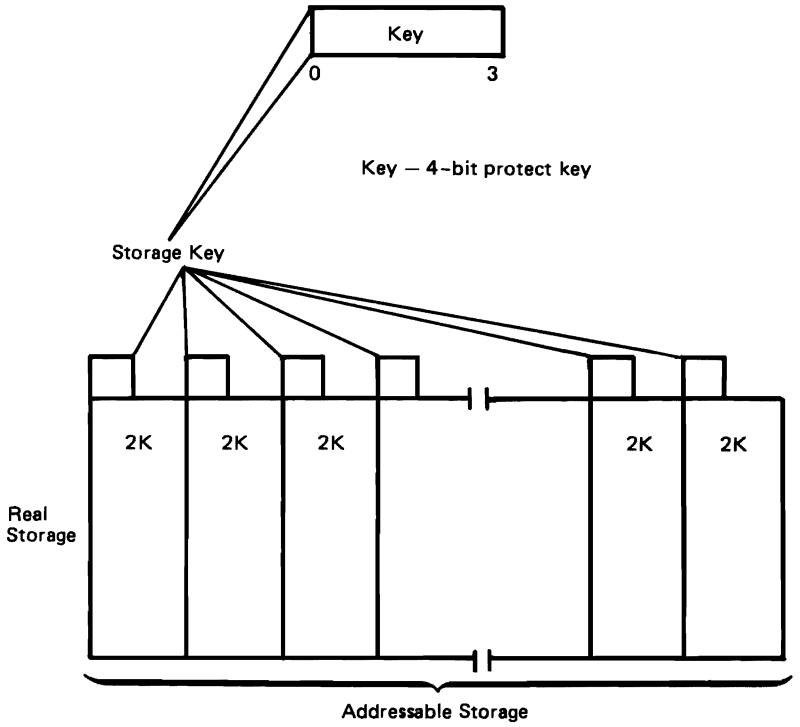


Figure 1. 2K Storage Protection Key

Storage keys protect information in real storage from unauthorized use. A storage key contains a four-bit control field that is associated with an area of real storage. When VM/SP is executing natively, each *2K* area of storage is protected by one storage key.

VM/SP contains support that allows it to execute as a guest virtual machine on a processor that uses single-key real storage frames. Single-key storage frames associate one storage key with each *4K* area of storage. VM/SP does not run natively on processors that have single-key storage frames; however, under control of the VM/SP High Performance Option program product, VM/SP executes as a guest virtual machine operating system.

When VM/SP High Performance Option (Release 2 or subsequent release) is controlling the processor equipped with single key storage frames, it simulates virtual storage for the guest that resembles the type of real storage installed on the processor. If the storage simulated for the VM/SP guest requires 4K storage protection keys, VM/SP issues two key instructions for each storage frame.

Chapter 3. Virtual Storage Preservation

A VM/SP user can choose to preserve the contents of his virtual machine in case:

- The system operator forces the machine off the system.
- The virtual machine is abnormally terminated by VM/SP.
- No shared pages of data to be saved are present in the virtual machine.
- VM/SP abnormally terminates.

This chapter describes how to enable this CP option with:

- The class G command, SET VMSAVE.
- The NAMESYS macro. It defines in the system name table which machines are to be saved, the order in which they are saved, the number of pages to be saved (up to 4096), and the DASD where they will be saved. The NAMESYS macro is described in detail in the *VM/SP Planning Guide and Reference.*

Using the VMSAVE Option

Subject to certain constraints, the user can control whether or not the contents of his virtual machine are saved and which DASD area is used if there is more than one DASD area. If the user has a single DASD area defined, VMSAVE can be enabled either by the VMSAVE directory option or by the SET VMSAVE ON command. A single VMSAVE area can be designated for use by several virtual machines. However, the area is assigned to only one user at a time; the user who first enables VMSAVE has priority. The user releases the VMSAVE area with the SET VMSAVE OFF command.

In contrast, the user with multiple DASD areas enables the VMSAVE option with the SET VMSAVE *name* command. The SET VMSAVE OFF command disables VMSAVE. Also, to release the VMSAVE area, the user must either issue the SET VMSAVE OFF command, or log off, or issue the SET VMSAVE *name* command specifying another area. The DASD save area can be released only by the owner of the data stored in it. To release a DASD area containing a saved system, the owner of the saved area must log on and issue the SET VMSAVE *name* command for that area, then issue either a SET VMSAVE OFF command or a DEFINE STORAGE command, or log off.

The current status of the VMSAVE option (ON or OFF) can be obtained from the QUERY SET command. The QUERY VMSAVE command displays the current status of the VMSAVE option, the names of the areas assigned to the user, the page frames of each area, and the date and time that their contents were saved.

If the VMSAVE option is enabled and an abnormal termination occurs (such as a VM/SP abend and restart), the pages of the virtual machine specified are saved. They are saved in the previously assigned DASD area in the order specified at system generation time by the NAMESYS macro. Afterwards, a logged on user can use the IPL command to bring a page image copy of the saved virtual machine into an active virtual machine. This does not give the saved virtual machine control. The copy can always be dumped; however, it may or may not be executable.

The V=R area of the real machine (if active) is preserved if the system is performing a warm start. The V=R area is cleared if the system terminates to a hard wait state or if a different V=R user logs on.

Setting Priority

The SAVESEQ operand of the NAMESYS macro allows the user to force a priority in the saving order of multiple virtual machines. The priority is determined by number. If two virtual machines have the same priority, and both have the VMSAVE option enabled, they are saved in the order in which they enabled VMSAVE. Disabling and then enabling VMSAVE causes a virtual machine to be last among all the virtual machines having the same SAVESEQ priority value.

If SAVESEQ specifies a high priority for a production virtual machine and lower or equal priorities are specified for other virtual machines, the production machine is saved first; other virtual machines are saved in the order in which the virtual machines logged onto the system.

If a different value of SAVESEQ is specified for each user (the range is 0-255), the order for saving each virtual machine is predictable.

Defining VMSAVE Areas

The VM/SP FORMAT/ALLOCATE program must format DASD space used for VMSAVE areas before any user can store into the area. (Detailed information on using the FORMAT/ALLOCATE program is contained in "Format/Allocate Service Program" on page 122.)

You can specify multiple VMSAVE target areas for a single user; you do this by including in the system name table more than one NAMESYS macro with the same USERID= operand. Different target areas are required if a user wishes to IPL a VMSAVE system and have the VMSAVE option enabled at the same time. Once the VMSAVE is enabled, the area referred

to cannot be referred to by the IPL command until a recovery operation has been effected. Similarly, if a VMSAVE area currently contains a saved system, it can be released only by the user who caused the system to be stored there. That area cannot be the VMSAVE target area referred to by a VMSAVE enable from another user until the stored system has been released.

Specifying Overlapping Areas

The system programmer, at his option, can specify overlapping DASD areas for VMSAVE target areas through NAMESYS macro specifications. However, if two areas overlap, they must start at the same physical cylinder and page. They can end at different locations if the areas are of different lengths. Overlapping areas are useful for different environments of the same user, and they are also valid as VMSAVE target areas for different users.

Only one user can be using the area (for IPL or for a VMSAVE target area) at any one time. In addition, if one user has caused a virtual machine to be stored into an area, no other user can access that area. The user also cannot issue the SET VMSAVE command with that area as the VMSAVE target area, until the user who caused the virtual machine to be stored does the following:

1. Enables VMSAVE to that area via the SET command, which effectively clears the area

2. Releases the area by issuing a SET VMSAVE command to another area command, a SET VMSAVE OFF command, a DEFINE STORAGE command, or a normal LOGOFF command.

Only when the area has been cleared and released in this manner is it available for other users.

For overlapping target areas, the user must load a system that has the same name that it was saved under. This ensures that the page range returned with the load is the same as the one stored by VMSAVE.

Only when the complete page range specified has been saved does the area become valid and available. If an error occurs in the middle of a save operation, the area is not valid, and therefore is not retrievable.

The user cannot force a save directly. The MESSAGE command may be used to ask the operator to force the user off the system. The FORCE command causes an automatic save, assuming that VMSAVE is enabled. The user can also disconnect with a READ pending. After 15 minutes the system logs off the user, causing an automatic save if VMSAVE is enabled.

Saving Other Systems

Systems loaded by name under VM/SP must be saved by the VM/SP SAVESYS command. Because of control block changes, systems saved under other releases of VM/370 are not loaded properly on VM/SP. Conversely, systems saved on VM/SP will not load properly on a system that does not have this product installed.

For details of the SAVESYS command, refer to the *VM/SP CP Command Reference* and the *VM/SP Planning Guide and Reference.*

Chapter 4. I/O Management

This chapter describes how CP manages the I/O requirements of virtual machines in single-, attached-, and multiprocessor environments. A virtual I/O device can be dedicated to an equivalent real device, or shared by several virtual machines, or spooled to a temporary buffer.

Virtual Machine I/O

A real disk device can be shared by several virtual machines. Virtual device sharing is specified in the VM/SP directory entry or by a user command. There are several ways to preserve data integrity on a shared disk.

A user can limit access by defining a password for the virtual device. Virtual machines can be assigned either read-only or read/write access to a shared disk device. CP checks each virtual machine I/O operation against the parameters in the virtual machine configuration to ensure device integrity. Virtual Reserve/Release support can also increase data integrity on shared minidisks. For details on Reserve/Release support, refer to the *VM/SP System Logic and Problem Determination Guide Volume 1 (CP).*

The virtual machine operating system is responsible for the operation of all virtual devices associated with it. These virtual devices can be defined in the VM/SP directory entry of the virtual machine. A user can also attach or detach a virtual device during a terminal session. Virtual devices may be dedicated, shared, or spooled. A dedicated virtual device is mapped to a fully equivalent real device. A shared virtual device is mapped to a minidisk or is specified as a shared virtual device. A virtual device can also be spooled by CP to intermediate direct access storage.

In a real machine, I/O operations are normally initiated when a problem program requests the operating system to issue a START I/O instruction to a specific device. Device error recovery is handled by the operating system. In a virtual machine, the operating system performs these same functions, but the device address and the storage locations are both virtual. It is the responsibility of CP to translate the virtual specifications to real.

In addition, the interrupts caused by the I/O operation are reflected to the virtual machine for its interpretation and processing. If I/O errors occur, CP records them but does not initiate error recovery operations. The virtual machine operating system must handle error recovery, but does not record the error (if SVC 76 is used).

In an attached processor environment, either processor can initiate virtual I/O, but only the main processor has real I/O capability. All real I/O requests must be executed by the main processor, and all I/O interrupts must be received on the main processor. I/O requests by the attached processor are transferred to the main processor.

In a multiprocessor environment, both processors have real I/O capability. If either processor receives an I/O request, that processor tries to initiate I/O operations. If none of the online paths from the executing processor to the required device are available, that processor queues the I/O request on all busy and scheduled paths to the device, including the alternate paths to the device from the second processor. If there is no online path from the executing processor, that processor queues the I/O request on the first online and available path for the second processor as well as on all its own busy or scheduled paths.

Input/output operations initiated by CP for its own purposes (paging and spooling) are performed directly and are not subject to translation.

Virtual machines may access data on mass storage volumes using that virtual machine's standard 3330 device support. MSS cylinder faults, and associated asynchronous interruptions, are transparent to the virtual machine in this situation.

Dedicated Channels

In most cases, the I/O devices and control units on a channel are shared among many virtual machines as minidisks and dedicated devices and shared with CP system functions such as paging and spooling. Because of this sharing, CP has to schedule all the I/O requests to achieve a balance between virtual machines. In addition, CP must reflect the results of the subsequent I/O interrupts to the appropriate storage areas of each virtual machine.

By specifying a dedicated channel (or channels) for a virtual machine via the Class B ATTACH CHANNEL command, the CP channel scheduling function is bypassed for that virtual machine. A virtual machine assigned a dedicated channel has exclusive use of that channel and all of its devices. CP translates the virtual storage locations specified in channel commands to real locations and performs any necessary paging operations, but does not perform any device address translations. The virtual device addresses on the dedicated channel must match the real device addresses; thus, a minidisk cannot be used.

Spool Files

CP spooling facilities allow multiple virtual machines to share real unit record devices. Since virtual machines controlled by CMS ordinarily have low requirements for unit record input/output devices, real device sharing is advantageous, and is the standard mode of system operation.

CP, not the virtual machine, controls the unit record devices that are designated as spooled in the virtual machine directory entry. When the virtual machine issues a START I/O instruction to a spooled unit record device, CP intercepts the instruction and changes it. CP moves data into page-size records (4096-byte blocks) in a disk area that serves as intermediate storage between the real unit record device and the virtual machine.

A virtual unit record device can be dedicated to a real unit. The real device is then controlled completely by the virtual machine's operating system. A virtual machine should not issue a clear channel to any dedicated channel. If the CLRCH instruction is issued, the results are unpredictable.

Spooling operations cease if the direct access storage space assigned to spooling is filled or the virtual unit record devices appear in a not-ready status. The system operator or the spooling operator can make additional spooling space available by purging existing spool files or by assigning additional direct access storage space to the spooling function. The spooling operator can use the class D SPTAPE command to retrieve spool files from tape for output processing when spooling space requirements are not critical. See the description of the SPTAPE command in the *VM/SP CP Command Reference* for further information.

Specific files can be transferred from the spooled card punch or printer of a virtual machine to the card reader of the same or another virtual machine; such files are not physically punched or printed. With this method, files can be made available to multiple virtual machines or to different operating systems executing at different times in the same virtual machine.

Files may also be spooled to remote systems via Remote Spooling Version 2 Communications Subsystem (RSCS) Networking.

CP spooling includes many desirable options for the virtual machine user and the real machine operator. These options include printing multiple copies of a single spool file, backspacing any number of printer pages on unbuffered printers, and defining spooling classes for the scheduling of real output. Each output spool file has, associated with it, a 136-byte area known as the spool file tag. The information contained in this area and its syntax are determined by the originator and receiver of the file. For example, whenever an output spool file is destined for transmission to a remote location via the Remote Spooling Communications Subsystem Networking Version 2, RSCS expects to find the destination identification in the file tag. Tag data is set, changed, and queried using the CP TAG command.

It is possible to spool terminal input and output. All data sent to the terminal from the virtual machine, the control program, or the virtual machine operator, is spooled. Spooling is particularly desirable when a virtual machine is run with its console disconnected. Console spooling is usually started by the command

```
SPOOL CONSOLE START
```

There is an exception to this when a system operator logs on using a graphics device. In this instance, console spooling is automatically started and continues in effect even if the system operator should disconnect from the graphics device and log on to a nongraphic device. To stop automatic console spooling, the system operator must issue the command

```
SPOOL CONSOLE STOP
```

Recovering Spool Files

If the system should suffer an abnormal termination, there are three degrees of recovery for the system spool files: warm start (WARM), checkpoint start (CKPT), and force start (FORCE). Warm start is automatically invoked if SET DUMP AUTO is in effect. Otherwise, the operator can choose the method of recovery at IPL, when this message is presented:

```
Start ((WARM|CKPT|FORCE|COLD) (DRAIN))|(SHUTDOWN) :
```

Note that a cold (COLD) start does not recover any spool files.

Warm Start

After a system failure, the warm start procedure copies spool file, accounting, and system message data either to the warm start area on the IPLed system residence volume (the default) or to the area designated in an alternate nucleus definition. When the system is reloaded, this information is retrieved and the spool file chains and other system data are restored to their original status. If the warm start procedure fails because certain required areas of storage are invalid, the operator is advised to use the checkpoint or force starts.

Checkpoint Start

Any new or revised status of spool file blocks, spooling devices, and spool hold queue blocks is copied to the checkpoint area on the IPLed system residence volume as it occurs. When a checkpoint (CKPT) start is requested, this is the information that is used to recreate the spool file chains. It differs from warm start data in that only spool files are restored; accounting and system messages information is not recovered. Also, the order of spool files on any particular restored chain is not the original sequence but a random one.

Force Start

A force start is required when a checkpoint start encounters I/O errors or invalid data. The procedure is the same as for checkpoint start except that unreadable or invalid files are bypassed.

Chapter 5. Alternate Nucleus Support

You can improve system availability if you define and save multiple versions or copies of your CP nucleus. Then, if the primary nucleus is damaged or unavailable, the system operator can select an alternate nucleus to IPL.

For example, device 140 might contain your primary CP nucleus while 141 contains the alternate nucleus. If IPL 140 fails, or the primary nucleus has a serious error, then the system operator can IPL 141 to bring up the alternate CP nucleus.

The system operator can also use the REIPL option in the SHUTDOWN command to IPL an alternate nucleus.

In addition to maintaining an alternate nucleus, you can also improve your IPL procedure by:

- Sharing the same warm start data, checkpoint data, and error recording data between two or more different versions of CP
- Protecting SPOOL files from hardware failures by defining the warm start area and the checkpoint area on two different disk volumes
- Defining a length in the SYSNUC operand of the SYSRES macro instruction to prevent a nucleus area overflow condition from occurring
- Using the same copy of the DMKSYS ASSEMBLE file for more than one nucleus under the following conditions:
  - SYSCKP, SYSERR, and SYSWRM area are shared
  - The nucleus volumes are compatible with the specified SYSTYPE
  - The nucleus area is defined in the same location on all nucleus volumes.

You can define an alternate nucleus and the other IPL improvements described above by using the options available in the SYSRES macro instruction. Refer to the *VM/SP Planning Guide and Reference* for information about preparing the DMKSYS ASSEMBLE file and for a description of a sample alternate nucleus configuration. The *VM/SP Operator's Guide* describes how the alternate nucleus can be used.

Backup Directories and Override Files

You can also improve system availability by maintaining backup CP directories. Any number of backup directories may be used (one on each SYSOWN volume, if needed). The primary directory is the directory on the current IPL volume. If an error occurred reading the primary directory, CP initialization searches CP-owned volumes in the order in which they are specified in the SYSOWN macro for backup directories.

Backup directories may include, for example, a copy of the primary directory, or a small emergency directory that only you can use to logon and repair any damage to the primary directory.

Use the DIRECT command to maintain directory files. Refer to the *VM/SP Planning Guide and Reference* for information about the VM/SP directory program (DIRECT).

Be aware that if your installation wants to override the IBM-defined classes, an override file must be established on each volume that contains a CP directory. This is because CP uses the override file found on the same volume as the directory. Therefore, when CP is forced to use a backup directory, it loads the override file from the same volume as the backup directory.

Use the OVERRIDE command to maintain override files. (See "Creating a Class Override File" on page 28.) Refer to the *VM/SP Planning Guide and Reference* for information about the override program.

Chapter 6. Changing Commands and Privilege Classes

Each CP command has one or more privilege classes assigned to it. Each user is also assigned one or more privilege classes. The privilege class(es) for each user are stored in the VM/SP directory. If a user tries to issue a CP command in a class that is not assigned to him, the system will not process the command. This prevents users from altering system functions for which they are not authorized.

This chapter describes how to modify the class structure of your system to meet your installation's needs.

Command Privilege Classes and Types

privilege classes and types The privilege class(es) assigned to administrative personnel are determined by the size and configuration of the system and by the installation's particular circumstances. For example, in a small installation one person may be assigned all privilege classes to allow him to handle all administrative tasks. In larger installations several people may be assigned different privilege classes depending on the commands they need to use to do their jobs.

Each version of each CP command is assigned a type code that corresponds to the level of system control that is provided (operations, resource, programming, spooling, analyst, general and CE).

IBM defines privilege classes for each command according to administrative tasks that a typical installation might want to assign to the functions of that command. The IBM-defined privilege classes can be changed by an installation, but the type is permanent. Figure 2 on page 20 shows the relationship of privilege class, type, and administrative function. This relationship is discussed in the text that follows.

IBM Defined Class	User	Functional Type
A	Primary System Operator	Operations Type = O
B	System Resource Operator	Resource Type = R
C	System Programmer	Programmer Type = P
D	Spool Operator	Spooling Type = S
E	System Analyst	Analyst Type = A
F	Service Representative (CE)	CE Type = C
G	Virtual Machine User	General Type = G
ANY	Any user	No type

Figure 2. Relationships of Privilege Classes, Types, and Administrative Functions

The **Operations** administrative function (type O) is assigned to those commands used for primary system operations. The IBM-defined privilege class for these commands is class A.

The **Resource** administrative function (type R) is assigned to those commands used for the distribution of real system resources (such as channels and devices) as they are requested by virtual machine users. The IBM-defined privilege class for these commands is class B.

The **Programmer** administrative function (type P) is assigned to commands used to control trace table information and to commands used to locate, display, print, or change the information in specific storage locations to aid in trouble analysis. The IBM-defined privilege class for these commands is class C.

The **Spooling** administrative function (type S) is assigned to those commands used to control the spool files and certain aspects of the real card readers, punches, and printers. The IBM-defined privilege class for these commands is class D.

The **Analyst** administrative function (type A) is assigned to commands used for monitoring the system resources to ensure that enough resources are available for the virtual machine users. The IBM-defined privilege class for these commands is class E.

The **CE** administrative function (type C) is assigned to commands used for problem determination and problem isolation. These commands allow the service representative to get data about the VM/SP system. The IBM-defined privilege class for these commands is class F.

The **General** function (type G) is assigned to commands used to control the functions associated with a particular virtual machine. The IBM-defined privilege class for these commands is class G.

Some commands have an IBM-defined privilege class of ANY. These commands do not have a type associated with them and may be used by any user. The privilege class for these commands cannot be changed.

For descriptions of all the CP commands, see the *VM/SP CP Command Reference*. For descriptions of DIAGNOSE codes, see *VM System Facilities for Programming*. The coding of the command table in DMKCFC is described in *VM/SP System Logic and Problem Determination Guide Volume 1 (CP)*.

Tailoring the Class Structure

An installation can change the IBM-defined privilege classes to suit its need to control over its resources and information. (This applies to all IBM-defined privilege classes except class ANY, which cannot be changed.) You can define up to 32 classes, A through Z and 1 through 6.

The rest of this chapter describes how to change command and DIAGNOSE code classes, how to change users' access to commands and DIAGNOSE codes, and how to change class access to certain internal CP functions.

Changing Privilege Classes and User Access

To change the IBM-defined privilege classes, you must prepare a file that contains the commands and DIAGNOSE codes for which you want to change the privilege classes. This file is called the **class override file**. Each control statement in the class override file shows the installation supplied class for that particular command.

Because you are changing the existing class structure, you must also change the VM/SP directory to include the newly defined classes.

In general to redefine the class authorization for your system, you must do the following steps (each step is described more fully later in this section):

1. Plan for the effect of the changes.

   - Determine the different kinds of users of your system and what types of administrative functions they should be able to do.

   - List the commands that you want each kind of user to be able to use. If a command is assigned to more than one type, be sure to include the type or types of that command that you want the user to have access to.

- Determine from this which privilege classes you want to associate with each command and type and with each kind of user. Note that if you do not change the class on a command, the class remains the IBM-defined class. For example, if the IBM-defined class for a particular command is A and you do not change it in the override file, it will remain class A.

- Make sure you will not compromise system integrity or system security with these changes.

- Plan for updates to any HELP files caused by the changes.

- Check whether you need to change the classes of any internal functions using the SYSFCN macro (see "How to Change the Privilege Class of Certain Internal CP Functions" on page 36).

2. Make any required class changes to commands:

   a. Create a class override source file on a CMS formatted minidisk to which only the system administrator or other authorized person has access. This same user should have WRITE access to the system resident disk.

   b. A class override file consists of:

      - A DESTINATION control statement followed by,

      - OVERRIDE control statements for those commands and DIAGNOSE codes for which you want to change the IBM-defined classes.

      The DESTINATION statement must be the first control statement in the override file.

   c. Enter the OVERRIDE command with the EDIT operand to validate the class override file.

   d. When you are satisfied that the class override file is correct, enter the OVERRIDE command without the EDIT operand to convert the class override file to an internal format.

   e. Set up a matching CP directory.

   f. To activate the class overrides, IPL the system.

3. Make any required class definition changes to the virtual machine directory:

   - Normally, to assign additional or different classes to a virtual machine, change the CLASS field on the USER control statement.

- If the other parameters on the USER control statement do not leave sufficient space for all of the new classes, place an * in the CLASS field on the USER control statement for that virtual machine and add a CLASS control statement on the next line.

When you want to make additional changes to the class overrides, make the changes or additions to the class override file and enter the OVERRIDE command to convert the changes to internal format as described above. Then IPL the system.

To revert to the IBM-defined classes, enter the OVERRIDE command with the FREE operand and then IPL the system. If your directory was updated specifically for your new class structure, you need to install your original directory when you issue the OVERRIDE command with the FREE operand.

Note: If extensive changes are made in the command structure, you must arrange to update the directory immediately before you IPL with your new override file. Extensive changes in the use of classes A - F might also require an update to the SYSFCN macro in DMKSYS.

Planning the Command Authorization for the System

Before changing the classes of commands, carefully consider the effect of the changes on users and on system integrity. Such changes to the existing command structure will either limit or extend access to system commands. The key elements of this planning are system integrity, system security, and how well these changes enhance your installation's organization and requirements.

Determining Functions to be Done by Users

The first step in restructuring your command classes is to determine the different kinds of users of your system and what types of functions each of these users needs to do the tasks associated with his job. You must consider the existing structure and the users' needs and requirements. This can best be shown in an example.

Consider an Insurance Company where several individuals' job responsibilities and tasks vary. The installation has decided to implement a new class structure. First the users of the system are closely examined to determine their requirements. The system administrator has determined that the users fall into the following categories:

Job Title	Abbrev	Duties
System Administrator	SAD	Responsible for general management of the system and for determining how the system will be structured and used.
System Programmer 1	SP1	Responsible for planning, generating, maintaining, extending, and controlling the use of the operating system with the aim of improving the general productivity of the installation.
System Programmer 2	SP2	Has same responsibilities as System Programmer 1 except that since the system is large, SP1 handles a different portion of the system.
System Analyst 1	SA1	Responsible for analyzing the system to determine what new applications, system programs, and devices are needed by the installation.
System Analyst 2	SA2	Responsible for analyzing the system performance.
Primary System Operators	SO	For each shift, there is a primary system operator who is responsible for ensuring the smooth running of the system and carrying out such duties as changing tapes and disk packs.
Data Base Administrator	DBA	Responsible for resources associated with and access to the main data base of the system. Also responsible for resources associated with spooling, printing, and archiving.
Service Representative	CE	Obtains and examines certain data about input and output devices connected to the system. Also, controls intensive error recording and some machine check error recording.
Experienced Application Programmers	EP	Responsible for developing, testing, and supporting applications to do the work of the company.
Inexperienced Application Programmers	IP	Same as Experienced Application Programmers except that IP develop less sophisticated application programs and therefore do not require access to some functions needed by the more experienced application programmers.
Non-DP Users	U1, U2	Two different types of non-DP users with different requirements identified.

Figure 3. Different System Users and Their Responsibilities

Assigning Commands to Kinds of Users

So far you have determined the kinds of users you have on your system and what types of system functions each user will need to access. Now for each kind of user list all commands that each user will need to do the indicated function. Do not list commands the user does not need or commands whose IBM-defined class is ANY. For those commands that do different functions depending on their assigned type, list the type that corresponds to the functions that you want the user to able to do. For some users, you may want to list more than one type for a particular command.

In our example of the insurance company, one way of doing this is to make a chart like the following one that lists all the commands along the side and the types of users across the top. (Note that the example chart lists only a few commands. You should list all commands.) It will help you to list user types in order by level of system control. You should also include a column for Type (especially if a command has more than one Type) and a column for the new classes to be assigned. The next step is to decide which commands you want each user to be able to use. For the example, asterisks (*) were placed under each user if that user could access the command in the left column.

Command	Type	New Class	SAD	SP1	SP2	SA1	SA2	SO	DBA	CE	EP	IP	U1	U2
ACNT	O					*								
ADSTOP	G										*	*	*	
ATTN	G										*	*	*	
AUTOLOG	O							*	*	*				
CHANGE	S							*	*					
CHANGE	G										*	*	*	*
DCP	P			*	*	*			*					
DEFINE	R							*						
DEFINE	G										*	*	*	*
INDICATE	O							*						
INDICATE	A					*	*							
INDICATE	G										*	*	*	
IPL	G										*	*	*	*
MESSAGE	O							*						
QUERY	O							*						
QUERY	R							*						
QUERY	P			*	*									
QUERY	A					*	*							
QUERY	S							*	*					
QUERY	C									*				
QUERY	G										*	*	*	*
SAVESYS	A		*	*	*									
SPOOL	G										*	*	*	*
DIAG04			*	*	*	*	*							
DIAG1C										*				
DIAG30			*	*	*	*	*							
DIAG38			*	*	*	*	*							
DIAG74			*	*	*									
DIAG84														

Note: DIAGNOSE code X'84' is not marked in any column. This is an example of a function that might be restricted to one or two users.

Associating Privilege Classes with Commands and Users

Once you have associated commands and command types with particular users, you should be able to determine which privilege classes you want to associate with each command and type and with each kind of user.

In our insurance company example, the system administrator could assign a different user class to each type of user. Then, each command could be assigned the list of classes that corresponds to the users who need access. In the chart, each asterisk can be changed to the appropriate user class and copied to the "New Class" field as indicated below:

Command	Type	New Class	SAD A	SP1 B	SP2 C	SA1 D	SA2 E	SO F	DBA G	CE H	EP I	IP J	U1 K	U2 L
ACNT	O	D				D								
ADSTOP	G	IJK									I	J	K	
ATTN	G	IJK									I	J	K	
AUTOLOG	O	FGI						F	G		I			
CHANGE	S	FG						F	G					
CHANGE	G	IJKL									I	J	K	L
DCP	P	BCDH		B	C	D				H				
DEFINE	R	F						F						
DEFINE	G	IJKL									I	J	K	L
INDICATE	O	F						F						
INDICATE	A	DE				D	E							
INDICATE	G	IJK									I	J	K	
IPL	G	IJKL									I	J	K	L
MESSAGE	O	F						F						
QUERY	O	F						F						
QUERY	R	F						F						
QUERY	P	BC		B	C									
QUERY	A	DE				D	E							
QUERY	S	FG						F	G					
QUERY	C	H								H				
QUERY	G	IJKL									I	J	K	L
SAVESYS	A	ABC	A	B	C									
SPOOL	G	IJKL									I	J	K	L
DIAG04		ABCDE	A	B	C	D	E							
DIAG1C	H								H					
DIAG30		ABCDEH	A	B	C	D	E		H					
DIAG38		ABCDE	A	B	C	D	E							
DIAG74		ABC	A	B	C									
DIAG84		1												

As you can see, DIAGNOSE code X'84' is still not available to any of these user groups. However, a few individuals could be given access to this function by assigning class 1 in addition to their normal privilege classes.

You will probably notice that the users with access to system functions and resources (classes A-H) do not have any of the commands that would be useful in controlling their own virtual machine (e.g. SPOOL). Users with classes I - L have varying levels of control over their own virtual machine. This arrangement allows the system administrator to control a user's access to system commands separately from his access to virtual machine commands.

With a change as extensive as this, it is necessary to redefine the privilege classes that control certain internal CP functions. For this example, SYSFCN should be coded:

```
SYSFCN OPER=F,CPRD=BC,CPWT=B,SERV=H,PRIV=ABCF,DFLT=K
```

For an explanation of the parameters to SYSFCN, see "How to Change the Privilege Class of Certain Internal CP Functions" on page 36.

Further Considerations

While customizing the command privilege class structure of your system, you should keep in mind:

Security and System Integrity
With the ability to define the command access to suit your installation's security and system integrity requirements, you have great flexibility and control over each user's access to CP commands. This can be used to enhance security and system integrity at your installation by restricting access to system resources and information controlled by commands or DIAGNOSE codes. However, when you change the privilege class of commands and make changes to user access, be careful not to compromise security or system integrity by allowing users to use commands that could provide access to unauthorized information or that could affect system operation.

Help Files
You may also want to update the HELP files if changes to the command classes affect a type G command. Refer to the *VM/SP CMS User's Guide* for information on tailoring the HELP facility.

Documentation
If you change the privilege class for commands or DIAGNOSE codes, the privilege classes documented in this and other publications for commands and DIAGNOSE codes might not be correct for your installation.

Migration
Altering the VM/SP directory to take advantage of the 32 class command access support will make your directory incompatible with earlier releases of the system.

How to Assign Privilege Classes to Commands and DIAGNOSE Codes

If you want to assign privilege classes other than the IBM-defined classes to certain commands or DIAGNOSE codes (that is, override the IBM-defined privilege classes) you must:

1. Allocate DASD space for the override file. Refer to *VM/SP Planning Guide and Reference* for information on allocating DASD space for the override file.
2. Create a class override file.
3. Verify the syntax of the control statements in the class override file by issuing the OVERRIDE command with the EDIT option.
4. Issue the OVERRIDE command without the EDIT option.
5. IPL the system.

Creating a Class Override File

To override the IBM-defined privilege classes for commands and DIAGNOSE codes, you must first create a class override file. Since VM/SP does not assign a filename to this file, you must assign the name. The default filetype is OVERRIDE. You will specify the name on the OVERRIDE command, which is used to process the class override file and convert it to internal format.

The class override file consists of one DESTINATION control statement followed by an OVERRIDE control statement for each command or DIAGNOSE code whose IBM-defined privilege class is to be overridden. The first statement in this file (the DESTINATION control statement) gives the location of the CP-owned volume that contains the internal override information. The DESTINATION statement has the same syntax as the DIRECTORY statement in the CP directory file. The override space should be on the same volume as the directory, so you could copy the DIRECTORY statement, changing the first term from DIRECTORY to DESTINATION, and use it in the override file.

In the class override file, follow the DESTINATION control statement with the OVERRIDE control statements for the commands and DIAGNOSE codes to be overridden. The format of the OVERRIDE control statement is:

command	**[Type=*c*] Class=** { *classes* } { * }

where:

command
specifies the command or DIAGNOSE code name. It must be the first parameter on the control statement.

Note: VM/SP CP Command Reference lists the CP commands that you can specify on the OVERRIDE control statement. Figure 4 lists the DIAGNOSE codes that you can specify on the OVERRIDE control statement. Only the commands and DIAGNOSE codes listed can have their classes changed. Commands or DIAGNOSE codes defined by the system as class ANY are *not* valid on the OVERRIDE control statement.

Class = classes
*****
specifies the classes to be assigned to this command. This parameter is required. The minimum abbreviation of the keyword is C. *classes* can be from 1 to 32 alphanumeric characters (with no intervening spaces) from A - Z and 1 - 6. Duplicate characters are not allowed and the characters may be in any order. An asterisk (*) specifies that this command or DIAGNOSE code can be executed regardless of the class defined for the virtual machine.

Type=c
specifies the functional type to which the command belongs. (The types correspond to the IBM-defined privilege classes. The classes can be changed but type is permanently associated with the command.) The minimum abbreviation of the keyword is T. This parameter is required only when a command belongs to more than one functional type. For example, QUERY belongs to types O, R, P, S, A, C, and G. *c* specifies one of the following functional types:

O Operations
R Resource
P Programmer
S Spooling
A Analyst
C Customer engineer
G General

The TYPE field is invalid for DIAGnn or DIAGnnn.

DIAGNOSE Code	IBM-Defined Class
DIAG04	CE
DIAG1C	F
DIAG2C	CEF
DIAG30	CEF
DIAG34	CE
DIAG38	CE
DIAG3C	ABC
DIAG50	ABC
DIAG74	ABC
DIAG84	B

Figure 4. DIAGNOSE Instructions That Can Be Respecified on an OVERRIDE Control Stat.

Class Override File Example

Using the example of the insurance company from the section on planning, the override file would contain the following entries:

```
DESTINATION  250  3350  VMSRES
*
* CP COMMAND OVERRIDES FOR 'OUR INSURANCE COMPANY'
*
* USER CLASSES REPRESENT:
*
*        A = SYSTEM ADMINISTRATOR (SAD)
*        B = SYSTEM PROGRAMMER - LEVEL 1 (SP1)
*        C = SYSTEM PROGRAMMER - LEVEL 2 (SP2)
*        D = SYSTEM ANALYST - LEVEL 1 (SA1)
*        E = SYSTEM ANALYST - LEVEL 2 (SA2)
*        F = SYSTEM OPERATOR (SO)
*        G = DATA BASE ADMINISTRATOR (DBA)
*        H = IBM SERVICE REPRESENTATIVE (IBM)
*        I = EXPERIENCED APPLICATION PROGRAMMER (EP)
*        J = INEXPERIENCED APPLICATION PROGRAMMER (IP)
*        K = COMPLEX USERS (U1)
*        L = SIMPLE USERS (U2)
*
*        1 = SPECIAL CLASS ASSIGNED TO DIAGNOSE CODE X'84'
*
* COMMANDS ARE ARRANGED IN BROAD CATEGORIES:
*
*        A-G = SYSTEM CONTROLS OR INFORMATION REQUESTS
*        H   = IBM SERVICE REPRESENTATIVE
*        I-L = VIRTUAL MACHINE OPERATION
*
* NOTES:
*
* (1)    MOST SYSTEM USERS NEED PRIVILEGE CLASS 'I' IN ADDITION TO
*        THEIR PRIMARY FUNCTIONAL CLASS.  THIS ENABLES THE SYSTEM
*        USER TO FULLY CONTROL A VIRTUAL MACHINE.
*
```

```
* (2)     USER CLASS '1' SHOULD BE RESTRICTED TO REDUCE THE POTENTIAL
*         FOR MISUSE OF DIAGNOSE CODE X'84'.
*
ACNT      TYPE=O  CLASS=D
ADSTOP    TYPE=G  CLASS=IJK
ATTN      TYPE=G  CLASS=IJK
AUTOLOG   TYPE=O  CLASS=FGI
CHANGE    TYPE=S  CLASS=FG
CHANGE    TYPE=G  CLASS=IJKL
DCP       TYPE=P  CLASS=BCDH
DEFINE    TYPE=R  CLASS=F
DEFINE    TYPE=G  CLASS=IJKL
INDICATE  TYPE=O  CLASS=F
INDICATE  TYPE=A  CLASS=DE
INDICATE  TYPE=G  CLASS=IJK
IPL       TYPE=G  CLASS=IJKL
MESSAGE   TYPE=O  CLASS=F
QUERY     TYPE=O  CLASS=F
QUERY     TYPE=R  CLASS=F
QUERY     TYPE=P  CLASS=BC
QUERY     TYPE=A  CLASS=DE
QUERY     TYPE=S  CLASS=FG
QUERY     TYPE=C  CLASS=H
QUERY     TYPE=G  CLASS=IJKL
SAVESYS   TYPE=A  CLASS=ABC
SPOOL     TYPE=G  CLASS=IJKL
DIAG04            CLASS=ABCDE
DIAG1C            CLASS=H
DIAG30            CLASS=ABCDEH
DIAG38            CLASS=ABCDE
DIAG74            CLASS=ABC
DIAG84            CLASS=1
```

Note that the TYPE operand must be coded for the DEFINE and QUERY commands because they are associated with more than one type. Note also that to assign more than one class to a command or command type, all new classes are placed on the same override control statement (see the QUERY TYPE=S statement in the above example).

Verifying and Activating the Override File

To verify that the control statements in the class override file have the correct syntax, issue the OVERRIDE command with the EDIT option:

```
OVERRIDE fn ft fm (EDIT
```

where `ft fn fm` are the filename, filetype, and filemode of. the class override file. (Note that the default filetype is OVERRIDE.) If an error is detected, the statement in error is displayed and a message informs you what the error is.

For information on messages, refer to *VM/SP System Messages and Codes*.

After you create the class override file and verify the syntax of the control statements in it, issue the OVERRIDE command with no options to make the new privilege classes effective for the specified commands:

```
OVERRIDE fn ft fm
```

where `ft fn fm` are the filename, filetype, and filemode of. the class override file. If an error is detected, the statement in error is displayed, a message informs you what the error is, and processing of the class override file continues in edit mode but does not write the class override data.

If no errors are detected, the OVERRIDE command converts the class override file to an internal format. This command is similar to the DIRECT command used for converting an external directory source to internal format. If an internal override file already exists, issuing the OVERRIDE command replaces the existing override file with the new one. At this time, however, the new class overrides do not take effect.

To make the new class overrides effective after issuing the OVERRIDE command, IPL the system.

Warning: Restricting the user class on a console command (for example, IPL) does NOT restrict the function of the analogous directory control statement. Thus, a command (such as IPL or LINK) may work at IPL time but not work when issued by the user during his session.

Reverting to the IBM-Defined User Classes

If you want the commands to be assigned their IBM-defined privilege classes again:

1. Issue the OVERRIDE command with the FREE option:

   ```
   OVERRIDE fn ft fm (FREE
   ```

   where `ft fn fm` are the filename, filetype, and filemode of. the class override file.
2. IPL the system.

If, after reverting to the IBM-defined classes, you want to return to the classes you defined in the override file:

1. Enter the OVERRIDE command without the FREE option:

   ```
   OVERRIDE fn ft fm
   ```

2. IPL the system.

Note: If your changes are quite extensive, you may need to install a different directory and/or build a new nucleus with an updated SYSFCN.

How Users Can Find Which Commands They Can Issue

To find out which IBM-defined and user-defined commands are available, the user can issue the COMMANDS command. For example, if the IBM-defined classes are in effect for all commands, a user whose virtual machine is assigned privilege classes E and F would receive the response:

```
LOGON     DIAL      DISCONN   LOGOFF    MESSAGE   SLEEP
*         CP        COMMANDS  DCP       DMCP      INDICATE
MONITOR   PER       QUERY     SAVESYS   SET
DIAG00    DIAG04    DIAG08    DIAG0C    DIAG10    DIAG14
DIAG18    DIAG1C    DIAG20    DIAG24    DIAG28    DIAG2C
DIAG30    DIAG34    DIAG38    DIAG40    DIAG48    DIAG4C
DIAG54    DIAG58    DIAG5C    DIAG60    DIAG64    DIAG68
DIAG6C    DIAG70    DIAG78    DIAG7C    DIAG80    DIAG8C
DIAG94    DIAG98
```

Remember, this would include any commands of class E or F added by the user's installation.

Changing a User's Privilege Classes

The VM/SP directory contains an entry for every virtual machine permitted on the VM/SP system. Each entry includes the privilege class or classes of commands that the virtual machine is permitted to issue. (For additional information about the VM/SP directory and how to generate it, refer to *VM/SP Planning Guide and Reference.*) The control statements in the VM/SP directory that define the command privilege classes for a virtual machine are the USER control statement and the CLASS control statement.

The USER control statement defines a virtual machine and creates a VM/SP directory entry. It identifies the directory entry for one user. You must prepare a separate USER statement for each virtual machine in your system. The format of the USER statement is described in *VM/SP Planning Guide and Reference.* Use the *cl* operand of the USER statement or the CLASS control statement to define privilege classes for the virtual machine. (Coding this operand is described later in this section.) You can define up to 32 classes.

Warning: Make sure you have enough free disk space before editing and making changes to the existing directory so that you can file the updated directory. Refer to *VM/SP Planning Guide and Reference* for information on how to allocate DASD space for the directory.

Directory Entry Example

For the example of the insurance company that we used before the directory might include the following USER and CLASS control statements:

```
DIRECTORY 250  3350  VMSRES
*
* CP DIRECTORY FOR 'OUR INSURANCE COMPANY'
*
* NOTES:
*
* (1)    MOST SYSTEM USERS NEED PRIVILEGE CLASS 'I' IN ADDITION TO
*        THEIR PRIMARY FUNCTIONAL CLASS.  THIS ENABLES THE SYSTEM
*        USER TO FULLY CONTROL A VIRTUAL MACHINE.
*
* (2)    USER CLASS '1' SHOULD BE RESTRICTED TO REDUCE THE POTENTIAL
*        FOR MISUSE OF DIAGNOSE CODE X'84'.
*
* (3)    USERID ALTMAINT IS SET ASIDE FOR EMERGENCY USE.  THIS USER
*        HAS ACCESS TO ANY CP COMMAND NO MATTER WHAT OVERRIDE FILE
*        IS APPLIED.  THE PASSWORD SHOULD ONLY BE KNOWN TO A
*        ***VERY*** FEW KEY PEOPLE.
*
USER ALTMAINT SECRET 2M 8M *
  CLASS ABCDEFGHIJKLMNOPQRSTUVWXYZ123456
  (other control statements)
*
USER ADM XXXXXXXX 2M 8M AI1
  (other control statements)
*
USER SP1 XXXXXXXX 1M 2M BI
  (other control statements)
*
USER SP2 XXXXXXXX 1M 2M CI
  (other control statements)
*
USER SA1 XXXXXXXX 1M 2M DI
  (other control statements)
*
USER SA2 XXXXXXXX 1M 2M EI
  (other control statements)
*
USER SO XXXXXXXX 1M 2M FI
  (other control statements)
*
USER DBA XXXXXXXX 1M 4M GI
  (other control statements)
*
USER IBM XXXXXXXX 1M 4M HI
  (other control statements)
*
USER EP XXXXXXXX 1M 4M I
  (other control statements)
*
USER IP XXXXXXXX 1M 2M J
  (other control statements)
*
USER U1 XXXXXXXX 1M 2M K
  (other control statements)
*
USER U2 XXXXXXXX 512K 1M L
  (other control statements)
```

If no user named "OPERATOR" is defined, DMKSYS should be updated to identify "SO" as the system operator.

Defining Privilege Classes for a Virtual Machine

To change the definition of privilege classes for a virtual machine do the following steps. Step 3 will differ depending on whether you are defining eight or fewer privilege classes, or more than eight privilege classes.

1. Use the XEDIT command to edit the VM/SP directory. The directory will probably have a fileid of USER DIRECT, but it can have any *filename filetype*.

2. Find the USER control statement for the virtual machine whose privilege classes you want to change.

3. This next step is dependent on whether you are defining eight or fewer classes or more than eight classes.

To define eight or fewer privilege classes for a virtual machine: Change the *cl* operand to the classes that you want this virtual machine to have. The *cl* operand consists of one to eight EBCDIC characters (with no intervening blanks) that can be A - Z, and 1 - 6. These characters define privilege classes for the virtual machine. If *cl* is not coded, the default is G. For example, if you want the user whose virtual machine userid is DATABASE to be able to use commands with the privilege classes D, E, L, and M, code the USER control statement as:

```
USER DATABASE pass stor mstor DELM pri le ld cd es
```

Note: For information on coding other operands of the USER control statement, refer to *VM/SP Planning Guide and Reference*.

To define more than eight privilege classes: statement, do the following:

a. Change the *cl* operand to an asterisk (*).

b. Immediately following the USER control statement, insert a CLASS control statement. The format of the CLASS control statement is:

CLASS	classes

where:

classes
specifies up to 32 privilege classes that can consist of any letters from A - Z and any numbers from 1 - 6 with no intervening blanks or commas. Duplicate characters are not allowed. The characters may appear in any order.

For example, if you want to assign privilege classes A through Q to a virtual machine that belongs to a user whose userid is SYSADM,

code the USER control statement and the CLASS control statement as follows:

```
USER SYSADM pass stor mstor * pri le ld cd es
CLASS ABCDEFGHIJKLMNOPQ
```

Note: For information on coding other operands of the USER control statement, refer to *VM/SP Planning Guide and Reference.*

4. After you make all the desired changes to the directory, file the directory file.

5. To verify that the CMS file can be used as a directory file, issue the DIRECT command with the EDIT option. (For the format of the DIRECT command, refer to *VM/SP Planning Guide and Reference.*) If you made a syntax error, an error message informs you of the error.

6. When you have verified that the directory file is correct, replace the old directory with the updated directory. Issue the command:

   ```
   DIRECT filename
   ```

   Note: The virtual machine that issues the DIRECT command must have write access to the volume that will contain the new directory. If you create a directory that is to be written on the active VM/SP system residence volume, your virtual machine's current directory entry must have write access to the volume that contains the current VM/SP directory.

7. Once the directory is updated, directory changes for a virtual machine currently logged on to the system do not take effect until the user logs off the system and then logs back on.

8. If the new directory is written for a new system residence volume, to have the new directory take effect, IPL the system. This causes the new system resident volume to be loaded.

How to Change the Privilege Class of Certain Internal CP Functions

Certain internal functions are preset and need the SYSFCN macro to change them. You can use the SYSFCN macro to change the privilege classes of the following internal CP functions:

- Authorization to logon during CP initialization
- Authorization for intensive recording
- Authorization to issue Diagnostic Load/Write or Sense/Read commands
- Authorization to issue diagnostic reads to a non-dedicated control unit

- Authorization to issue the Buffer Unload command
- Authorization for IOCP Read
- Authorization for IOCP Write

A default SYSFCN macro is supplied in the DMKSYS macro. If you want to change some or all the privilege classes assigned to internal CP functions, you must include a SYSFCN macro statement in DMKSYS that specifies the changes you want to make. The macro format is:

```
[label]     SYSFCN   [ PRIV= {ABCDEF}                ]
                     {classes}
                     [,OPER= {A}                     ]
                     {classes}
                     [,CPRD= {CE}                    ]
                     {classes}
                     [,CPWT= {C}                     ]
                     {classes}
                     [,SERV= {F}                     ]
                     {classes}
                     [,DFLT= {G}                     ]
                     {classes}
```

where:

PRIV specifies the classes authorized to issue X'42' CCW on a 37xx emulation line that is not dedicated to the user. The default classes are A through F.

OPER specifies the classes authorized to logon during initialization. The default class is A.

CPRD specifies the classes authorized to issue IOCP READ. The default classes are C and E.

CPWT specifies the classes authorized to issue IOCP WRITE. The default class is C.

SERV specifies the classes authorized to issue Diagnostic Load/Write and Sense/Read CCW commands. The default class is F.

DFLT specifies the default class or classes for a user who does not have a class defined. The default class is G.

For an example of how our insurance company used the SYSFCN macro, refer back to "Associating Privilege Classes with Commands and Users" on page 26.

Chapter 7. Interrupt Handling

I/O Interrupts

Input/output interrupts from completed I/O operations initiate various completion routines and the scheduling of further I/O requests. The I/O interrupt handling routine also gathers device sense information.

Missing Interrupt Handler

An I/O operation, such as a minidisk operation or a paging operation, that does not complete in a specified time period causes a missing interrupt condition. An incomplete minidisk operation can lock out a virtual machine user or an incomplete paging I/O operation can degrade the performance of the system. The missing interrupt handler detects incomplete I/O conditions by monitoring I/O activity and, in addition, it takes action to correct incomplete I/O conditions without operator intervention. The missing interrupt handler, therefore, is designed to improve the availability of the system by preventing user lockout and system degradation.

The missing interrupt handler scans the real device blocks (RDEVBLOKs) at specified time intervals. If the device is busy (RDEVBUZY flag is on), a bit (RDEVMID) is set that indicates a possible missing interrupt condition. The first level interrupt handler, DMKIOT, resets RDEVBUZY and RDEVMID when the device causes an interrupt at the completion of an I/O operation. Therefore, if RDEVMID is on at the end of the next time interval, a missing interrupt condition exists.

The installation may use the default time interval for each distinct device category or may specify a time value. For example, if the default time interval value of ten minutes for tape devices is not appropriate for an installation's configuration, the installation may change this value. See "Changing the Time Interval" on page 41 for a list of the default time interval values and how you can change these values.

Using the Missing Interrupt Handler

To use the Missing Interrupt Handler (MIH), DMKDID must be included in the load list during system generation. MIH can be set on either by including it as an option in the directory or by issuing the SET command. The default is MIH OFF. With MIH on, when a missing interrupt is detected, CP simulates the interrupt. With MIH off, when a missing interrupt is detected, message DMKDID546I is issued but CP does not simulate the interrupt. If DMKDID is deleted from the load list during system generation, support for the Missing Interrupt Handler is removed and no messages are written to notify the operator of a missing interrupt.

If you want to change the interval time value, you must include the optional macro SYSMIH in the system control file (DMKSYS). You must place this macro before the SYSLOCS macro.

When a missing interrupt occurs, the control program tries to correct the condition and issues a message that either the condition is cleared or the condition is pending. This message warns the system operator or system programmer that a problem may exist. The system operator or the system programmer can reset the hardware and schedule maintenance for the device that caused the missing interrupt condition. If the same device class causes frequent interrupts, the system programmer may want to set a larger time interval for that particular device class.

The class G SET command can be used to turn MIH on and off. Use either

SET MIH ON or SET MIH OFF

To determine the status of MIH, use

QUERY SET

The response is

MIH ON or MIH OFF

Devices Monitored

Each device group has an expected time interval during which an I/O operation should be completed. This interval varies widely among devices. Therefore, the missing interrupt handler provides a means to specify a time interval for the following distinct categories of I/O devices:

- Count-key-data devices (CLASDASD) and FB-512 devices (CLASFBA)
- Tape devices (CLASTAPE)
- Graphic devices (CLASGRAF) except TYP1053 and TYP328X
- Unit record devices (CLASURI and CLASURO) except TYP3800 and TYP3289E

- Miscellaneous devices (MISC) include: Mass storage system (MSS) devices (specified at system generation as CLASSPEC TYP3851, and CLASDASD FEATURE = VIRTUAL or FEATURE = SYSVIRT), graphics devices TYP1053 and TYP328X, and UR output devices TYP3800 and TYP3289E.

Note: The missing interrupt handler does not support terminal devices, remote graphic devices, SNA devices, pass-through virtual machine (logical) devices, and special class devices (with the exception of MSS).

Changing the Time Interval

An installation may want to change the default time intervals because of their particular configuration. For example, an installation that generates a large number of devices might want to set the time interval value to a larger number to prevent frequent timer interrupts.

Device Class	Class Parameter	Default Time Interval
CLASDASD or CLASFBA	DASD	15 seconds
CLASGRAF	GRAF	30 seconds
CLASTAPE	TAPE	10 minutes
CLASURI/CLASURO	UR	1 minute
MISCELLANEOUS	MISC	12 minutes

The system programmer or the system operator can change the time interval in the following ways:

- Regenerate the system and, using the SYSMIH macro, specify a time interval value in the system control file (DMKSYS) for the specific device class to be changed. Specify the time interval value in minutes and seconds:

  ```
  SYSMIH GRAF=00:15,UR=00:00,TAPE=05:00
  ```

  This example changes the time interval for graphic devices from the default value of thirty seconds to fifteen seconds. In this example, no further monitoring takes place for unit record devices since the user specified a time value of zero for that class. In addition, the example changes the time interval value for tape devices from ten minutes to five minutes. This example does not change the time interval value for DASD and MISC devices. If you do not specify a device class, or if you do not include the SYSMIH macro in DMKSYS, the missing interrupt handler uses the default value for that class.

- To change the value specified in DMKSYS for a particular device class, issue the class B CP command specifying the new time interval value for that class in minutes and seconds:

  ```
  SET MITIME GRAF 00:10
  ```

This example changes the time interval for graphic devices to ten seconds. This change is in effect until the system is reinitialized, or until a class B user issues another SET MITIME command. If the user specifies a time value of zero for a specific device class, no further monitoring takes place for that device class.

Note: If you set the time interval for a device class below its default value, be careful not to shorten the time interval too much. This may cause unnecessary missing interrupt handler processing for devices that are functioning properly.

- To set all time values to zero and to prevent any monitoring for missing interrupts for any devices, issue the class B CP command:

  ```
  SET MITIME OFF
  ```

  Monitoring for missing interrupts does not take place until the system is reinitialized, or until the class B user issues another SET MITIME command.

Determining Time Interval Settings

The class B user can determine the current missing interrupt handler time intervals by issuing the following CP command:

```
QUERY MITIME
```

The system makes one of these four responses:

- The time interval setting for each device group in minutes and seconds
- The response MITIME OFF
- An error message if the user specified an invalid parameter
- The response that the missing interrupt handler is not available if DMKDID is not in the load list during system generation.

Diagnostic Aids

Missing interrupt handler support provides aids so that the system programmer can determine the frequency and status of interrupts and also know when he has made an error in using the support. Diagnostic aids available when using the missing interrupt handler include:

- System messages
- Macro notes
- VM/SP system's error recording area
- Trace table.

System messages
: Messages inform the system operator when a missing interrupt occurs and indicate if the condition has been cleared or if the interrupt is still pending. Other messages indicate that the module DMKDID is not in the load list or that the user specified an invalid parameter on the QUERY or SET MITIME command. See *VM/SP System Messages and Codes* for a complete discussion of messages that the missing interrupt handler issues.

  The system programmer can use message information to increase the availability of the system. If a particular device class causes frequent interrupts even if the system clears the condition, the system programmer may want to change the time interval. Changing the time interval prevents the overhead of frequent timer interrupts, frequent trips through the detector routine, and rescheduling of timer request queues. On the other hand, if the control program did not clear the condition, the messages make the system programmer or system operator aware of the condition and one of them can reset the hardware either physically or using CP commands.

Macro notes
: Macro notes (MNOTES) inform the user that SYSMIH is not present in DMKSYS or that the user specified an invalid time value in the SYSMIH macro. The system uses the default interval time values and informs the user.

System's Error Recording Area
: Whether or not CP succeeds in correcting a missing interrupt situation, it creates a record of the event in the system's error recording area (LOGREC).

Trace table
: CP also traces the simulated interrupt and records it as trace table entry X'19'. The system programmer uses the trace table to determine the events that preceded a CP system failure. There is a figure that shows the format of CP trace table entries in *VM Diagnosis Guide.*

Program Interrupt

Program interrupts can occur in two states. If the processor is in supervisor state, the interrupt indicates a system failure in the CP nucleus and causes the system to abnormally terminate. If the processor is in problem state, a virtual machine is executing. CP takes control to perform any required paging operations to satisfy the exception, or to simulate the instruction. The fault is transparent to the virtual machine execution. Any other program interrupt is a result of the virtual machine processing and is reflected to the machine for handling.

Machine Check Interrupt

When a machine check occurs, the CP Recovery Management Support (RMS) gains control to save data associated with the failure for the Field Engineer. RMS analyzes the failure to determine the extent of damage.

Damage assessment results in one of the following actions being taken:

- System termination (CP disabled wait state)
- Attached processor disabled (system continues in uniprocessor mode)
- One processor of a multiprocessor configuration disabled (system continues in uniprocessor mode)
- One or more failing channels disabled (system continues in same mode as at time of the error)
- Selective virtual user termination
- Selective virtual machine reset
- Refreshing of damaged information with no effect on system configuration
- Refreshing of damaged information with the defective storage page removed from further system use
- Error recording only for certain soft machine checks.

The system operator is informed of all actions taken by the RMS routines. When a machine check occurs during VM/SP startup (before the system is sufficiently initialized to permit RMS to operate successfully), the processor goes into a disabled wait state and places a completion code of X'00B' in the leftmost bytes of the current PSW.

SVC Interrupt

When an SVC interrupt occurs, the SVC interrupt routine is entered. If the machine is in problem mode, the type of interrupt (if it is other than an SVC 76 or ADSTOP SVC) is reflected to the pseudo-supervisor (that is, the supervisor operating in the user's virtual machine). Control is transferred to the appropriate interrupt handler for ADSTOP SVCs and all SVC 76s.

If the machine is in supervisor mode, the SVC interrupt code is determined, and a branch is taken to the appropriate SVC interrupt handler.

External Interrupt

If a timer interrupt occurs, CP processes it according to type. The interval timer indicates time slice end for the running user. The clock comparator indicates that a specified timer event occurred, such as midnight, scheduled shutdown, or user event reached.

The external console interrupt invokes CP processing to switch from the 3210 or 3215 to an alternate operator's console.

A service signal is a class 24 external interrupt that is generated when:

- A logical device signals completion of an operation initiated by a program (DIAGNOSE X'7C'), or
- The Maintenance and Service Support Facility (MSSF) signals completion of an operation initiated by CP (MSSFCALL DIAGNOSE X'80').

Descriptions of DIAGNOSE codes are in *VM System Facilities for Programming*. *IBM System/370 Principles of Operation,* contains a general description of external interrupts.

Synchronous Interrupts with Multiple Processors

Generally, when synchronous interrupts (such as program and SVC interrupts) occur in an attached processor or multiprocessor system, the processing of the interrupt can proceed without the global system lock for mainline, nonerror paths. Otherwise, the global system lock is required. If the global system lock is needed and it is already in use, the processing of the interrupt is deferred until the global system lock is available. In this case, the interrupted processor tries to run another user.

Real I/O Interrupts

In an attached processor configuration, only the main processor can receive real I/O interrupts. To ensure this, the channel masks in control register 2 on the main processor are initialized to ones to enable interrupts from any available channel. On the attached processor, the channel masks in control register 2 are initialized to zeroes. In a multiprocessor configuration, both processors can receive real I/O interrupts. The channel masks in control register 2 on both processors are initialized to ones to enable interrupts from any available channel.

Chapter 8. Accounting Records

The accounting data gathered by VM/SP can help in analysis of overall system operation. Also, accounting data can be used to bill VM/SP users for time and other system resources they use.

There are three types of accounting records: the virtual machine user records, records for dedicated devices and T-disk space assigned to virtual machine users, and accounting records generated by a DIAGNOSE X'4C' instruction. A CMS batch virtual machine creates an accounting record with the userid and account number of the user who sent the job to the batch machine.

Accounting records are prepared as 80-character card images and usually spooled to disk. There are two ways to send the data to the punch for punched output or spool it to the user's reader for additional processing. The SYSACNT macro does it when a specified number of records have accumulated. The ACNT CLOSE command does it immediately.

Accounting Records for Virtual Machine Resource Use

The information stored in the accounting record in card image form when a user ends his terminal session (or when the ACNT command is invoked) is as follows (columns 1-28 contain character data; all other data is in hexadecimal form, except as noted):

Column	Contents
1- 8	Userid
9-16	Account number
17-28	Date and Time of Accounting (mmddyyhhmmss)
29-32	Number of seconds connected to VM/SP System
33-36	Milliseconds of processor time used, including time for VM/SP supervisor functions
37-40	Milliseconds of virtual processor time used
41-44	Number of page reads
45-48	Number of page writes
49-52	Number of virtual machine SIO instructions for nonspooled I/O
53-56	Number of spool cards to virtual punch
57-60	Number of spool lines to virtual printer (This includes one line for each carriage control command)
61-64	Total number of spool records from virtual reader (This is not the number of records read, rather it is the total number of records in the spool file (SFBRECNO) when the file is open for processing.)
65-78	Reserved

79-80	Accounting record identification code (01)

Accounting Records for Dedicated Devices and Temporary Disk Space

Accounting records are recorded and spooled to disk when a previously dedicated device and temporary disk space is released by a user via DETACH, LOGOFF, or releasing from DIAL (dedicated device only). A **dedicated device** is any device assigned to a virtual machine for that machine's exclusive use. These include devices dedicated by the ATTACH command, those being assigned at logon by directory entries, or by a user establishing a connection (via DIAL) with a system that has virtual 2702 or 2703 lines. The information on the accounting record in card image form is as follows (columns 1-28 contain character data; all other data is in hexadecimal form, except as noted):

Type 02

Column	Contents
1- 8	Userid
9-16	Account number
17-28	Date and Time of Accounting (mmddyyhhmmss)
29-32	Number of seconds connected to VM/SP system
33	Device class
34	Device type
35	Model (if any)
36	Feature (if any)
37-64	Unused
65-72	Terminal identification
73-78	Unused
79-80	Accounting record identification code (02)

Type 03

Column	Contents
1- 8	Userid
9-16	Account number
17-28	Date and Time of Accounting (mmddyyhhmmss)
29-32	Number of seconds connected to VM/SP system
33	Device class
34	Device type
35	Model (if any)
36	Feature (if any)
37-38	Number of cylinders of temporary disk space used (if any) or number of blocks used (columns 37-40) for fixed-block devices.
39-78	Unused (columns 41-78 unused for fixed-block devices)
79-80	Accounting record identification code (03)

VM Diagnosis Guide has a figure that shows the codes that go in columns 33-36 (device class, device type, model, and feature).

Accounting Records for LOGON, AUTOLOG, and LINK Journaling

When LOGON, AUTOLOG, and LINK journaling is on, VM/SP may write type 04, type 05, type 06, type 07, or type 08 records to the accounting data set. These records are written under the following circumstances:

- Type 04 records are written when VM/SP detects that a user has issued enough LOGON or AUTOLOG commands with an invalid password to reach or exceed an installation defined threshold value.
- Type 05 records are written when VM/SP detects that a user has successfully issued a LINK command to a protected minidisk not owned by that user.
- Type 06 records are written when VM/SP detects that a user has issued enough LINK commands with an invalid password to reach or exceed an installation defined threshold value.
- Type 07 records are written when a user logs off a device controlled by the VTAM Service Machine (VSM). The records indicate the user's share of the VSM resources used.
- Type 08 records are written when a user disconnects or logs off.

These records have the following formats:

Type 04

Column	Contents
1- 8	USERID specified on the command
9-16	Reserved for IBM use
17-28	Date and time of accounting (mmddyyhhmmss)
29-32	Terminal address (see Note 1)
33-40	Invalid password (see Note 2)
41-48	USERID that issued the AUTOLOG command
49-51	Reserved for IBM use
52-53	Current invalid password count
54-55	Accounting record limit (JPSLOGAR)
56-70	Reserved for IBM use
71-78	LUNAME for SNA terminal
79-80	Accounting card identification code (04)

Type 05

Column	Contents
1- 8	USERID that issued the command
9-16	Account number
17-28	Date and time of accounting (mmddyyhhmmss)
29-32	Terminal address (see Note 1)
33-40	Reserved for IBM use
41-48	USERID of user that owns the minidisk
49-51	Minidisk address for which the LINK command was issued
52-70	Reserved for IBM use
71-78	LUNAME for SNA terminal
79-80	Accounting card identification code (05)

Type 06

Column	Contents
1- 8	USERID that issued command
9-16	Account number
17-28	Date and time of accounting (mmddyyhhmmss)
29-32	Terminal address (see Note 1)
33-40	Invalid password (see Note 2)
41-48	USERID of user that owns the minidisk
49-51	Minidisk address for which the LINK command was issued
52-53	Invalid password count
54-55	Invalid password limit (JPSLNKAR)
56-70	Reserved for IBM use
71-78	LUNAME for SNA terminal
79-80	Accounting card identification code (06)

Type 07

Column	Contents
1- 8	USERID or terminal identification
9-16	Accounting number or 0000
17-78	VTAM Service Machine accounting data
79-80	Accounting card identification (07)

Notes:

1. *For the terminal address, columns 29-32 may contain one of the following:*

   - *'NONE'—if no terminal is found*
   - *resource id—for remote bisynchronous terminals*
   - *real device addr—for all other cases.*

2. *For the invalid password, columns 33-40 may contain one of the following:*

   - *incorrect password*
   - *'TERM/ERR' - if the line dropped during password entry*
   - *'TOO LONG' - if entered password is more that eight characters.*

Type 08

Column	Contents
1- 8	userid
9-16	Account number
17-28	Date and time of accounting (mmddyyhhmmss)
29-64	Reserved for IBM use
65-72	Terminal identification
73-78	Reserved for IBM use
79-80	Accounting record identification code (08)

Accounting Records Created by the User

A virtual machine user can initiate the creation of an accounting record that contains up to 70 bytes of information of his own choosing. To do this, he issues a DIAGNOSE code X'4C' instruction with the following operands:

- The address of a data area in virtual storage containing the information, in the actual format, that he wishes to have recorded in columns 9 through 78 of the card image record.
- A hexadecimal function code of X'10'
- The length of the data area in bytes

The information on the accounting record is as follows:

Column	Contents
1- 8	Userid
9-78	User formatted data
79-80	Accounting record identification code (C0)

For information on using DIAGNOSE code X'4C' see *VM System Facilities for Programming*.

For SNA users, the VTAM Service Machine (VSM) uses the VM/SP user accounting record. See the *VCNA Installation and Terminal Use Guide* for the format of this record.

The Transparent Services Access Facility uses the VM/SP user accounting record. See *VM/SP Transparent Services Access Facility Reference* for the format of this record.

User Accounting Options

You may insert your own accounting procedures in the accounting routines. See Chapter 10, "CP Conventions" for information on CP coding conventions and load list requirements. Operator responsibilities in such cases should be defined by the installation making the additions. When designing such accounting procedures, you should understand that:

1. The accounting routines are designed to be expanded. The entry point provided in the accounting module for installation use is called DMKACON. If you want to perform additional accounting functions, you should modify the following copy files:

   ACCTON (account on) -- for action at logon time. This is provided as a null file. It can be expanded to provide additional functions at logon time. The ACCTON routine can request the system to force the user off by returning a nonzero value in SAVER2. However, if the operator is automatically logged on during system initialization, the nonzero return code has no effect.

   Note: The ACCTON COPY file distributed with VM/SP contains the basic logic required to enhance system security based on the 3277 Operator Identification Card Reader feature. Additional checking may be added to examine or validate the data read from the identification card.

   ACCTOFF (account off) -- for action at logoff time. This section contains the code that fills in the account card fields. It does not reset any internal data. This file exists in both DMKACO and DMKCKF (checkpoint). If the ACCTOFF copy file is changed, both modules should be reassembled.

2. In addition to CP accounting, your installation can use the accounting routines to supply virtual machine operating system accounting records. This provides a means of job accounting and operating system resource use accounting.

3. If you specify, in the SYSACNT system generation macro, that your spooled accounting records are to be sent to the reader of a virtual machine, you can process the accounting data directly with your own accounting routines.

Chapter 9. Saved Systems, Discontiguous Saved Segments, and Shared Segments

Saved systems are systems you can IPL in a virtual machine, initialize, and save on a disk along with all the information you need to resume execution at the point where you save the system. Saved systems provide an efficient means of IPLing systems by bypassing many system initialization steps.

Discontiguous saved segments (DCSS) are areas of virtual storage outside the address range of a virtual machine. These segments can contain read-only data or reentrant code. Discontiguous saved segments provide an efficient means of fetching programs by merely connecting discontiguous segments to a virtual machine's address space.

Shared segments definition are segments within a saved system or DCSS. These segments can contain read-only data or reentrant code that many users can share. Many users can share all or portions of a saved system or DCSS. This reduces the demand for real storage for the overall system.

A segment of storage is 64K bytes long on a 64K byte boundary.

The *VM/SP Planning Guide and Reference* contains more information on:

- Saved systems, discontiguous saved segments, and shared segments
- Using the CP SAVESYS command
- Creating a system name table
- Coding the NAMESYS, NAMENCP, and NAME3800 macros.

The *VM/SP Installation Guide* contains more information on:

- Loading and saving discontiguous saved segments
- Using the SPGEN EXEC to reassemble DMKSNT.

Before a discontiguous saved segment can be attached and detached by name, it must be loaded and saved. The load address of the named segment must be beyond the highest address of any virtual machine to which it will be attached. Once the named segment is loaded at the correct address, you can save it by issuing the CP SAVESYS command.

The system programmer should make sure the named segment is loaded at an address that does not overlay the defined virtual machine or any other named segment that may be attached at the same time.

To be sure that the CMS discontiguous saved segment has segment protection, set the storage key for the segment to something other thanX'F' before you save it. Use the CMS SETKEY command to change the storage key.

CP DIAGNOSE Code Interface With A DCSS

The linkage to attach and detach discontiguous saved segments is supported through several CP DIAGNOSE codes.

The virtual machine is responsible for insuring that the discontiguous saved segment it is attaching does not overlay other programming code. To do this, the virtual machine must know how much virtual storage it has. By issuing DIAGNOSE code X'60' during its initialization process, the virtual machine can determine its virtual machine storage size.

When the virtual machine needs to attach a discontiguous saved segment, it must first ensure that the segment is available and that it does not overlay existing storage. By issuing the DIAGNOSE codeX'64' with subcode X'000C', the virtual machine can verify that a loadable copy of the discontiguous saved segment exists on a CP-owned volume. This DIAGNOSE code is called the FINDSYS function. FINDSYS returns the starting address of the segment. The virtual machine should compare the starting address of the segment to its own ending address; if the segment does not overlay existing storage, it can be loaded.

CP DIAGNOSE code X'64' with subcode X'0000' orX'0004' provides a LOADSYS function. Subcode X'0000' loads a named segment in shared mode, and subcode X'0004' loads a named segment in nonshared mode. *VM System Facilities for Programming* contains a complete description of the DIAGNOSE codes used in the discontiguous saved segment interface.

To load the named segment in nonshared mode, use the CMS command:

```
SET NONSHARE segmentname
```

before CMS attaches the named segment. If the segment is loaded in nonshared mode you can test and debug it using the CP TRACE, STORE,

and ADSTOP commands and the CMS DEBUG subcommands BREAK and STORE.

When CMS loads a named segment in shared mode, it issues the CP DIAGNOSE code X'64' with subcode X'0000'. CMS issues the same code with subcode X'0004' to load the named segment in nonshared mode.

When a discontiguous saved segment is loaded (or attached) to a virtual machine, CP expands its segment table entries for that virtual machine to reflect the highest address of the virtual machine.

When a named segment is successfully loaded, all of its storage is addressable by the virtual machine. For example, when CMS attaches a named segment, it can execute the routines contained in that segment. All of the commands that are executable for CMS are also executable for the attached named segment, with the following exceptions:

- The response for the CP QUERY VIRTUAL STORAGE command does not reflect the storage occupied by the named segment.
- If you execute a command that alters storage (such as STORE), you are given a nonshared copy of the named segment.

When the named segment is no longer needed, it can be detached. The CP DIAGNOSE code X'64' subcode X'0008' is called the PURGESYS function; it detaches named segments. When a named segment is detached, its storage is no longer addressable by the virtual machine and CP updates its segment tables. The entries for segments beyond the original virtual machine size are deleted and the associated real storage is released.

Shared Segment Protection

VM/SP protects shared segments by default. However, at system generation time, the system programmer can designate whether a shared segment is to be protected or not. To do this, the programmer uses the PROTECT operand of the NAMESYS MACRO. (See *VM/SP Planning Guide and Reference* for details of coding the NAMESYS macro.)

Generally, the information contained in a protected shared segment should not be modified. When segments are protected, CP ensures that one virtual machine does not access a shared segment that another virtual machine has changed. In addition, CP does not allow any user to change the storage keys on the protected page, thus preventing other users from accessing the information on that page.

If a named system is specified as protected, segment zero must not be shared. Page zero in this segment contains areas that may change (such as PSWs) and sharing it in protected mode can have unpredictable results.

Unprotected shared segments differ from protected shared segments in that they contain data that can be modified by any user that accesses the shared

segment. CP takes no action to protect either the contents or accessibility of these pages. When segments are not protected, CP lets one virtual machine access a page in the shared segment that may have been changed by another virtual machine. As a result, all the virtual machines that share that storage must be aware of the change activity that can occur and must act accordingly.

In addition, CP allows a user to change the storage keys on an unprotected page by using the privileged instruction SSK. Changing the storage keys can prevent users from accessing storage on the shared page; however, CP only simulates a privileged instruction for a user in virtual supervisor state. Therefore, only a user in virtual supervisor state is able to change the keys on unprotected shared pages.

Virtual Machine Operation with Protected Segments

When dealing with protected shared segments, CP determines if the current virtual machine altered any pages within a segment before it dispatches another virtual machine. Altering a page causes CP to take additional action before dispatching the next virtual machine. The action that CP takes depends on what the virtual machine did to alter the protected page.

The current virtual machine may have altered a protected shared page by issuing one of the following commands:

- CP TRACE
- CP ADSTOP
- CP STORE

In this instance CP gives exclusive use of the modified page to the virtual machine that modified it. The user is given his own copy of the shared system that contained the altered page. The user who issued the command receives the message:

```
DMKATS181E Shared system name replaced with non-shared copy
```

This user's virtual machine continues to execute using the private copy of the shared system which contains the changes that were made to the page. CP provides an unmodified copy of the page for other virtual machines to share.

The current virtual machine may have altered a protected shared page as a result of issuing the STCP command. When STCP is issued, CP does not assign the modified page to the user issuing the STCP command. Instead, the page changed by the STCP command is written to the paging volumes. As a result, the change made by the single user reflects to all the virtual machines using that shared page.

If operations overlap and a STCP command is issued for a shared page that is about to be assigned to a particular user (because that user just altered it), the user that issued the STCP command receives the following message:

```
DMKCDS161E Shared page hexloc altered by userid
```

It should be noted that it is invalid to issue the STCP command to a shared segment in attached processor systems. The store function is not performed and the user receives the following message:

```
DMKCDS004E Invalid hexloc - xxxxxx
```

If the current virtual machine alters a protected shared page in any other way, then the following happen:

1. CP sends this message to the current virtual machine to identify the altered page:

   ```
   DMKVMA456W CP entered; sysname - shared page hexloc
   altered
   ```

2. CP frees the storage occupied by the page, thus making it inaccessible. Later, when a virtual machine refers to the page, CP brings a fresh copy of the page into storage.

3. CP places the current virtual machine into console function mode, thereby stopping the virtual machine. To resume execution, the operator of this virtual machine must issue the class G BEGIN command. The virtual machine then continues to execute the unaltered system in shared mode.

4. CP then dispatches another virtual machine.

I/O activity into protected shared segments is monitored by channel program translators. A channel protection error occurs if a virtual machine tries to read data into a protected page. A virtual machine is able to write from a page in a protected segment. Shared systems contain segments that are either protected or non-protected. No distinction is made between shared and nonshared systems for storage key fetch instruction simulation, DISPLAY command execution, and page key handling. In addition, the Extended Control Program Support (ECPS) and the Virtual Machine Assist feature (VMA) are available to users running shared systems, except that there is no microcode assist for the SSK instruction. This exception is necessary because VMA updates the key on SSK instructions (including SWPTABLE fields), but the new value is not detected by the hardware change bit monitoring.

A single bit in control register 6 determines whether the ISK (Insert Storage Key) and SSK (Set Storage Key) instructions are handled by the VMA feature. The dispatcher sets up control register 6 based on the type of system that the virtual machine is running. If the virtual machine is running a shared system of any kind (either protected or unprotected) then the control registers are set up so that the SSK instruction is not microcode assisted. Otherwise, the dispatcher sets up control register 6 so that the SSK instruction is performed by the VMA feature.

Chapter 10. CP Conventions

This chapter describes the conventions followed in the source code for CP and the proper order of modules in the loadlist.

CP Coding Conventions

The following are coding conventions used by CP modules:

- FORMAT:

Column	Contents
1	Labels
10	Op Code
16	Operands
31, 36, 41, etc.	Comments (see Item 2)

- COMMENT:

  Approximately 75 percent of the source code contains comments. Sections of code performing distinct functions are separated from each other by a comment section.

  Constants follow the executable code and precede the copy files and/or macros that contain DSECTs or system equates. The section defining constants is followed by a section containing initialized working storage, which is followed by working storage. Each of these sections is identified by a comment. As much as possible in modules that are more than than a page long, a reference to a constant or to working storage is on the same page as the definitions.

- No program modifies its own instructions during execution.

- No program uses its own unlabeled instructions as data.

- REGISTER USE:

  For CP, in general

Register	Use
6	RCHBLOK, VCHBLOK
7	RCUBLOK, VCUBLOK
8	RDEVBLOK, VDEVBLOK
10	IOBLOK
11	VMBLOK

12	Base register for modules called via SVC
13	SAVEAREA for modules called via SVC
14	Return linkage for modules called via BALR
15	Base address for modules called via BALR

For Virtual-to-Real address translation

Register	Use
1	Virtual address
2	Real address

- When describing an area of storage in mainline code, a copy file, or a macro, DSECT is issued containing DS instructions.

- Meaningful names are used instead of self-defining terms: for example, 5,X'02',C'I' to represent a quantity (absolute address, displacement, length, should be symbolic and defined by an equate (EQU) assembler instruction. For example:

```
VMSTATUS  EQU  X'02'
```

To set a bit, use:

```
OI  BYTE,BIT
```

where BYTE = name of field, BIT has been defined by an EQU instruction.

To reset a bit, use:

```
NI  BYTE,X'FF'-BIT
```

To set multiple bits, use:

```
OI  BYTE,BIT1+BIT2
```

All registers are referred to as:

```
R0, R1, ..., R15.
```

All lengths of fields or control blocks are symbolic, that is, length of VMBLOK is:

```
VMBLOKSZ  EQU  *-VMBLOK
```

- Avoid absolute relative addressing in branches and data references, (that is, location counter value (*) or symbolic label plus or minus a self-defining term used to form a displacement).

- When using a single operation to refer to multiple values, specify each value, for example:

```
LM  R2,R4,CONT    SET R2=CON1
                  SET R3=CON2
                  SET R4=CON3
```

```
.
.
.
CON1    DC    F'1'
CON2    DC    F'2'
CON3    DC    F'3'
```

- Do not use PRINT NOGEN or PRINT OFF in source code.
- MODULE NAMES:

  Control Section Names and External References are as follows:

  Control Section or Module Name

  The first three letters of the module name are the assigned component code.

  ```
  Example:  DMK
  ```

  The next three letters of the module name identify the module and must be unique.

  ```
  Example:  DSP
  ```

  This three-letter, unique module identifier is the label of the TITLE card.

  Each entry point or external reference must be prefixed by the six letter unique identifier of the module.

  ```
  Example:  DMKDSPCH
  ```

- TITLE CARD:

  ```
  DSP TITLE 'DMKDSP (CP) * VIRTUAL MACHINE PRODUCT * 5664-167'
  ```

- ERROR MESSAGES:

  There should be no insertions into the message at execution time and the length of the message should be resolved by the assembler. If insertions must be made, the message must be assembled as several DC statements, and the insert positions must be individually labeled.

- For all Rx instructions use a comma (,) to specify the base register when indexing is not being used, that is:

  ```
  L  R2,AB(,R4)
  ```

- To determine whether you are executing in a virtual machine or in a real machine, issue the Store Processor ID (STIDP) instruction. If STIDP is issued from a virtual machine, the version code, which precedes the CPUID field, will be X'FF'.

The CP loadlist EXEC contains a list of CP modules used by the VMFLOAD procedures when punching the text decks that will make up the CP system. All modules following DMKCPE in the list are pageable CP modules. Each 4K page in this area may contain one or more modules. The module grouping is governed by the order in which they appear in the loadlist. An SPB[1] (Set Page Boundary) card, a loader control card placed in the text file, forces the loader to start this module at the next higher 4K boundary. The loader automatically moves a module to the next higher 4K boundary if it cannot fit in with its predecessors on the load list. In this case a message is placed on the load map:

```
"SPB INSERTED"
```

as part of the line

```
 **EXTERNAL SYMBOL DICTIONARY FOR DMKXXX
```

An SPB card is required only for the first module following DMKCPE. If more than one module is to be contained in a 4K page, only the first can be assembled with an SPB card. The second and subsequent modules for a multiple module 4K page must not contain SPB cards.

The positions of several modules in the loadlist are critical:

- We recommend that DMKPSA be the first module in the CP resident nucleus.
- All modules following DMKCPE are pageable. They must be reenterable and must not contain any address constants referring to anything in the pageable CP area.
- The following modules must be the last six modules in the loadlist and they must appear in this order:

  DMKCKP
  DMKCKD
  DMKCKF
  DMKCKH
  DMKCKM
  DMKCKN

- No change should be made to the sequence of modules in the resident or pageable portion of the loadlist.

---

1 A 12-2-9 multipunch must be in column 1 of an SPB card and the characters SPB in columns 2, 3, and 4 respectively.

Chapter 11. Security Measures

Virtual Machine/System Product has facilities for detecting and foiling attempts to break system security:

- Journaling lets you monitor, record, and act on possible attempts to gain unauthorized access to system resources.
- It is possible to suppress the display of passwords entered as part of the LOGON and LINK commands.
- Automatic deactivation of passwords keeps restricted passwords from being assembled in the object directory.
- The access verification routines let you install control routines that further impede unauthorized use of certain commands.

Journaling the Logon, Autolog, and Link Commands.

LOGON, AUTOLOG, and LINK journaling detects and records certain occurrences of the LOGON, AUTOLOG, or LINK commands. Using the recorded information, an installation may be able to identify attempts to logon to VM/SP by users who issue invalid passwords. Also, the installation may be able to identify any user who successfully issues the LINK command to a protected minidisk that he does not own.

Briefly, LOGON, AUTOLOG, and LINK journaling works like this. While journaling is turned on, CP monitors all occurrences of the LOGON, AUTOLOG, and LINK commands. CP counts of the number of times a user issues one of these commands with an invalid password. CP can be set to take one or more of these actions when the count reaches a threshold value:

- Write a record to the accounting data set to record the incident
- Reject subsequent LOGON, AUTOLOG, or LINK commands issued by the user
- Lock that terminal for a designated period
- Send a message to a designated userid to alert the installation to the incident.

While journaling is turned on, CP creates an accounting record each time it detects that a user has successfully issued a LINK command to a protected

minidisk not owned by that user. A protected minidisk is a minidisk whose password is anything but ALL for the type of LINK attempted.

For a description of the accounting records that CP writes for LOGON, AUTOLOG, and LINK journaling, see the "Accounting Records for LOGON, AUTOLOG, and LINK Journaling" on page 49.

The SYSJRL macro instruction, the SET command, and the QUERY command enable an installation to control LOGON, AUTOLOG, and LINK journaling. To make journaling available and to specify options, code the SYSJRL macro instruction in module DMKSYS. Instructions for coding this macro instruction are in the *VM/SP Planning Guide and Reference.* To turn journaling on or off, use the class A SET command. To determine whether journaling is on or off, use the class A QUERY command.

Suppressing Passwords Entered on the Command Line

CP can be set to reject LOGON or LINK commands that have the password entered on the same line as the command. Rejecting these commands prevents passwords from being displayed or from being printed without masking (masking a password means overprinting the password so it cannot be read).

This capability is also available to virtual machines that issue LINK commands via DIAGNOSE code X'08'. For a description of DIAGNOSE code X'08', see *VM System Facilities for Programming.*

To request password suppression, specify it as an option on the SYSJRL macro instruction in module DMKSYS during system generation. Once requested, password suppression is always on; an operator cannot turn it off. Refer to the *VM/SP Planning Guide and Reference* for information on how to use and code SYSJRL in DMKSYS.

Auto-Deactivation of Restricted Passwords

Some installations have risked breaches of system security by using published passwords or passwords that are included in sample directories shipped with the system.

Auto-deactivation of restricted passwords keeps commonly known passwords out of the object directory. A file called RPWLIST DATA resides on the CMS system disk and contains a list of IBM's restricted passwords. When you execute the directory program to convert the source directory to an object directory, passwords are checked against the passwords in the RPWLIST DATA file. Passwords found in the restricted list are replaced with the password NOLOG in the object directory (thereby NOLOGing the userid) and the virtual machine issuing the DIRECT command receives a warning message.

Before you install your system, you should assign a new password to any directory entry with a restricted password listed in the RPWLIST DATA file.

Warning: This support can NOLOG the system administrator if that password is in the RPWLIST DATA file. The userid is NOLOGed when you run the directory program. Either change the system administrator's password to a non-restricted one or remove it from the RPWLIST DATA file. If you do not do this, the system administrator will not be able to logon after the directory program is run.

The auto-deactivation of restricted passwords is a part of the directory build process and is invoked each time the source directory is converted to an object directory. You do not need to do anything to invoke it. You can turn off auto-deactivation of restricted passwords by erasing the RPWLIST DATA file. If you do this, however, a warning message is issued each time the directory program is run.

Access Verification Routines

The access verification routines of VM/SP, when used with the Resource Access Control Facility (RACF)/VM 1.7.1, can be used to increase security: for example, by tightening control over minidisk access, logon passwords, and the movement of spool files. The access verification routines do not themselves provide security for your installation; they allow you to install software that does.

For a description of RACF/VM 1.7.1 (Program Number 5740-XXH), refer to the following publications:

- *Resource Access Control Facility (RACF) General Information Manual*
- *Resource Access Control Facility (RACF) Command Language Reference*
- *System Programming Library: Resource Access Control Facility (RACF)*
- *Resource Access Control Facility (RACF) Security Administrator's Guide*
- *Resource Access Control Facility (RACF) Auditor's Guide*
- *Resource Access Control Facility (RACF) Messages and Codes*

By themselves, access verification routines do not change any of VM/SP's functions. All commands work as they always did, but inside CP, these routines change the flow of control for several CP commands, so that RACF/VM 1.7.1 is called if it is installed.

You may want to use one feature furnished by the access verification routines. To identify userids as belonging to certain groups, you can define a new directory control statement, ACIGROUP. (See *VM/SP Planning Guide and Reference.*) You can also issue a DIAGNOSE code X'A0' to find which group a userid belongs to. (See *VM System Facilities for Programming.*)

Note: These groups are not related to the groups as defined for the Group Control System (GCS) component.

Several commands can invoke an access verification routine:

- LOGON/AUTOLOG
- LOGOFF (FORCE)
- LINK
- TRANSFER
- SPOOL
- TAG
- STCP

When a user enters one of these commands, the associated command processor passes information to the appropriate access verification routine. If you assign a group name to a user in the directory, CP passes the group name to the verification routine with the userid.

Part 2: Performance

Part 2 contains information about factors that affect the performance of virtual machines:

- How CP manages the processor's resources
- Performance guidelines, including VM/SP's interaction with:
  - Virtual Machine Assist Feature
  - VM/370 Extended Control-Program Support
  - VM/VS handshaking;
- Performance observation and analysis.

Chapter 12. Time and Storage Management

CP allocates processor resources to virtual machines according to their operating characteristics, their priorities, and the system resources available. This chapter describes how CP handles virtual machine time management and virtual machine storage management. You need to understand CP's decision processes in order to use the commands that alter CP's algorithms for resource allocation.

Virtual Machine Time Management

CP allocates the real processor's time among the virtual machines and various system tasks. CP tries to give each user his fair share of time. It also tries to keep users who are running many short jobs from getting stuck behind a machine that is waiting for some external process to be completed or that simply has a long job to do. Virtual machines that are executing many short jobs are called **interactive**. Short jobs mean frequent terminal interrupts, as happens when you are editing. Interactive virtual machines are given access to the real processor more frequently than virtual machines that are doing longer jobs (for example, doing a compilation), but such a **noninteractive** machine's turn is longer than an interactive machine's turn. Over a long period of time, a consistently interactive machine should get about the same amount of processor time as a consistently noninteractive machine.

At the end of each turn, CP determines the execution characteristics of a virtual machine and puts it in an **eligible list** to wait for processor time.

CP's scheduler and dispatcher use several lists to determine which CP task or virtual machine is to be given processor time, as shown in Figure 5 on page 71:

- The **run list** contains virtual machines that are currently being given access to the processor. That is, they are taking their turns. The run list has sometimes been called the *dispatch list.*

- The **eligible lists** contain machines waiting to move to the run list. There are separate eligible lists for interactive and noninteractive virtual machines. A machine waits in an eligible list until there is enough real storage to meet its requirements and the requirements of the machines already in the run list. The eligible lists have sometimes been called *queue 1* and *queue 2/queue 3.*

- The **dispatch request queues** contain pointers to CP tasks that are awaiting execution. The tasks are dispatched before virtual machines in the run list.

Queue Levels

The scheduler implements the distinction between interactive and noninteractive users by classifying a virtual machine as **queue 1 (Q1)**, **queue 2 (Q2)**, or **queue 3 (Q3)**. These **queue levels** determine a virtual machine's **queue slice** and how long it must wait in the eligible list.

Note: Although users are sometimes spoken of as being "in queue 1," etc., the "queues" do not refer to real lists, like the run list and the eligible lists. However, for historical reasons, a machine in the run list is said to be **in queue** *(not "in a queue"), and being dropped from the run list is called* **queue drop** *or being* **dropped from queue**.

The **queue slice** is the maximum length of time that a machine can stay in the run list.

Q1 machines are interactive. They get into the run list eight times as often as Q2 machines, but a Q1 queue slice is only one-eighth as long as a Q2 queue slice. In fact, Q1 users do not use their entire queue slices, since a machine is put into Q1 if it finished its task and went into a wait for terminal input during its last time in the run list. Machines are also put into Q1 at logon.

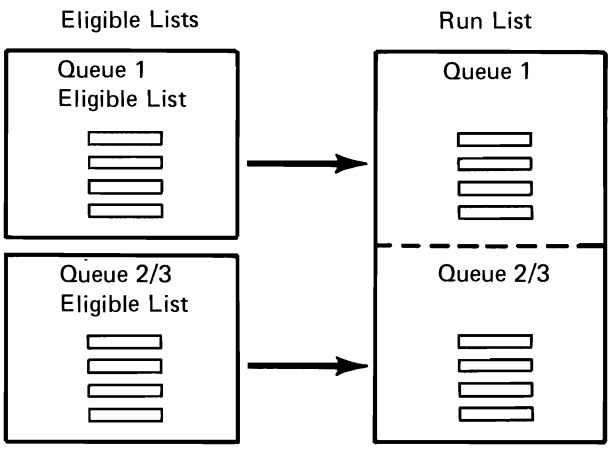
Q2 machines are noninteractive. A Q2 machine completed its last queue slice without finishing its job, that is, without going into a terminal I/O wait.

Q3 machines have not done any terminal I/O for a long time. They are executing very long jobs rather than switching back and forth between Q2 and Q1. A Q2 machine is classified as Q3 after it has used six Q2 queue slices without becoming a Q1 machine. Q3 machines have to wait eight times as long as Q2 machines to get into the run list. The Q3 queue slice is the same as the Q2 queue slice, but a Q3 machine gets to stay in the run list for eight Q2 queue slices before it is dropped.[2]

Over an extended period, a Q3 virtual machine should receive as much processor time as it would as a Q2 machine. However, its performance may be better because it is using less overhead doing queue drops. For some programs using a lot of virtual storage, Q3 operation cuts in half the total

---

[2] The CMS BLIP facility causes CMS to perform a write operation to the terminal after every 2 seconds of virtual processor use. This feature effectively cancels Queue 3 use for normal, connected CMS virtual machines, regardless of what types of programs they are running. The CMS BLIP facility can be turned off with the CMS SET BLIP OFF command or it can be disabled with the CP SET TIMER OFF command.



1. Virtual machines in the eligible lists are sorted in deadline priority order.
2. Scheduler searches Q1 eligible list from top down.
3. If the Q1 eligible list is empty, the Q2 eligible list is searched.
4. If the highest priority virtual machine can fit in the run list, it is moved from the eligible list to the run list, which is also ordered according to deadline priority.

Figure 5. Relationships of Scheduler Lists and Queue Levels

use of processor resources, compared with Q2 operation. Q2 machines have a similar gain in performance over Q1 machines.

Selecting a Virtual Machine to Run

Virtual machines in the run list take turns using the processor. When a machine completes its task or finishes its queue slice, it is dropped from the run list and inserted into an eligible list according to its deadline priority. A virtual machine is then selected from the top of an eligible list and inserted into the run list according to its deadline priority.

The Eligible Lists

This section describes how a virtual machine is sorted into an eligible list and eventually moves into the run list.

In order to put a virtual machine in the right place in the eligible list, the scheduler calculates its execution characteristics at queue drop time. Queue drop occurs in three circumstances: when the virtual machine first logs on; when the virtual machine uses up its queue slice; and when the virtual machine recovers from a long wait, during which it could have been waiting for terminal input, in CP console mode, for a timer to expire, or because it had loaded a wait PSW.

Three calculations are made at queue drop:

- CP assigns a queue level, as described above. Machines that are recovering from a long wait or that have just logged on are assigned to Q1.

- CP computes the **projected working set size**. Virtual machines are expected to do a certain amount of paging during a queue slice, but the working sets are adjusted to keep paging activity within an acceptable range. CP first calculates the average amount of main storage the virtual machine used during its queue slice. It compares the average with an ideal level of storage use, which is based on how much of the machine's queue slice was spent swapping pages between virtual and main storage. If the virtual machine is not doing too much paging, the projected working set is the average. Otherwise, the projected working set size is adjusted toward the ideal.

  The projected working set size is important in several ways. CP uses working set size as a measure of resource use when it determines where to place the virtual machine in the eligible list (see next paragraph). A virtual machine at the top of an eligible list cannot enter the run list if its working set is too large to fit in main storage with the other in-queue machines'. Finally, the projected working set size is part of the feedback mechanism by which CP adjusts the system to the current load.

- CP calculates the machine's **deadline priority**.

  Once a machine has its deadline priority, it can be sorted into the appropriate eligible list. The deadline priority is a list-position number that indicates the delay until the user can be expected to finish its next queue slice. It is based upon how well the user has been doing compared with the average of all the users and is adjusted to bring the user toward the average. The idea is to give every user a fair share of the system's resources by trying to make everyone an "average user."

  The deadline priority is explained in more detail below.

The queue level, projected working set size, and deadline priority interact with one another in affecting a machine's performance. For instance, it was noted above that a machine's performance improves when it enters Q3, because overhead from doing queue drops is reduced. Some of the gain in performance can come from reduced paging activity. At first, as a Q1 and Q2 machine, it would have spent much of each queue slice swapping pages (page fault) and getting little work done. The high number of page faults would cause its working set and deadline priority to be increased at each queue drop, until it was classified Q3. Then, when it got into the run list, it would probably have the time and space to work effectively.

The Run List

A virtual machine in the run list gets to execute, but it still has to share storage and processor time with the other in-queue machines.

Entry into the run list is controlled by deadline priority and the availability of real storage. As users drop from the run list, the top machine in the Q1 eligible list becomes the candidate to be put into the run list. If the Q1 eligible list is empty, the top virtual machine in the Q2 eligible list becomes the candidate. Thus, a Q1 machine generally gets into the run list sooner than a Q2 machine. If the candidate's working set is too large to fit into main storage, it and the machines lower down in the eligible list must wait until enough machines have dropped from queue to free the storage needed by the candidate.

CP's promotion logic contains provisions for special cases. The two most important are :

- Within limits, CP will squeeze Q1 machines into the run list.
- CP makes sure that no user waits too long in the Q2 eligible list: a Q2 machine that has passed its deadline is given priority over any Q1 machines. If it is necessary, CP will search the run list to see if any in-queue virtual machines can be dropped ("preempted") to make room for this large, behind schedule Q2 machine.

The in-queue machines also share the processor in a round-robin fashion. A virtual machine is not allowed to run continuously during a queue slice. It is allowed to run only for short periods of time called **time slices**.[3] After it executes for a time slice, it must yield to another **runnable** machine. A machine in the run list becomes **nonrunnable** if it is waiting for a page of storage, for an input/output operation—except terminal I/O—to be translated and started, for a CP command to finish execution, or for some other activity or resource.

A user stays in the run list, getting access to the processor for a time slice at a time, until he completes the current job (and goes into terminal-wait state, waiting for a new command) or when he uses up his queue slice. In either case, he is dropped from the run list into an eligible list. CP selects the next CP task or virtual machine to run from the dispatch request queue or the run list, respectively.

---

[3] The **time slice** is the basic unit of scheduling calculations. Its length varies with the speed of the processor. It is set at initialization but can be modified by the SET SRM DSPSLICE command. The different queue slices are:

Q1 8 time slices

Q2 64 time slices

Q3 64 time slices. However, the Q3 user remains in the run list until he has used 8 queue slices.

Deadline Priority

CP calculates the deadline priority at queue drop time by the following formula:

```
deadline priority = TOD + Virtual machine queue delay factor
```

where:

TOD
is the current time of day.

Virtual machine queue delay factor
is the user bias ratio * prioritized Q2 delay factor.

User bias ratio
depends on the amount of specified resources the particular virtual machine is currently receiving. It is the weighted average of the paging and processor resource ratios.

Q2 delay factor
is based on configuration and load. It is the average elapsed time required by a virtual machine to receive an amount of processor time equal to one Q2 queue slice.

For Q1 virtual machines, the user bias ratio is divided by 8, since the Q1 queue slice is one-eighth the Q2 queue slice.

A virtual machine can be assigned a priority of execution. Priority, a parameter in the virtual machine's directory entry, determines how soon a particular virtual machine is allowed to run again, compared with other virtual machines that have the same general execution characteristics. The system operator can reset the priority parameter with the class A SET PRIORITY command.

Virtual Machine Storage Management

The normal and maximum storage sizes of a virtual machine are defined as part of the virtual machine configuration in the VM/SP directory. You may redefine virtual storage size to any value that is a multiple of 4K and not greater than the maximum value defined in the directory. VM/SP implements this storage as virtual storage. The storage may appear as paged or unpaged to the virtual machine, depending upon whether or not the extended control mode option was specified for that virtual machine. This option is required if operating systems that control virtual storage, such as OS/VS1, VM/370 or VM/SP are run in the virtual machine.

Virtual machine storage is logically divided into 4096-byte areas called *pages*. *Segments* are contiguous 64K areas of virtual storage. Segment and page tables describe the storage of each virtual machine. A page table shows whether a page is in real storage and matches virtual addresses to real storage addresses. A segment table is used with dynamic address

translation to control user access to virtual storage segments. Each entry shows the length, location, and availability of a corresponding page table. CP updates these tables to reflect the allocation of virtual storage pages to blocks of real storage. These page and segment tables are used for virtual storage addressing in a System/370 machine.

To make the best use of real storage, CP keeps only active virtual storage pages in real storage. Further, CP can bring a page into any available page frame. During program execution, a combination of VM/SP and the dynamic address translation feature on the System/370 relocates the page. The active pages from all logged on virtual machines and from the pageable routines of CP compete for available page frames. When the number of page frames available for allocation falls below a threshold value, CP determines which virtual storage pages currently allocated to real storage are relatively inactive and starts suitable page-out operations for them.

CP keeps track of where each virtual machine's page zero resides. Normally, CP does this by issuing a TRANS macro that checks for page residency (LRA) and demands a page-in if the page is not in real storage. However, if an in-storage pointer in the VMBLOK contains the address of the virtual machine's page zero, the page is resident and CP bypasses issuing the TRANS macro. Thus, unnecessary LCTL and LRA instructions are eliminated.

Inactive pages are kept on a direct access storage device (DASD). If an inactive page has been changed during virtual machine execution, CP assigns it to a paging device, selecting the fastest such device with available space. If the page has not changed, it remains in its original direct access location and is paged into real storage when the virtual machine next refers to that page. A virtual machine program can use the DIAGNOSE instruction to tell CP that the information from specific pages of virtual storage is no longer needed. CP then releases the paging devices areas which were assigned to hold the specified pages.

Paging is done on demand by CP. This means that a page of virtual storage is not read (paged) from the paging device to a real storage block until it is actually needed for virtual machine execution. CP makes no attempt to anticipate what pages might be required by a virtual machine. During paging for one virtual machine, another virtual machine can be executing. Any paging operation started by CP is transparent to the virtual machine.

If the virtual machine is executing in extended control mode with translate on, two additional sets of segment and page tables are kept. The virtual machine operating system must map the virtual storage created by it to the storage of the virtual machine. CP uses these tables and the page and segment tables created for the virtual machine at logon to build *shadow page tables* for the virtual machine. These shadow tables map the virtual storage created by the virtual machine operating system to the storage of the real computing system. The tables created by the virtual machine operating system may describe any page and segment size permissible in the IBM System/370.

Chapter 13. Performance Guidelines

The performance characteristics of an operating system running in a virtual machine environment depend upon:

- The System/370 model used
- The characteristics of the operating system and its work level
- The total number of virtual machines executing
- The type of work being done by each virtual machine
- The speed, capacity, and number of the paging devices
- The order in which devices are selected for preferred paging and spooling
- The amount of real storage available
- The degree of channel and control unit contention, as well as arm contention, affecting the paging device
- The type and number of VM/SP performance options in use by one or more virtual machines
- The degree of MSS 3330 volume use
- The amount of fixed head paging storage (drum, 3340, 3344, 3350, 3380)

The performance of any virtual machine can be improved by the choice of hardware, operating system, and VM/SP options. This section describes:

- The performance options available in VM/SP to improve the performance of a particular virtual machine.
- The system options and operational characteristics of operating systems running in virtual machines that affect their execution in the virtual machine environment.

The performance of a specific virtual machine may never equal that of the same operating system running alone on the same System/370, but the total throughput obtained in the virtual machine environment may equal or better that obtained on a real machine.

When a function executing in a virtual machine cannot be performed completely by the hardware, the virtual machine's performance is degraded to some degree. As the control program for the real machine, CP initially processes all real interrupts. The instructions of a virtual machine's operating system always execute in *problem state*. Any privileged instruction issued by the virtual machine causes a real privileged instruction exception interruption. The amount of work to be done by CP to analyze and handle a virtual machine-initiated interrupt depends on the type and complexity of the interrupt.

The simulation effort required of CP may be trivial, as for a supervisor call (SVC) interrupt (which is generally reflected back to the virtual machine), or may be more complex, like a Start I/O (SIO) privileged instruction, which starts extensive CP processing.

When you plan the virtual machine environment, consider the number and type of privileged instructions to be executed by the virtual machines. Reducing the number of privileged instructions issued by the virtual machine's operating system reduces the amount of extra work CP must do to support the machine.

Before deciding which performance options to apply to your system, you should monitor the current performance of your system to decide which options would most likely give the system a performance gain and where performance bottlenecks are occurring. Refer to Chapter 14, "Performance Observation and Analysis" on page 107 for guidelines and functions you can use to observe the present system performance.

Reducing the Number of Virtual Machine SIOs Handled by CP

Handling of SIOs for virtual machines can be one of the most significant causes of reduced performance in virtual machines. To support I/O processing in a virtual machine, CP must translate all virtual machine channel command word (CCW) sequences to refer to real storage and real devices and, for minidisks, real cylinders. When a virtual machine issues an SIO, CP must:

1. Intercept the virtual machine SIO
2. Allocate real storage space to hold the real CCW list to be created
3. Translate the virtual data addresses to real data addresses
4. Translate the virtual device addresses referred to in the virtual CCWs to real device addresses
5. Page into real storage and lock, for the duration of the I/O operation, all virtual storage pages required to support the I/O operation
6. Generate a new CCW sequence building a Channel Indirect Data Address list if the real storage locations cross page boundaries

7. If the real device is a 3330V, append an MSS cylinder fault prefix to the CCW prefix to prevent the channel from doing channel command retry

8. Schedule the I/O request

9. Present the SIO condition code to the virtual machine

10. Recognize an MSS cylinder fault, queue the I/O request, and reschedule the request when the subsequent interruption is received (indicating staging is complete)

11. Intercept, retranslate, and present the channel end and device end interrupts to the appropriate virtual machine, where they must then be processed by the virtual machine operating system.

The number of SIO operations required by a virtual machine can be significantly reduced by:

- Using large blocking factors (up to 4096 bytes) for user data sets to reduce the total number of SIOs needed

- Using preallocated data sets

- Using virtual machine operating system options (such as chained scheduling in OS) that reduce the number of SIO instructions

- Substituting a faster resource (virtual storage) for I/O operations, by building small temporary data sets in virtual storage rather than using an I/O device.

Frequently, performance is improved when CP paging is substituted for virtual machine I/O operations. To improve the performance of an operating system, such as OS, specify frequently-used OS functions (transient subroutines, ISAM indexes, and so forth) as resident in second level storage if possible. In this way, paging I/O is substituted for virtual machine-initiated I/O. Thus, CP only needs to place the page that contains the desired routine or data into real storage.

You can use the following CP performance options to reduce the CP overhead associated with privileged instructions used by the virtual machine's I/O Supervisor (for example, virtual machine I/O instructions).

- The virtual=real (V=R) option in VM/SP removes the need for CP to translate storage references and to do paging before each I/O operation for a specific virtual machine.

- The virtual machine assist feature reduces the real supervisor state time used by VM/SP. For a detailed description of the feature, see "Virtual Machine Assist Feature" later in this section. For a list of processors on which the feature is available, see the *VM/SP Planning Guide and Reference*.

- VM/370 Extended Control-Program Support (ECPS) further reduces the real supervisor state time used by VM/SP. For a detailed description of ECPS, see "Extended Control-Program Support: VM/370" on page 98. For a list of processors on which ECPS is available, see the *VM/SP Planning Guide and Reference.*

Assignment and use of these options are discussed in "VM/SP Performance Options" on page 87.

Reducing Paging Activity

When virtual machines refer to virtual storage addresses not currently in real storage, they cause a paging exception and associated CP paging activity.

The addressing characteristics of programs executing in virtual storage significantly affect the number of page exceptions for that virtual machine. Routines with widely scattered storage references tend to increase the paging load of a particular virtual machine. When possible, place modules of code that depend on each other in the same page. Place reference tables, constants, and literals near the routines that use them. Do not place infrequently used exception or error routines in main routines, put them elsewhere.

When an available page of virtual storage contains only reenterable code, paging activity can be reduced, since the page, although referred to, is never changed, and thus does not cause a write operation to the paging device. The first copy of that page is written on the paging device when that frame is needed for some other more active page. Only inactive pages that have changed must be paged out.

Virtual machines that reduce their paging activity by controlling their use of addressable space improve resource management for that virtual machine, the VM/SP system, and all other virtual machines. The total paging load handled by CP is reduced, and more time is available for productive virtual machine use.

Using the SYSCOR Macro to Control Free Storage Allocation

The more dynamic paging storage available, the less paging activity occurs. To gain additional dynamic paging storage, control the amount of free storage allocated at VM/SP initialization time. When you generate the system, use the FREE operand of the SYSCOR macro statement to specify the number of free storage pages to be allocated at system load time.

At IPL time, if the amount of storage that these pages represent is greater than 25 percent of the VM/SP storage size (not including the V=R area, if any), a default number of pages is used. The default value is 3 pages for the first 256K bytes of storage plus 1 page for each additional 64K bytes (not including the V=R size, if any).

The SYSCOR macro definition can be found in *VM/SP Planning Guide and Reference*.

Paging Performance Options

To reduce the paging requirements of virtual machines, CP provides locked pages, reserved page frames, and a V=R area. Generally, these facilities require some dedication of real storage to the chosen virtual machine and, therefore, improve its performance at the expense of other virtual machines.

Locking Pages into Real Storage

To fix or lock specific pages of virtual storage permanently into real storage, use the LOCK command (privilege class A). In so doing, all paging I/O for these page frames is eliminated.

Since the LOCK command reduces total real storage resources (real page frames) available for other virtual machines, lock only frequently used pages into real storage. Since page zero (the first 4096 bytes) of a virtual machine storage is referred to and changed frequently (for example, whenever a virtual machine interrupt occurs or when a CSW is stored), consider locking page zero of a particular virtual machine first. Also consider locking virtual machine interrupt handler pages.

Other pages to be locked depend upon the work being done by the particular virtual machine and its use of virtual storage.

The normal CP paging mechanism selects inactive page frames in real storage for replacement by active pages. Page frames belonging to inactive virtual machines are normally all selected eventually and paged out if the real storage frames are needed to support active virtual machine pages.

When virtual machine activity is started on an infrequent or irregular basis, such as from a remote terminal in a teleprocessing inquiry system, some or all of its virtual storage may have been paged out before the time the virtual machine begins processing. Some pages then have to be paged in so that the virtual machine can respond to the teleprocessing request. This paging activity might increase the time to respond to the request compared with running the same teleprocessing program on a real machine. Further response time is variable, depending upon the number of paging operations that must occur. Locking specific pages of the virtual machine's program into real storage can ease this problem, but you may not be able to identify which specific pages are required.

Once a page is locked, it remains locked until either the user logs off or the system operator (privilege class A) issues the UNLOCK command for that page. If the "locked pages" option is in effect and the user loads his system again (via IPL) or loads another system, the virtual machine's locked pages are unlocked by the system. When a user issues the SYSTEM CLEAR

command, virtual machine storage is cleared, and the user's locked pages are unlocked.

Note: In a system generated for attached processor or multiprocessor operation, no shared pages are locked. If the system operator tries to lock a shared page or an address range containing one or more shared pages, he receives the message:

```
DMKCPV165I Page hexloc not locked; shared page
```

for each of the shared pages within the range.

Reserving Page Frames

The reserved page frames option is a more flexible approach than locked pages. To provide a specified virtual machine with an essentially private set of real page frames, use the CP SET RESERVE command. If the program code or data required to satisfy the request was in real storage at the time the virtual machine became inactive, paging is not required for the virtual machine to respond.

This option is usually more efficient than locked pages since the pages with the most references at that moment remain in real storage, as determined automatically by the system. Although multiple virtual machines may use the LOCK option, only one virtual machine at a time may have the reserved page frames option active. Assignment of this option is discussed further in "VM/SP Performance Options" on page 87.

The reserved page frames option provides performance that is generally consistent from run to run with regard to paging activity. This can be especially valuable for production-oriented virtual machines with critical schedules, or those running teleprocessing applications where response times must be kept as short as possible. The SET RESERVE command can be used to increase the efficiency of certain noninteractive virtual machines such as system control programs and special service machines. You can use the SET RESERVE command to reserve page frames for multiple virtual machines.

To specify the maximum number of reserved page frames, use the class A command:

```
SET RESERVE userid nnnn
```

where *nnnn* is the maximum number required (1-4096). The number of frames held is *nnnn* or the working set size whichever is smaller. You can specify SET RESERVE for multiple virtual machines at any one time.

Note: *nnnn* should never approach the total available pages, since CP overhead is substantially increased in this situation, and excessive paging activity is likely to occur in other virtual machines.

Eliminating CP Paging for a Selected Virtual Machine

To eliminate CP paging for the selected virtual machine, use the VM/SP V=R directory option. All pages of virtual machine storage, except page zero, are locked in the real storage locations they would use on a real computer. CP controls real page zero, but the remainder of the CP nucleus is relocated and placed beyond the V=R machine in real storage.

Since the entire address space required by the virtual machine is locked, these page frames are not available for use by other virtual machines except when the V=R area is unlocked. This option often increases the paging activity for other virtual machine users, and sometimes for VM/SP. (Paging activity on the system may increase substantially, since all other virtual machine storage requirements must be managed with fewer remaining real page frames.)

The V=R option may be desirable or mandatory in certain situations. The V=R option is desirable when running a virtual machine operating system (like DOS/VS or OS/VS) that does paging of its own because the possibility of double paging is eliminated. You must use the option to allow programs that execute self-modifying channel programs or have a certain degree of hardware timing dependencies to run under VM/SP.

For this option, the VM/SP nucleus is reorganized to provide an area in real storage large enough to contain the entire V=R machine. In the selected virtual machine, each page from page 1 to the end is in its true real storage location; only its page zero is relocated. The virtual machine is still run in dynamic address translation mode, but since the virtual page address is the same as the real page address, no CCW translation is required.

For information about generating a V=R system, see the *VM/SP Installation Guide*.

Figure 6 shows an example of a real storage layout with the V=R option. The V=R area is 128K and real storage is 512K.

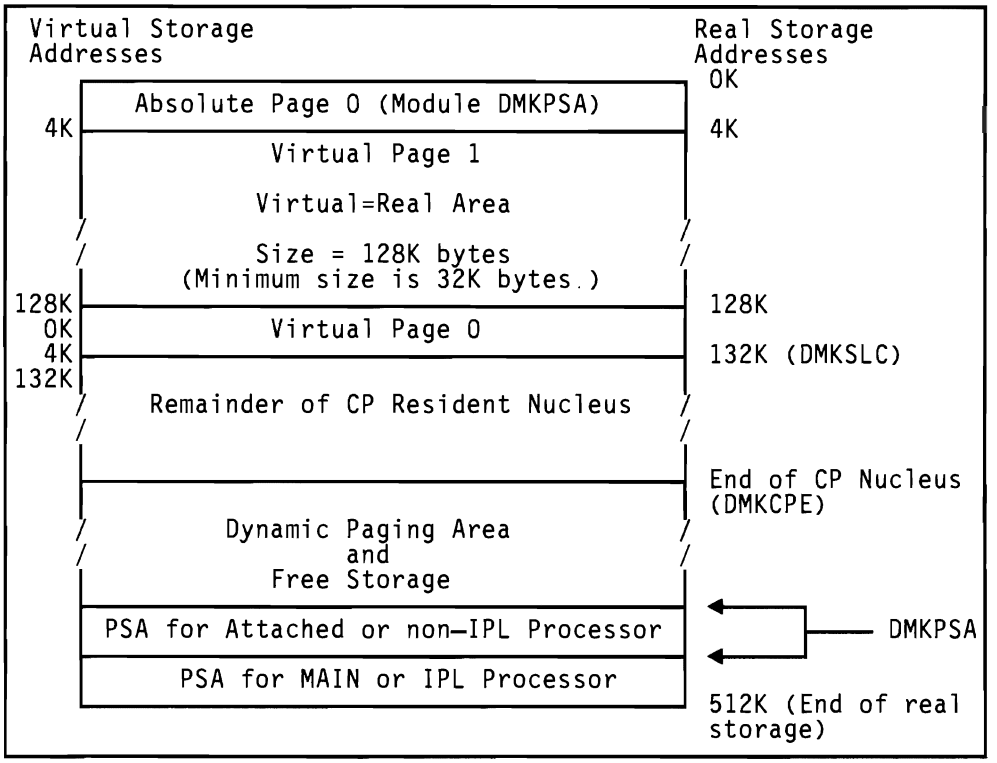


Figure 6. Storage Layout in a Virtual=Real Machine

Consider the following when planning to use the V=R option because of the effect on overall system operation:

1. The area of contiguous storage built for the V=R machine must be large enough to contain the entire addressing space of the largest V=R machine. During system generation when the V=R option is selected, define the V=R storage size for the VM/SP system.

2. Only a virtual machine with the V=R option specified in its directory entry can use the storage reserved for a V=R machine. This storage is not available to other users for paging space, nor for VM/SP use until released from V=R status by a system operator via the CP UNLOCK command. Once this storage is released, VM/SP must be loaded again before the V=R option can become active again.

3. The virtual machine with the V=R option operates in the preallocated storage area with normal CCW translation in effect until the CP SET NOTRANS ON command is issued. At that time, with several exceptions, all subsequent I/O operations are performed from the virtual CCWs in the V=R space without translation. The exceptions occur when:

   - SIO tracing is active
   - The first CCW is not in the V=R region
   - I/O operation is a sense command
   - I/O device is a dial-up terminal

- I/O is for a device that is not dedicated (spooled unit record console virtual CTCA or minidisks that are less than a full volume)
- I/O device has an alternate path
- Device status is pending.

Any one of the above conditions forces CCW translation. Since minidisks are nondedicated devices, they may be used by programs running in the V=R region even though CP SET NOTRANS ON is in effect.

4. If the V=R machine performs a virtual reset or IPL, the normal CCW translation goes into effect until the CP SET NOTRANS ON command is again issued. This permits simulation of an IPL sequence by CP. Only the V=R virtual machine can issue the command. A message is issued if normal translation mode is entered.

5. A V=R machine must not IPL a named or shared system. It must IPL by device address.

6. When NOTRANS is in effect for a V=R machine, no significant SEEK data is collected by MONITOR operations for the V=R machine.

7. If you define a V=R area on a 3081 processor, the reliability and availability of the V=R machine can be improved if the V=R machine issues the TEST BLOCK instruction to validate storage in the V=R area. Note that the only two SCPs that issue TEST BLOCK are MVS/SP and VM/SP. The hardware system area (HSA) on a 3081 processor can reside in the middle of the V=R area; these two control programs mark the HSA as invalid and continue validating storage. Any other system control program, such as OS/VS, validates storage with the MVCL instruction. When OS/VS encounters the beginning of the HSA, it assumes that it has reached the end of storage. Therefore, such a control program running in the V=R area of VM/SP on a 3081 processor may not have access to the full V=R area.

8. If your system runs in single processor mode on a 3081 processor, the system operator must issue a VARY OFF PROCESSOR nn VLOG command.

9. A V=R machine running in extended control mode on a 3081 processor can issue a MSSFCALL (DIAGNOSE X'80') for VARY PROCESSOR commands, MSSF SCPINFO commands, and Input/Output Configuration Program (IOCP) commands. MSSF processes these commands.

Managing Page Migration

To keep 12% of the preferred paging area available, CP moves inactive pages from preferred to nonpreferred paging areas. The preferred paging area includes a fixed-head area and a movable-head area. The fixed-head paging area is paging space on a drum and space under the fixed heads of a DASD volume that has the fixed head feature installed. The movable-head paging area is paging space on a DASD volume that is accessed by a movable arm. Normally, CP invokes page migration, based on calculated load levels, once every ten minutes.

Inactive pages in the fixed-head preferred paging area are moved every time CP invokes migration. For pages in the movable-head preferred paging area, you can decide at what point inactive pages are selected for migration. Use the SET SRM MHFULL command to set movable-head page migration limits.

If a percentage for MHFULL has been specified, CP moves pages from movable-head preferred paging areas only when that percentage is reached and ten minutes has elapsed, rather than whenever fixed-head areas are full. Thus, migration from movable-head preferred paging areas and fixed-head preferred paging areas can take place separately.

In addition, you can use the MIGRATE command to invoke page and swap table migration immediately. Page migration can also be invoked only for a specific virtual machine. The format of the MIGRATE command is described in the *VM/SP CP Command Reference.*

Displaying, Changing, or Setting System Resource Management Variables

To display internal system activity counters or parameters, use the QUERY SRM command. To set or change internal system activity counters or parameters, use the SET SRM command. Formats for the QUERY SRM and SET SRM commands are contained in the *VM/SP CP Command Reference.*

Use the class A or class E QUERY SRM command to display the following information:

- Current number of pageable pages
- Size of the queue slice
- Setting of the maximum working set estimate
- Maximum drum page allocation limit
- Current page migration counters
- Unused segment elapsed time as criteria for page migration
- Current PCI flag setting mode for 2305 page requests
- Maximum page bias value
- Current interactive shift bias value
- Moveable head page migration limit.

Use the class E SET SRM command to set some of the system variables that can affect the values displayed by the QUERY SRM command.

Displaying and Setting Paging Variables

The paging variable is used in the working set size algorithm. The current paging load is constantly compared with the paging variable. CP adjusts the working set size estimates based on how the actual load compares with the paging load variable.

Use the QUERY PAGING command to display the paging variable used in the working set size estimate control algorithm. To get information on the paging rate use the INDICATE LOAD command.

Use the SET PAGING command to change the paging variable used in the working set size estimate.

Information about the formats of the QUERY PAGING and SET PAGING commands is contained in the *VM/SP CP Command Reference.*

VM/SP Performance Options

VM/SP provides a number of options you may use to improve the performance of virtual machines and VM/SP. Several options improve the performance of installation specified virtual machines; other options improve the performance of all virtual machines and VM/SP. The options described in the following discussion are:

- Small CP option
- Favored execution
- Virtual machine priority
- Affinity
- Multiple shadow table support
- Shadow table bypass
- Single processor mode
- Dynamic SCP transition to or from native mode
- Queue drop elimination
- Virtual machine assist
- Extended Control-Program Support
- MVS Extensions support.

When you specify a performance option, you may be improving the performance of one virtual machine at the expense of VM/SP and other virtual machines. For example, after an operator specifies favored execution for a virtual machine, that virtual machine receives more processor time than other virtual machines. Therefore, before specifying any performance option, identify the option's performance trade-offs and assess their impact on system performance. (See Chapter 14, "Performance Observation and Analysis" on page 107.)

The Small CP Option

During CP system generation, the installation can specify the Small CP Option, which removes some of the resident CP nucleus functions. This reduces the size of the resident CP nucleus, making more storage available for the area where virtual machine pages reside. The Small CP option can improves performance on any system. However, it is better used when limited real processor storage size has already limited function, that is, real storage is less than 2 megabytes. See *VM/SP Planning Guide and Reference* for a full description of the Small CP option.

Giving More Time to a Virtual Machine

The favored execution option and virtual machine priority option change the normal scheduler algorithm. The virtual machine priority option tends to take precedence over the favored execution option even when you specify a percentage. For example, suppose a user with the required privilege class issues a SET FAVORED command for USERIDA. If USERIDA was assigned a lower priority than USERIDB, USERIDA may get a smaller percentage of processor time than was specified with the favored option.

Favored Execution Options

To change the normal CP deadline priority calculations in the fair share scheduler to force the system to devote more of its processor resources to a given virtual machine, use the favored execution options. The options are:

- The basic favored execution option
- The favored execution percentage option.

To specify that a virtual machine is to remain in the run list at all times, unless it becomes nonexecutable, use the *basic favored execution option.* When the virtual machine is executable, it is placed in the run list at its normal priority position. However, any active virtual machine represents either an explicit or implicit commitment of main storage. You can specify an explicit storage commitment by either the V=R option or the reserved page frames option. An implicit commitment exists if neither of these options is specified, and the scheduler recomputes the virtual machine's projected working set at what it would normally have been at queue drop time. You can set multiple virtual machines for the basic favored execution option. However, if their combined main storage requirements exceed the system's capacity, performance can suffer since the system can do little useful work because of excessive paging.

If the favored task is highly compute bound and must compete for the processor with many other tasks of the same type, you should define how much time the favored task should get. In this case, you can use the favored execution percentage option. This option specifies that the selected virtual machine is requesting a specified minimum percentage (from 1 to 100) of the total processor time. If the requested percentage is from 1 to 99, CP tries to place the virtual machine in the run list so that it gets that percentage of processor time, if it can use it. If you want the virtual

machine to stay in the run list, you must also invoke the basic favored option.

If a virtual machine requests 100 percent of the processor time, CP attempts to keep that virtual machine at the top of the run list. Since the dispatcher searches the run list top down when selecting a virtual machine to be dispatched, the 100% favored userid will usually be the first virtual machine examined (and thus be selected to run).

To select the favored execution option, specify the FAVORED operand on the class A SET command. The description of the SET command is in the *VM/SP CP Command Reference*. After the option is invoked, VM/SP provides processor time for the selected virtual machine as follows:

1. CP multiplies the queue slice by the specified percentage to arrive at the virtual machine's requested processor time.

2. The scheduler tries to place the virtual machine, when it is executable, at the top of the run list until it has obtained its requested processor time.

3. If the virtual machine obtains its requested processor time before the end of its queue slice, it is placed in the run list according to its calculated dispatching priority.

4. In either case (2 or 3), at the end of the queue slice the requested percentage is recomputed as in step 1 and the process is repeated.

If a percentage is not specified, a virtual machine with the favored execution option active is kept in the run list except under the following conditions:

- Entering CP console function mode
- Loading a disabled PSW
- Loading an enabled PSW with no active I/O in process
- Logging on or off.

When the virtual machine becomes executable again, it is put back on the run list as a Q1 virtual machine. If it is dropped from Q1, the virtual machine is placed directly in the run list as a Q2 virtual machine. If you specified the percentage option of the SET FAVORED command, the deadline priority is calculated at queue drop time by:

$$\text{Deadline} = \text{current time-of-day} + \frac{\text{length of queue slice}}{\text{favored percentage}}$$

For example, if the queue slice is 1 second, and the specified percentage is 10, then 10 seconds are added to the current time-of-day. The virtual machine should receive one queue slice (1 second) once every 10 seconds.

Note, however, that these options can affect response times of other virtual machines. To provide a virtual machine with both options, basic and

percentage, issue both forms of the command for that virtual machine. You can use the percentage form of the SET FAVORED command to specify any number of logged-on virtual machines.

Although the SET FAVORED command prevents specifying more than 100% for a particular virtual machine, nothing prevents you from allocating a total of more than 100% to several virtual machines. Where more than 100% has been allocated, the favored virtual machines compete for the available resources on a pro rata basis. An individual virtual machine's allocation is roughly proportional to the percentage allocated to it, divided by the total percentage allocated to all virtual machines. The effect of allocating more than 100% of the system on interactive (Q1) responses is unpredictable.

Note: The percentage of the processor time actually received by the favored user normally remains close to the percentage specified in the command. However, it is not an absolute value and varies depending on the total load and type of load on the system. If, for example, multiple virtual machines on the run list are compute bound (that is, are not queue dropped before the end of their queue slices), the favored user may not receive its requested percentage of the total processor time.

Setting Virtual Machine Priority

The VM/SP operator can assign specific priority values to different virtual machines. A virtual machine with higher priority is allocated a larger share of the system resources than a virtual machine with lower priority. To assign specific priority values to different virtual machines, use the following class A command:

SET PRIORITY *userid* {*nn*}
{64}

where

userid is the user's identification

nn is the priority value. It is an integer value from 0 to 99. The default is 64. The priority value figures in the deadline priority calculations and thereby affects the virtual machine's dispatching priority with respect to other users.

Selecting Attached Processor or Multiprocessor Affinity for a Program

To allow virtual machines that operate on attached processor or multiprocessor systems to select the processor of their choice for program execution, use the affinity option. To select the affinity option, use the directory OPTION statement, or specify the AFFINITY operand on the class A or G SET command. The directory OPTION statement is described in the *VM/SP Planning Guide and Reference.* The class A SET command

and the class G SET command are described in the *VM/SP CP Command Reference*.

The affinity setting of a virtual machine implies a preference of operation to either (or neither) processor. Affinity of operation for a virtual machine means the program of that virtual machine is executed on the selected or named processor. It does not imply that supervisory functions and CP housekeeping functions associated with that virtual machine are handled by the same processor.

In attached processor systems, all real I/O operations and associated interrupts are handled by the main processor. Virtual I/O started on the attached processor that is mapped to real devices must transfer control to the main processor for real I/O execution. Therefore, your system can benefit in a virtual machine "mix" if you relegate those virtual machines that have a high I/O-to-compute ratio to the main processor, and those virtual machines that have a high compute-to-I/O ratio to the attached processor. Weigh such decisions carefully as every virtual machine is contending with other virtual machines for system resources.

To improve a virtual machine's performance on a multiprocessor where the path(s) to a user's primary minidisks are from one processor only, set the user's affinity to that processor.

More importantly, use of the affinity setting in applications where a virtual machine program requires special hardware features available on one processor and not the other. Such features could be a performance enhancement such as virtual machine assist (described later) or a special RPQ required for a particular program's execution.

Virtual Relocation and Shadow Table

CP allows the virtual machine to use the dynamic address translation (DAT) feature of the real System/370. Programming simulation and hardware features are combined to allow the virtual machine to use the available features in the real hardware (2K or 4K pages, 64K segments).

First-level storage
The physical storage of the real processor, in which CP resides.

Second-level storage
The virtual storage available to any virtual machine. This storage is maintained by CP.

Third-level storage
The virtual storage space defined by the system operating in second-level.

Page and segment tables
Logical mapping between first-level and second-level storage.

Virtual page and segment tables
Logical mapping between second-level and third-level storage.

Shadow page and segment tables
Logical mapping between first-level storage and third-level storage.

Figure 7. Definitions of storage levels and segment tables.

A standard, nonrelocating virtual machine uses control register 0 for:

- Extended masking of external interruptions
- Special interruption traps for SSM
- Enabling virtual block multiplexing.

A virtual machine that is allowed to use the extended control feature of System/370 is provided with a full complement of 16 control registers, allowing virtual monitor calls, PER, extended channel masking, and dynamic address translation.

An extension to the normal virtual machine VMBLOK is built in when an extended control virtual machine logs on to CP. This ECBLOK contains the 16 virtual control registers, two shadow control registers, and several words of information for maintenance of the shadow control tables, virtual processor timer, virtual TOD clock comparator, and virtual PER event data.

When an extended-control virtual machine is first active, it has only the real page and segment tables provided by CP and operates entirely in second-level storage. CP determines when the virtual machine enters or leaves extended control or translate mode. CP also determines any changes in the virtual machine's operating mode. The virtual machine can load or store any control register, enter or leave extended control mode, take interruptions, and so forth without invoking the address translation feature.

If the virtual machine, already in extended control mode, turns on the translate bit in the EC mode PSW, CP examines the virtual control registers and builds the required shadow tables. (Shadow tables are required because the real DAT hardware can map only first-level storage.) CP determines whether control registers 0 and 1 contain valid information for use in constructing the shadow tables. Control register 0 specifies the size of the page and segment the virtual machine is using in the virtual page and segment tables. The shadow tables are always in the same format as the virtual tables.

This shadow segment table is constructed in first-level storage and initialized to indicate that all segments are unavailable. CP also constructs the shadow control registers 0 and 1. Shadow control register 0 contains the external interruption mask bits used by CP, mixed with the hardware controls and enabling bits from virtual control register 0. Shadow control register 1 contains the segment table origin address of the shadow segment table.

When the virtual machine is operating in virtual translate mode, CP loads the shadow control registers into the real control registers and dispatches the virtual machine. The immediate result of trying to execute an instruction is a segment exception. CP examines the virtual segment table in second-level storage. If the virtual segment is marked available, CP:

1. Allocates a segment of the shadow page table in the format specified by virtual control register 0

2. Sets the page table entries to indicate that the page is not in storage

3. Marks the segment available in the shadow segment table

4. Dispatches the virtual machine again.

The immediate result is an interruption (a paging exception), which refers to the virtual page table in second-level storage to determine whether the virtual page is available. If the page is not available, the paging interruption is reflected to the virtual machine. However, if the virtual page is marked in storage, the virtual page table entry determines which page of second-level storage is being referred to by the third-level storage address provided. CP next determines whether that page of second-level storage is resident in first-level storage at that time. If so, the appropriate entry in the shadow page table is filled in and marked in storage. If not, the required page is brought into the first-level storage and the shadow table filled in as above.

As the virtual machine continues execution, more shadow tables are filled in or allocated as the third-level storage locations are added. Whenever a new segment is referred to, another segment of shadow page tables is allocated. Whenever a new page is referred to, the appropriate shadow table entry is validated. No changes are made in the shadow tables if the virtual machine leaves translate mode unless it also leaves extended control mode. Dropping out of EC mode is the signal for CP to release all shadow page and segment tables and the copy of the virtual segment table.

Some situations require invalidating the shadow tables constructed by CP or even releasing and allocating them. Whenever CP pages out a page that belongs to a virtual relocating machine, it selectively invalidates the shadow page tables. If the stolen page is below the high-water mark, the shadow page table entry for the stolen page is invalidated. (The *high-water mark* is the highest contiguous address, starting from location zero, where the virtual system's real address equals the virtual system's virtual address.) If the stolen page is above the high-water mark and virtual machine assist is on, all of the shadow page tables above the high-water mark are invalidated when the virtual machine is about to be dispatched. The shadow tables are scanned to selectively invalidate shadow page table entries that map to the real page being stolen.

Reducing Purges When the Virtual Machine Dispatches New Address Space

To reduce the number of purges when the virtual machine dispatches a new address space (changes control register (CR1) values), VM/SP maintains a queue of segment table origins (STO) and associated shadow tables for the virtual machine.

To specify multiple shadow table support, use the SET STMULTI command. This command adds the segment table origin control block (STOBLOK) pointed to by the ECBLOK to the STO queue. The STOBLOK contains the the shadow segment table, information pertaining to it, and the virtual CR1 value. It also provides forward and backward queue pointers to the next STOBLOK on the queue. The first STOBLOK on the queue contains the shadow STO to be loaded into CR1 when the virtual machine is dispatched in translation mode. CP maintains the queue of STOBLOKs in the following manner:

1. If a virtual machine loads a new CR1 value, CP searches the queue of STOBLOKs for the virtual CR1 value.

2. If CP finds the proper STO, it places that STOBLOK first on the queue.

3. If CP does not find proper STO, it checks the maximum STO count.

    a. If the number of STOBLOKs equals the maximum STO count, CP steals the last STOBLOK, purges the shadow tables, and initializes the STOBLOK. The STOBLOK is reused by being chained first on the queue with the new virtual CR1 value.

    b. If the number of STOBLOKs is less than the maximum STO count, CP obtains free storage from VM/SP, and initializes the free storage area as the STOBLOK and chains it first on the queue.

Multiple shadow table support is controlled by the SET STMULTI command. The default minimum number of shadow tables is 3 and the maximum is 6 per virtual machine.

Eliminating and Reestablishing Shadow Table Bypass

Shadow table bypass, invoked by the SET STBYPASS command, allows CP to eliminate the shadow tables for an operating system running in the V=R area. When CP runs a V=R virtual machine, the shadow table for the V=R machine is identical to the virtual system's own page and segment tables, except for page zero. CP relocates the virtual machine's page zero (via the shadow table) to the highest real address within the V=R area. When STBYPASS is turned on, CP modifies the virtual operating system's page table to relocate virtual page zero to the highest real address. It can then dispatch the virtual machine with control register 1 pointing to the virtual page and segment tables.

To eliminate and reestablish shadow table bypass, use the SET STBYPASS command.

Note: If virtual machine assist is enabled on the system, the virtual machine must have the STFIRST directory option to issue the SET STBYPASS nnM/nnnnK command.

For the V=V User: This technique is based on several characteristics of VS systems:

- VS systems have a large area of addressing space starting with location zero where the virtual address is equal to the real address.
- This addressing space is common to each segment table when multiple segment tables are used (MVS or SVS address space).
- The VS system never pages within this fixed area.

Thus, you can establish an area starting at location zero where the second-level address equals the third-level address or virtual-virtual=virtual-real (VV=VR). A second-level address is the virtual address specified by the operating system operating in a first-level virtual machine; a third-level address is the address specified by a program running under control of the virtual machine guest. You can then establish the highest VV=VR address for a VS system. Because the second-level address is the same as the third-level address, a reverse translation allows the shadow tables to be indirectly indexed. Then, whenever VM/SP steals a page from the VV=VR area, it invalidates the shadow page table entry and executes a real PTLB (purge-translation-lookaside buffer) before redispatching the VS system's virtual machine.

In addition, whenever a shadow table is purged because a page frame is stolen from above the highest VV=VR address or the virtual machine executed a PTLB or LCTL, the invalidation starts above the highest VV=VR address. Thus, purge and revalidation time is reduced.

For the V=R User: You can use a V=R shadow table bypass technique to eliminate both the shadow tables and the overhead associated with maintaining them. This can be done by VM/SP changing the virtual operating system's page table to relocate virtual page zero to the highest real address in the V=R area. The virtual machine can then be dispatched pointing to its own page and segment tables.

Notes:

1. *With MVS single processor mode enhancement support, absolute page zero is made available to the MVS guest when single processor mode is set on.*

2. *If the MVS guest in single processor mode issues the SET STBYPASS VR command, CP issues an invalid option error.*

Eliminating Queue Drop Overhead for a Virtual Machine

VM/SP tries to optimize system throughput by monitoring the execution status of virtual machines. When a virtual machine becomes idle, VM/SP drops it from the run list. The virtual machine's page and segment tables are scanned, and resident pages are invalidated and put on the flush list.

VM/SP determines that a virtual machine is idle when it voluntarily suspends execution (by loading a virtual PSW with the wait state bit on, for example), and no high-speed I/O operation is active. Normally, this is an adequate procedure.

However, in certain special cases, an idle virtual machine that is dropped from a queue becomes active again sooner than expected. If this cycle of queue dropping and reactivation is executed repeatedly, the overhead of invalidating and revalidating the virtual machine's pages may become large.

The SNA VTAM service machine is an example of this special case. The VTAM service machine operates by processing an Inter-User Communication Vehicle (IUCV) message (or queue of messages), and then suspending execution until the next message arrives. VM/SP queue drops the VTAM service machine from the queue when it suspends execution. When the next message arrives, all the VTAM service machine's pages must be revalidated. If the message rate is moderate to high, the repeated queue dropping causes excessive overhead.

To control this situation, use the CP class A command SET QDROP userid ON/OFF [USERS]. If SET QDROP OFF is in effect for a virtual machine, the virtual machine is not dropped from the queue and its pages are not scanned or flushed.

If you specify SET QDROP OFF for a service virtual machine, system performance and throughput may improve when queue dropping would otherwise occur rapidly. But applying SET QDROP OFF indiscriminately

may degrade system throughput by defeating the page flush mechanism and forcing page stealing to take place.

A large overhead may be associated with a virtual machine being dropped from its queue during communications with a service machine for which the QDROP OFF specification is in effect. This can occur in small systems in which a high degree of virtual machine intercommunications occurs. If you specify SET QDROP userid OFF USERS, the QDROP OFF status is temporarily extended to any virtual machine communicating via VMCF or IUCV to the service virtual machine specified. The QDROP status for the "served" virtual machine remains in effect only while messages are outstanding between it and the service machine. Thus you can improve performance in systems that heavily use products such as VM/Interactive File Sharing (VM/IFS) or VM/Pass-Through Facility (PVM) (invoked via the CMS PASSTHRU command). This option will not improve performance in systems in which PVM is invoked via CP DIAL or with the SNA VTAM service machine, since the communication is with CP rather than another virtual machine.

To list the userids for which SET QDROP OFF and the USERS parameter have been specified, use the QUERY QDROP command (CP class A and E).

Improving Performance With the Virtual Machine Assist Feature

The Virtual Machine Assist Feature is a processor hardware feature that improves the performance of VM/SP. Virtual storage operating systems, which run in problem state under the control of VM/SP, use many privileged instructions and SVCs that cause interruptions that VM/SP must handle. When the virtual machine assist feature is used, many of these interrupts are intercepted and handled by the processor. Consequently, VM/SP performance is improved.

The virtual machine assist feature intercepts and handles interruptions caused by SVCs (other than SVC 76), invalid page conditions, and several privileged instructions. An SVC 76 is never handled by the hardware; it is always handled by CP.

Although the assist feature was designed to improve the performance of VM/SP, virtual machines may see a performance improvement because more resources are available for virtual machine users. For a list of processors on which the Virtual Machine Assist Feature is available, see the *VM/SP Planning Guide and Reference.*

Using the Virtual Machine Assist Feature

When you IPL VM/SP on a processor with the virtual machine assist feature, the feature is available for all VM/SP virtual machines. However, the class A or E SET command can make the feature unavailable to VM/SP and, subsequently, available again for all users. If you do not know whether the virtual machine assist feature is available to VM/SP, use the

class A and E QUERY command. For a complete description of the class A and E QUERY and SET commands, see the *VM/SP CP Command Reference.*

If the virtual machine assist feature is available to VM/SP when you log on to your virtual machine, it is also supported for your virtual machine unless you are running a second-level VM/370 or VM/SP system in your virtual machine. If your directory entry has the SVCOFF option, the SVC handling portion of the assist feature is not available when you log on. Use the class G SET command to disable the assist feature (or only disable SVC handling), or to enable the assist feature, or if the assist feature is available, to enable the SVC handling. Use the class G QUERY SET command to find whether you have full, partial, or none of the assist feature available. For details on the class G QUERY and SET commands, see the *VM/SP CP Command Reference.*

Restricted Use of the Virtual Machine Assist Feature

Certain interrupts must be handled by VM/SP. Consequently, VM/SP automatically turns off the assist feature in a virtual machine that:

- Has set an instruction address stop
- Is tracing SVC and program interrupts.

Since an address stop is recognized by an SVC interrupt, VM/SP must handle SVC interrupts while address stops are set. When you issue the ADSTOP command, VM/SP automatically turns off the SVC handling portion of the assist feature for your virtual machine. The assist feature is turned on again after the instruction is encountered and the address stop removed. If you issue the QUERY SET command while an address stop is in effect, the response shows that the SVC handling portion of the assist feature is off.

When a virtual machine issues a TRACE command with the SVC, PRIV, BRANCH, INSTRUCT, or ALL operands, the virtual assist feature is automatically turned off for that virtual machine. The assist feature is turned on again when the tracing is completed. If the QUERY SET command line is issued while SVCs or program interrupts are being traced, the response indicates the assist feature is off.

The virtual machine assist feature is not available to a second-level virtual machine, that is, a virtual machine that is running in a virtual machine.

Extended Control-Program Support: VM/370

Extended Control-Program Support: VM/370 (ECPS) extends, for specific privileged instructions, the hardware assistance that the virtual machine assist feature provides. ECPS also provides hardware assistance for frequently used VM/SP functions. ECPS improves VM/SP performance beyond the performance gains that the virtual machine assist feature provides.

ECPS consists of three functions:

- CP assist
- Expanded virtual machine assist
- Virtual interval timer assist.

CP assist provides hardware assistance for frequently used paths of specific CP functions.

Expanded virtual machine assist extends the hardware assistance that the virtual machine assist feature provides for the instructions LPSW, STNSM, STOSM, and SSM. In addition, expanded virtual machine assist provides hardware assistance for certain other privileged instructions.

Virtual interval timer assist provides hardware updating of the virtual interval timer at virtual address X'50'. Timer updating occurs only while the virtual machine is in control of the real processor. Virtual interval timer assist updates the virtual timer at the same frequency hardware updates the real timer, 300 times per second. Thus, virtual interval timer assist updates the virtual timer more frequently than CP updates it. Because the timer is updated more frequently, accounting routines might provide more accurate accounting data.

Using the Extended Control-Program Support: VM/370

You can control Extended Control-Program Support: VM/370 (ECPS) at two levels: the VM/SP system and the virtual machine.

At the VM/SP system level, ECPS is automatically enabled when the system is loaded (except for AP and MP systems in which ECPS is always disabled). You can use the class A command:

```
SET CPASSIST OFF
```

to disable both CP assist and expanded virtual machine assist. You can use the class A command:

```
SET SASSIST OFF
```

to disable only the expanded virtual machine assist facility and the virtual interval timer assist function of ECPS.

At the virtual machine level, whenever ECPS is enabled on the system, both expanded virtual machine assist and virtual interval timer assist are automatically enabled when you log on. If you issue the class G command:

```
SET ASSIST OFF
```

the expanded virtual machine assist, the virtual interval timer assist, and the existing virtual machine assist are disabled. If you issue:

```
SET ASSIST NOTMR
```

only the virtual interval timer assist is disabled. If CP assist is disabled for the system, and you issue the class A command:

```
SET SASSIST ON
```

the virtual machine assist is enabled. To enable virtual machine assist and virtual interval timer assist for your virtual machine, issue the class G command:

```
SET ASSIST ON TMR
```

Restricted Use of ECPS

The restrictions on the use of ECPS are the same as those described for the virtual machine assist feature with one addition. When a virtual machine traces external interrupts, the virtual interval timer assist is automatically disabled. When external interrupt tracing is completed, virtual interval timer assist is reenabled.

Improving Channel Use

Using the Virtual Block Multiplexer Channel Option

Virtual machine SIO operations are simulated by CP in three ways:

- Byte-multiplexer
- Selector
- Block multiplexer channel mode.

Virtual byte-multiplexer mode is reserved for I/O operations that apply to devices allocated to channel zero.

In virtual selector channel operations, CP reflects a busy condition (condition code 2) to the virtual machine's operating system if the system tries a second SIO to the same device, or another device on the same channel, before the first SIO is completed.

Block multiplexer channel mode is a CP simulation of real block multiplexer operation; it allows the virtual machine's operating system to overlap SIO requests to multiple devices connected to the same channel. If you select block multiplexer mode of operation, the virtual machine's throughput may increase, particularly for systems or programs designed to use the block multiplexer channels.

Note: CP simulation of block multiplexer processing does not reflect channel available interruptions (CAIs) to the user's virtual machine.

You can select the channel mode of operation for the virtual machine by a system generation DIRECTORY OPTION operand or by the CP DEFINE command. Enter the DEFINE command as:

```
DEFINE CHAN BMX
```

Alternate Path Support

With the Two-Channel Switch and Two-Channel Switch Additional Features, alternate path support for DASD or tape provides for up to four channels on one control unit to be attached to VM/SP (up to 2 channels per control unit in multiprocessing configurations). In addition, one device can be attached to two logical control units, providing support for the String Switch feature. This allows the control program up to eight paths to a given device when the maximum number of alternate channels and alternate control units is specified.

When an I/O request is received for a device that has alternate paths defined, and the primary path is unavailable, VM/370 searches for the first available path beginning with the first alternate path. It examines successive alternate paths, if required, until an available path is found. If no available path to the device exists, alternate path I/O scheduling queues the request on multiple busy/scheduled paths, and the first path to become available is the path the I/O request is started on.

The *VM/SP System Logic and Problem Determination Guide Volume 1 (CP)* describes how the scheduler selects the alternate path.

MVS/System Extensions Support

The MVS/System Extensions support in VM/SP allows an MVS system running in a virtual machine to use the enhancements available in the MVS/System Extensions Program Product (Program No. 5740-XE1) if the System/370 Extended Facility or System/370 Extended Feature is present on the hardware.

Included in the MVS/System Extensions Program Product enhancement is the use of:

1. The System/370 Extended Facility for the 303x and the 308x processors, or
2. The System/370 Extended Feature for the System/370 Model 158 and 168 processors, or
3. ECPS:MVS for the 4341.

Note: An RPQ (MK3272) is available for the 158-3 processor that allows the coexistence of virtual machine assist and System/370 Extended Facility (S370E) and VM/370 Extended Feature. Thus, an MVS/SE virtual machine can run under VM/SP with virtual machine assist active on a 158-3

processor. ECPS:MVS and ECPS:VM/370 are mutually exclusive in the 4341 Model Group 1 and 4341 Model Group 2. The control storage expansion feature of the Model Group 2 allows coexistence of ECPS:MVS and ECPS:VM/370.

The System/370 Extended Facility and System/370 Extended Feature, and ECPS:MVS are enabled by the MVS/System Extensions support as defined by the directory OPTION statement or via the CP SET command. For details, refer to the Planning Guide and Reference and to the *VM/SP CP Command Reference*, respectively.

MVS/System Extensions support includes:

- Low address protection facility[4]
- Common segment facility[4]
- Special MVS instruction operation facilities.

Low Address Protection Facility

Low address protection protects against improper storing by instructions using logical storage addresses in the range 0-511. It prevents inadvertent program destruction of those storage locations that the processor uses to fetch new PSWs during interruption processing. Low address protection does not apply to the storing of status by the processor (for example, old PSWs, logout data), nor does it apply to any channel stores (for example, CSW or LCL).

Bit 3 of control register 0 is the low address protection bit, and controls whether or not store instructions using logical addresses in the range 0 to 511 are permitted. When this bit is zero in real control register zero, stores are permitted; when this bit is one, stores are not permitted. When an instruction tries to store at an address in the range 0 to 511 and low address protection applies, the contents of the storage area addressed by the instruction are not modified. Execution of the current instruction is terminated or suppressed, and a protection exception occurs.

Common Segment Facility

The common segment facility allows addressing segments to be classified as private or common. If bit 30 of the segment table entry for a given segment is 1, the segment is a common segment; otherwise it is private. A private segment table entry and the page table it designates can be used with only the segment table origin (STO) that designates the segment table in which the segment table entry resides. A common segment table entry and the page table it designates may continue to be used for translating addresses even though a different STO is specified by changing control register 1.

---

[4] ECPS:MVS is identical to the Extended Facility, except that the Low Address Protection Facility and the Common Segment Facility are not included.

Special operations and instructions in the MVS/System Extensions Program Product that enhance MVS operations are handled by System/370 Extended Facility or System/370 Extended Feature, and are described in *System/370 Extended Facility*, GA22-7022. Invalidate Page Table Entry (IPTE) and Test Protection (TPROT) instructions described in this publication are simulated in VM/SP.

Enabling MVS/System Extensions Support

To enable the MVS/System Extensions support for all virtual machines, use the class A SET S370E ON command. The general user uses the class G SET 370E ON command (or 370E option on the directory OPTION control statement), to enable the support for a particular virtual machine.

Note: The virtual machine must be running with ECMODE on to set 370E on.

Improving Throughput of an OS/VS2 MVS AP or MP System

When an OS/VS2 MVS system runs on a multiprocessor under VM/SP, without using single processor mode, MVS runs in uniprocessor mode. That is, MVS programs do not execute simultaneously on both processors. Therefore, MVS does not attain the level of throughput it could attain were it running in multiprocessor mode.

To improve the throughput of an OS/VS2 MVS system in an AP or MP system, run MVS in the V=R machine and use single processor mode. In this mode, MVS has exclusive use of one processor while VM/SP and the V=R machine (running MVS) use the other processor. In other words, MVS runs on two processors instead of one. This improves MVS's throughput.

The throughput of an OS/VS2 MVS system in an AP or MP system running under VM/SP and using single processor mode is higher than the throughput would be were single processor mode not used. However, single processor mode may reduce the throughput of VM/SP and virtual machines not using the V=R area.

Single processor mode cannot improve the throughput of a VM/SP attached processor or multiprocessor system. A VM/SP AP or MP system initialized (by IPL) in the V=R machine with single processor mode on runs in uniprocessor mode.

Two commands provide operator control of single processor mode. SPMODE, a class A command, turns single processor mode on or off. QUERY, a class A or G command, indicates whether single processor mode is on or off.

Detailed instructions for turning single processor mode on or off are in *VM Running Guest Operating Systems.*

Switching the System Control Program (SCP) to or from Native Mode

Sometimes an installation benefits from switching an SCP to or from native mode. For example, when obtaining the best possible performance from an SCP is important, switch it to native mode. To do different kinds of work simultaneously, switch the SCP from native mode to the VM/SP environment.

Installations have always had the capability to switch an SCP to or from native mode, but to do so has been time consuming. Switching an SCP to native mode meant quiescing the SCP and VM/SP and then initial program loading the SCP. To return the SCP to the VM/SP environment meant quiescing the SCP and then initial program loading VM/SP and the SCP.

Dynamic SCP transition to or from native mode enables an operator to dynamically switch an SCP to or from native mode. Switching to native mode, there is no longer a need to quiesce or reinitialize (via IPL) the SCP. The SCP continues to run and can do productive work. Switching back to the VM/SP environment, there is no longer a need to quiesce the SCP or IPL VM/SP or the SCP.

Before switching an SCP to or from native mode, an operator must prepare VM/SP and the SCP for the switch: for example, all users except the VM/SP operator and the operator on the V=R machine must log off VM/SP. Detailed instructions on preparing the systems and switching to or from native mode are in *VM Running Guest Operating Systems.* The following discussion highlights the switching process and defines precautions that must be observed.

To switch an SCP to native mode, it must be running in the V=R machine. The VM/SP operator then prepares VM/SP and the SCP for the switch. To complete the switch, the operator issues the QVM command (quiesce VM).

After the switch to native mode is completed, there are two areas of real storage that must not be altered. Addresses 0-7 contain the restart PSW (program status word) used to make the transition back to the VM/SP environment. Storage above the upper limit of the V=R area contains the VM/SP nucleus. Altering either area may make it impossible to return to the VM/SP environment.

To return the SCP to the VM/SP environment, an operator uses the System/370 restart facility. After stopping the processor, the operator stores the value X'FF' into the real storage address located eight bytes prior to the address pointed to by the restart PSW. To complete the switching process, the operator restarts the processor.

Caution: This process does not work unless the SCP was switched to native mode with the QVM command.

The performance of an SCP switched to native mode depends on the size of the V=R area. The SCP's performance will be the same as it would attain if it initialized (by IPL) directly on a hardware configuration identical to the V=R machine's configuration with a real storage size equal to the storage size of the V=R area. In other words, the larger the V=R area, the better the SCP performs.

You can can switch to or from native mode using the procedures just described for:

- OS/VS1 running without VM VS1 Handshaking
- OS/VS2 SVS
- OS/VS2 MVS.

Performance

Chapter 14. Performance Observation and Analysis

You can use the INDICATE, QUERY SRM, and MONITOR commands to measure system performance.

INDICATE
: Displays load conditions while the system is running.

QUERY SRM
: Displays information about system activities and counters.

MONITOR
: Samples and records a wide range of data. Keywords in the MONITOR command enable the collection of data and identify the various data collection classes. Data can be collected from more than one class at a time. Other keywords control the recording of collected data on tape for later examination and reduction.

Using the INDICATE Command

Use the INDICATE command to check the system for persistently heavy loads, to judge when it is best to apply additional scheduling controls (if appropriate).

Use the INDICATE command to display the basic uses of and contentions for major system resources (possible bottleneck conditions) and characteristics of the active users and the resources that they use.

With the INDICATE command, virtual machine users can observe the basic smoothed conditions of contention and use of the primary resources of processor and storage. The INDICATE command allows them to base their use of the system on an intelligent guess of what the service is likely to be. Over a period of time, virtual machine users relate certain conditions of service to certain utilization and contention figures and know what kind of responses to expect when they start their terminal session.

The INDICATE command lets general users and the system analyst display on a console at any time the use of and contention for major system resources. They can also display the total amount of resources used during the terminal session and the number of I/O requests. If they use the INDICATE command before and after the execution of a program, users can determine the execution characteristics of that program in terms of resource use.

The system analyst can identify active users, the queues they are using, their I/O activity, their paging activity, and many other user characteristics and use data.

The system analyst can use the data on system resource usage and contention to monitor the performance of the system. The analyst can thus be aware of heavy load conditions or low performance situations that may require the use of more sophisticated data collection, reduction, and analysis techniques for resolution.

The VM/SP Scheduler maintains exponentially smoothed values for data provided by the LOAD option. Every few seconds (the exact interval depends on the processor model), the scheduler calculates the total activities for variables such as CP and storage use for the most recent interval and factors them into a smoothed wait value:

New smoothed value =
0.75 old smoothed value + 0.25 current interval

Thus, only one-fourth of the most recent interval is factored into the new smoothed value.

The remaining INDICATE components are sampled prior to a user being dropped from a queue. Because of the frequency of this event, the remaining components are subject to a heavier smoothing than the wait time. The general expression for the smoothing is:

nsv = ((rate - int) (osv) / rate) + civ

where:

nsv is the new smoothing value

rate is either the history interval (hrate) of 8 minutes, or data interval (drate) of 75 seconds

int is the current interval (time period being tested)

osv is the old smoothing value

civ is the current interval value (results found during the current interval (int))

Other operands of the command allow users to obtain other performance information that enables them to understand the reasons for the observed conditions. For the formats of the class G INDICATE command and the class E INDICATE command, see the *VM/SP CP Command Reference.*

The section "VM/SP Performance Options" on page 87 contains detailed information on favored execution. For information on the setting of favored execution options, refer to the *VM/SP Operator's Guide.*

Using the QUERY SRM and SET SRM Commands

Use the QUERY SRM and SET SRM commands to query and/or change internal system activity counters or parameters. Formats for the QUERY SRM and SET SRM commands are contained in the *VM/SP CP Command Reference.*

Use the Class E QUERY SRM command to display the following information:

- Current number of pageable pages
- Size of the dispatching time slice
- Setting of the maximum working set estimate
- Maximum drum page allocation limit
- Current page migration counters
- Unused segment elapsed time as criteria for page migration
- Current PCI flag setting mode for 2305 page requests
- Maximum page bias value
- Current interactive shift bias value
- Moveable head page migration limit.

Use the class E SET SRM command to set some of the system variables that can affect the values displayed by the QUERY SRM command.

Using the MONITOR Command

VM/SP Monitor collects data by:

- Handling interruptions caused by executing MONITOR CALL (MC) instructions.

- Using timer interruptions to give control periodically to sampling routines.

MONITOR CALL instructions with appropriate classes and codes are embedded in strategic places throughout the main body of VM/SP code (CP). When a MONITOR CALL instruction executes, a program interruption occurs if the particular class of MONITOR CALL is enabled. The classes of MONITOR CALL that are enabled are determined by the mask in control register 8. For the format and function of the MONITOR CALL instruction, refer to the *System/370 Principles of Operation.* The format of control register 8 is as follows:

xxxx	xxxx	xxxx	xxxx	0123	4567	89AB	CDEF

where:

x indicates an unassigned bit.

0-F (hexadecimal) indicates the bit associated with each class of the MONITOR CALL.

When a MONITOR CALL interruption occurs, the CP program interruption handler (DMKPRG) transfers control to the VM/SP monitor interruption handler (DMKMON) where data collection takes place.

Sixteen classes of separately enabled MONITOR CALL instructions are possible, but only eight are implemented in the VM/SP Monitor.

Monitor output consists of event data and sampled data. MONITOR CALL instructions in the VM/SP code obtain data. Sampled data is collected following timer interruptions. All data is recorded as though it were obtained through a MONITOR CALL instruction. This simplifies the identification of the records.

The following table shows the type of collection mechanism for each Monitor class:

Monitor Class	Class Name	Collection Mechanism
0	PERFORM	Timer requests
1	RESPONSE	MC instructions
2	SCHEDULE	MC instructions
3[5]	--	--
4	USER	Timer requests
5	INSTSIM	MC instructions
6	DASTAP	Timer requests
7	SEEKS	MC instructions
8	SYSPROF	Collected with class 2

Another function, separate from the VM/SP Monitor, is also handled by the MONITOR command. The MONITOR command can stop and start collecting CP internal trace table data, which is *not* initiated by MONITOR CALLs.

Note: The VM/SP Monitor record format and contents are shown in Appendix, "VM/SP Monitor Tape Format and Content"

The class A and E MONITOR command:

- Stops and starts CP internal trace table data collection.

---

[5] There is no class name for monitor class 3, but it is reserved.

- Displays the status of the internal trace table and each implemented class of VM/SP Monitor data collection.
- Displays the specifications for automatic monitoring defined by the SYSMON macro in DMKSYS.
- Displays those specifications for automatic monitoring that are overridden by Monitor commands.
- Displays whether the tape or spool file is the recording medium.
- Starts and stops VM/SP data collection using tape or spool file. It also closes the spool file, if desired.
- Specifies VM/SP monitor classes of data collection enabled, number of buffers used, and time of data collection. It also specifies other options which override the specifications for automatic monitoring on the SYSMON macro contained in DMKSYS.
- Specifies the interval to be used for timer driven data collection.
- Specifies direct access devices to be included or excluded from a list of devices. The list defines direct access devices for which CP is to collect data for the SEEKs class.

See the *VM/SP CP Command Reference* for the format and details of the MONITOR command.

Implemented Classes

The following MONITOR CALL classes correlate with the corresponding classes in control register 8. Refer to the *System/370 Principles of Operation* for details of the MC instruction and the bits in control register 8.

Monitor Class	Keyword	Data Collection Function
0	PERFORM	Samples system resource usage data by accessing system counters of interest to system performance analysts.
1	RESPONSE	Collects data on terminal I/O. Simplifies analyses of command usage, user, and system response times. It can relate user activity to system performance. This class is invalid. No data RESPONSE data (MONITOR class 1) can be collected unless the system programmer sets the TRACE(1) bit to a 1 in the LOCAL COPY file and reassembles DMKMCC. Refer to the *VM/SP Installation Guide* for details.

Figure 8 (Part 1 of 3). MONITOR CALL Classes

Monitor Class	Keyword	Data Collection Function
2	SCHEDULE	Collects data about scheduler queue manipulation, monitors flow of work through the system, and indicates the resource allocation strategies of the scheduler.
3	-------	Reserved.
4	USER	Periodically scans the chain of VMBLOKs in the system, and extracts user resource utilization and status data.
5	INSTSIM	Records every virtual machine privileged instruction handled by the control program (CP) standard simulation routines (DMKPRV, DMKPRW). Because simulation of privileged instructions is a major source of overhead, this data may lead to methods of improving performance. The fast path simulation routines (DMKFPS) result in significantly less control program overhead than the standard paths. Therefore, privileged instructions simulated by DMKFPS are not recorded. If the VMA feature is active, the number of privileged instructions that are handled by the control program is reduced for those virtual machines that are running with the feature activated. Privileged instructions handled by the VMA are not recorded.
6	DASTAP	Periodically samples device I/O activity counts (SIOs), for tape and DASD devices only. DASTAP samples only those tapes and DASD devices that are online when the MONITOR START command is issued. It is possible that the number of DASD and tape devices defined in DMKRIO may exceed 291 (the maximum number of MONITOR DASTAP records that fit in a MONITOR buffer). The following algorithm determines which devices are monitored: 1. If the total number of DASD and tape devices that are on-line is less than or equal to 291, all on-line DASD and tape devices are monitored. 2. If the total number of on-line DASD devices is less than or equal to 291, all on-line DASD devices are monitored. 3. Otherwise, the first 291 on-line DASD devices are monitored.

Figure 8 (Part 2 of 3). MONITOR CALL Classes

Monitor Class	Keyword	Data Collection Function
7	SEEKS	Collects data for every I/O request to DASD. Reveals channel, control unit, or device contention and arm movement interference problems. *Note:* When NOTRANS is in effect for a virtual=real machine, no meaningful data is collected. No data is collected for TIO or HIO operations. For SIO operations, data is collected when the request for the I/O operation is initially handled and again when the request is satisfied. This means that a single SIO request could result in two MONITOR CALLs. For example, if the request gets queued because the device is already busy, then a MONITOR CALL would be issued as the request is queued. Later, when the device becomes free and is restarted, a second MONITOR CALL is issued. Both MONITOR CALLs collect the same data, except that the first records a nonzero number of queued requests. The second records zero I/O requests in the queue. If the request for I/O is satisfied without being queued, only one MONITOR CALL results. In this case, too, the count of I/O requests queued for the device is zero.
8	SYSPROF	Collects data complementary to the DASTAP and SCHEDULE classes to provide a more detailed "profile" of system performance through a closer examination of DASD utilization.

Figure 8 (Part 3 of 3). MONITOR CALL Classes

Monitor Response to Special Tape Conditions

Suspension

When I/O to the tape is requested, the device may still be busy from the previous request. If this occurs, two data pages are full and data collection must be temporarily suspended. Control register 8 is saved and then set to zero to disable MONITOR CALL program interruptions and timer data collection. A running count is kept of the number of times suspension occurs. The current Monitor event is disregarded. When the current tape I/O operation ends, the next full data page is scheduled for output. MONITOR CALL interruptions are reenabled (control register 8 is restored), a record containing the time of suspension, the time of resumption, and the suspension count is recorded and data collection continues. Note that suspension records can cause the results of analyzing monitor data to be unpredictable. The suspension count is reset to zero when the MONITOR STOP TAPE is issued. If a MONITOR command is issued when monitor is suspended, a message is displayed to the invoker of the command stating:

```
SEEK, STOP, OR CLOSE CMD IN PROGRESS, RETRY
```

Unrecoverable Tape Error
When an unrecoverable error occurs, DMKMON receives control and tries to write two tape marks, rewind, and unload the tape. The use of the tape is discontinued and data collection stops. The operator is informed of the action taken. Whether or not the write-tape-marks, rewind, and unload are successful, the tape drive is released.

End-of-Tape Condition
When an end-of-tape condition occurs, DMKMON receives control. A tape mark is written on the tape and it is rewound and unloaded. The monitor is stopped and the operator is informed of the action taken.

Initial Program Load
MONITOR START CPTRACE is active after real system IPL (manual or automatic). The monitor tape data collection is off after IPL. If automatic performance monitoring is specified in the SYSMON macro and IPL occurs within the range of the TIME operand of the SYSMON macro, monitor data collection to a spool file is started.

System Shutdown
If the monitor data collection to a spool file is taking place, a system shutdown causes closing of the file and termination of monitoring. If data collection is to tape, a system shutdown implies a MONITOR STOP TAPE command. Normal command processing for the MONITOR STOP TAPE function is performed by the system.

System Failure
If the VM/SP system fails and data collection to a spool file is active, the spool file is closed and preserved, except for the last buffer. If the VM/SP system fails and data collection is active on tape, an attempt is made to write two tape marks, rewind, and unload the tape. If the tape drive fails to rewind and unload, be sure to write a tape mark before rewinding and unloading the tape. Monitor data collection is terminated by the system failure.

I/O Devices
If monitor data collection is active using tape, a supported tape drive must be dedicated to the system for the duration of the monitoring. For accounting purposes, all I/O is charged to the system.

Monitor Output
Monitor output requires that you have the IBM program product program VMMAP (Virtual Machine Monitor Analysis Program, 5664-191) or some other user application program to read the file and process it.

Monitor Data Volume and Overhead

Use of the monitor usually requires that three pages be locked in storage for the entire time the monitor is active; however, only two pages are required if the single buffer option is used with only the PERFORM class of data collection enabled. This reduces by three the number of page frames available for paging and can affect the performance of the rest of the system when there is a limited number of page frames available for paging.

PERFORM
: This class of data collection is activated once every 60 seconds (or as defined by the MONITOR INTERVAL command), and records system counters relevant to performance statistics. It is, therefore, a very low overhead data collection option.

RESPONSE
: This class collects terminal interaction data and, because of the human factor, has a very low rate of occurrence relative to processor speeds. Consequently, this class causes negligible overhead and produces a low volume of data.

SCHEDULE
: This class records the queue manipulation activity of the scheduler and generates a record every time a user is added to the eligible list, added to queue1, queue2, or queue3, or removed from queue. The recording overhead is very low.

USER
: This class of data collection is active once every 60 seconds (or as defined by the MONITOR INTERVAL command). Data is extracted from each user's VMBLOK, including the system VMBLOK. The overhead incurred is comparable with that of the statistical data of the PERFORM class; however, it increases with the number of users logged onto the system.

INSTSIM
: This class of data collection can give rise to large volumes of data because of the frequency of privileged instructions in some virtual machines. This may increase overhead significantly. It should be activated for short periods of time and preferably, though not necessarily, when other classes of data collection are inactive. If the Virtual Machine Assist feature is active for the virtual machine, the data volume and, consequently, the CP overhead may be reduced.

DASTAP
: This class of data collection samples device activity counts once every 60 seconds (or as defined by the MONITOR INTERVAL command) and is a very low source of overhead, similar to the PERFORM and USER classes.

SEEKS
This class of data collection can give rise to large volumes of data because every start I/O request to DASD is recorded by a MONITOR CALL. It should be activated for short periods of time and preferably, though not necessarily, when other classes of data collection are inactive.

SYSPROF
This class of data collection is complementary to the SCHEDULE and DASTAP classes and results in a small amount of additional overhead. It obtains more refined data on DASD resource usage.

Performance for Time-Shared Multibatch Virtual Machines

First you must determine how many similar users can be run concurrently on a given configuration before the throughput of individual users becomes unacceptable.

Monitoring Recommendations

To simplify and automate the collection of performance data, use the automatic monitoring facilities. You should also set up a virtual machine to analyze and report the collected data. The Virtual Machine Monitor Analysis Program (VMMAP) does such a task. (For more information about this program and for details about ordering VMMAP, see *VMMAP General Information*.) You should use VMMAP or user-written analysis programs on a daily basis to analyze the collected data. Run such analysis programs preferably at off-peak hours to minimize the effect on the performance of the system doing data reduction. Initially, analyze the data collected with MONITOR default options to establish a familiarity with the load environment and performance profile of each virtual machine system and its effect on CP.

Once you establish a performance profile for each system and associated virtual machines, you should be able to detect points of contention between processor(s) storage, I/O, and paging subsystems.

Normally, you should use the spool file monitoring options. However, if large volumes of monitor data are to be collected, then use monitoring to tape. Tape is also useful if benchmarking is set up frequently and all of the new monitor trace and sampled data must be archived for possible future use. The default mode of operation of the Performance/Monitor Analysis Program is to keep the condensed ACUM files and not the raw data.

If you need SEEKs data, use a sampling technique. One technique is to use a CMS EXEC procedure to enable SEEKs for ten seconds every ten minutes. This would produce SEEKs data while limiting the volume of data collected. An alternative is to create a list of devices for which data for the SEEKs class is to be collected. CP collects data for only those devices in the list. To create the list, use the INCLUDE or EXCLUDE options of the MONITOR command's SEEK operand. If data is collected for only a few devices, consider collecting data for longer periods of time.

Load Environments of VM/SP

Two distinct uses of VM/SP can be readily identified: The system may serve mostly time-sharing virtual machines running batch jobs, with interactive machines performing minor support roles; or, the system may primarily be required to provide good interactive time-sharing services in the foreground, with a batch background absorbing spare resources of real storage and processor. Because of these distinct uses, there may be some differences in criteria for acceptable performance.

After determining the minimum acceptable performance, perform external observations of turnaround time on benchmarks and specify a point beyond which adding more users would be unacceptable. However, when that point is reached, you must do more sophisticated internal measurement to determine the scarcest resource and how the bottleneck can be relieved by additional hardware or by reassigning resources.

Several conditions can result from different bottlenecks. They are:

- There is too little real storage for the number of contending users. The run list can accommodate only a small portion of the eligible users and each user is dispatched so infrequently that response time becomes intolerable.

- Storage may be adequate to contain the working sets of contending users, but the processor is being shared among so many users that each is receiving inadequate attention for good throughput.

- Real storage space may be adequate for the processor, and a high speed drum is used for paging; however, some virtual storage pages of some users have spilled onto slower paging devices because the drum is full. With low levels of multiprogramming, user page wait can become a significant portion of system wait time. Consequently, processor utilization falls and throughput deteriorates.

- Storage, processor, and paging resources are adequate, yet several users are heavily I/O-bound on the same disk, control unit, or channel. In these circumstances, real storage may be fully committed because the correct level of multiprogramming is selected, yet device contention is forcing high I/O wait times and unacceptable processor use.

Obtain estimates of typical working set sizes to determine how well an application may run in a multiprogramming environment on a given virtual storage system. A measure of the application's processor requirements may be required for similar reasons. Measurements may be required on the type and density of privileged instructions a certain programming system may execute, because, in the virtual machine environment, privileged instruction execution may be a major source of overhead. If the virtual machine environment is used for programming development, where the improvement in programmer productivity outweighs the disadvantages of extra overheads, the above points may not be too critical. However, if throughput and turnaround time are important, then the converse is true,

and the points need close evaluation before allocating resources to a virtual machine operation.

High levels of multiprogramming and over-commitment of real storage space lead to high paging rates. High paging rates can indicate a healthy condition, but you should be concerned about page stealing. Get evidence that this rate is maintained at an acceptable level. A system with a high rate of page stealing is probably thrashing.

Performance of Mixed Mode Systems

Most of the conditions for good performance, established for the time-shared batch systems, apply equally well to mixed mode systems. However, two additional criteria are important for evaluating system performance. First, in all circumstances, priority should be given to maintaining good interactive response and nontrivial tasks must be kept in the background. Second, background tasks, no matter how large, inefficient, or demanding, should not be allowed to dominate the overall use of the time-sharing system. In other words, in mixed mode operation, users with poor characteristics are discriminated against for the sake of maintaining a healthy system for the remaining users.

A number of other conditions are more obvious and straightforward. You need to measure response and determine at what point it becomes unacceptable and why. Studies of time-sharing systems have shown that a user's rate of working is closely correlated with the system response. When the system responds quickly, the user is alert, ready for the next interaction, and thought processes are uninterrupted. When the system response is poor, the user becomes sluggish.

For interactive environments, analyze command use. Average execution time of the truly interactive commands can provide data for validation of the Queue 1 execution time.

Part 3: Hardware Considerations

This section contains information about processor features, printers, and other peripheral devices.

Chapter 15. DASD Operations

This chapter describes three programs that perform operations on direct access storage devices: Device Support Facilities, Format/Allocate Program, and DASD Dump Restore (DDR).

Device Support Facilities

Device Support Facilities is a program used with IBM operating systems to perform various operations on direct-access storage devices. It replaces IBCDASDI, INITDISK, and SURFANAL in that it can:

- Initialize direct-access storage volumes so that they can be used in OS/VS or DOS/VSE systems
- Inspect a volume for defective tracks
- Reformat the volume label, and IPL bootstrap and program records
- Provide an analysis function for nonremovable DASD, both CKD and FBA.

See *Device Support Facilities User's Guide and Reference (Current Release)* , which describes procedures for initializing, formatting, and analyzing disk space.

Disk initialization and alternate track assignment should be performed by the Device Support Facility stand-alone utility, which is on the CMS system disk with a fileid IPL DSF S2.

All direct access volumes used by the VM/SP system (for paging, spooling, system residence, directory, or temporary disk allocation) must be properly labeled, formatted, and allocated. The CP Format/Allocate service program prepares disks for use by CP.

All direct access volumes (including both real disks and minidisks) to be used by VSAM under CMS, OS, or DOS, must be formatted by the device support facility utility.

If certain information in the OS Format 4 label on track 0 cylinder 0 is destroyed, no additional alternate tracks can be assigned by the device support facility until the volume is reformatted by the device support facility:

- The CMS Format Program destroys the OS Format 4 label when it formats a volume or when it formats a minidisk whose origin is cylinder 0 of the volume.
- The device support facility and the CP Format/Allocate program (IPL FMT) preserve the information in the OS Format 4 label.

Format/Allocate Service Program

The Format/Allocate service program formats, allocates, and labels direct access volumes for paging, spooling and CP file residence. This service program is executed as part of CP system generation procedures and may also be executed as a stand-alone program to:

- Format direct access volumes for CP use
- Allocate specific disk areas to particular functions or to CP use
- Write six-character volume serial number labels.

Note: The Format/Allocate program should be used with care since it destroys any existing data. Also, user minidisks and temporary minidisks must not begin on real cylinder zero of CP-owned volumes, because information critical to CP is stored in that cylinder.

An object deck version of the CP Format/Allocate service program is a stand-alone program and can be loaded from a virtual or real card reader into a virtual or a real machine. (If run in a virtual machine, the virtual machine must have write access to the volume being formatted.) The program accepts control statements from the operator's system console (commands) or from the IPL device (card reader).

Note: I/O error messages DMKFMT736E and DMKFMT735E may be issued if an available path to the device cannot be found after an appropriate number of retries. High activity can cause this situation.

CP Disk Formats

Cylinders used by CP for paging, spooling, and so on, must be preformatted (using FMT) with fixed length unblocked records of 4096 bytes.

It is important to note the differences in the terminology for fixed-block DASD devices (for example, 3310) and count-key-data DASD devices (for example, 3330).

The basic unit of DASD space used by the VM/SP system is the page. A page contains 4096 bytes of data. On count-key-data devices, a page is recorded in a DASD record data area that is 4096 bytes in size. The count area of this record indicates the location (by cylinder and track) and size

(4096 bytes) of the recorded page. The preformatting of count-key-data volumes initializes the count areas and sets the data areas to zero.

On FBA devices, a page is recorded in eight successive blocks of data. Each block contains 512 bytes of a page. Control data, such as the count area for count-key-data, is not required. Preformatting FBA volumes sets all the pages to zero.

Format for CP-Owned CKD Devices

The unit of formatting or allocation for count-key-data devices is the cylinder. After you decide how many pages of DASD space are required for system operation, convert the number to a corresponding number of cylinders for communication with the Format/Allocate program.

Count-key-data DASD capacities when formatted for CP use (4096-byte pages) are:

Device	Records/ track	Tracks/ cylinder	Highest cylinder
2305-1	3	8	47
2305-2	3	8	95
3330	3	19	403
3330-11	3	19	807
3340-35	2	12	347
3340-70	2	12	695
3350	4	30	554
3375	8	12	958
3380	10	15	884
3380(AE4/BE4)	10	15	1769

The format operation writes 4096-byte blocks on all cylinders being formatted. The service program does write-checking to verify that parts of the track are not defective. A count is maintained of pages with read check errors detected during the format operation. At the completion of the format operation, the count of the pages with read check errors is printed.

Format for All Cylinders Except Cylinder Zero For example, the 3330 track format for all formatted cylinders except cylinder 0 is shown in Figure 9 on page 124.

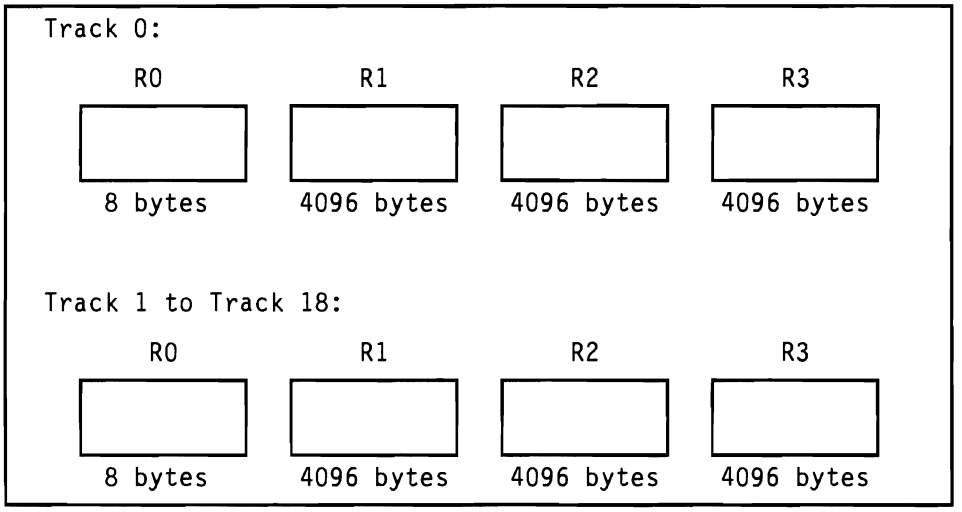


Figure 9. Format of 3330 Cylinders for Use by CP

Format for Cylinder Zero: All volumes containing space for CP use (paging, spooling, and so on) must have a properly formatted cylinder 0. The only service program that can do this is the Format/Allocate program .

Cylinder 0 is formatted like other cylinders except that the space associated with the first three 4096-byte blocks is reserved for system use. This area is then formatted as illustrated in Figure 10.

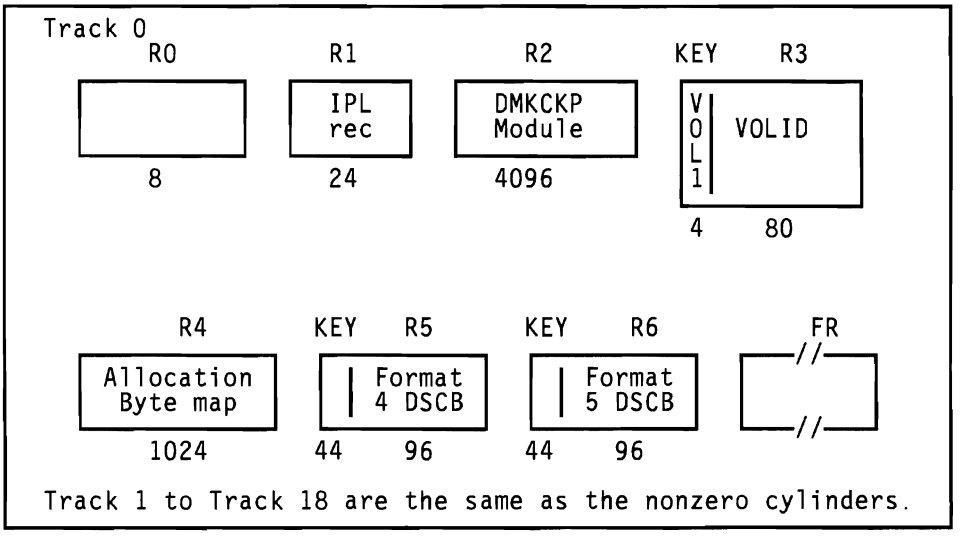


Figure 10. 3330, 3340, 3350 or 3380 Cylinder 0 Format

The contents of each record in cylinder 0 track 0 are:

R0 Nothing.

R1 IPL record. Puts the system into wait state if storage volume is loaded before CP nucleus is built.

R2 Checkpoint record. Used by CP to save and retrieve information for a warm start.

R3 Volume label. Same as OS VOL1 label. On CP system residence volume, area in data record marks the beginning of the system directory. A label is automatically written when cylinder 0 is formatted. The owner field of the label record contains "CP370" if there is allocation data present in R4.

R4 Allocation Byte Map. Each byte identifies a cylinder and specifies its usage (paging, spooling, directory, and so on). This map is filled in by the ALLOCATE function of the Format/Allocate service program. For certain 3380 models, this record is greater than 1024 bytes to accommodate a larger number of cylinders.

R5 Format 4 OS DSCB type label (for compatibility with OS). Also, the device support facility program uses this label to keep a record of how many alternate tracks remain available for assignment on this disk. The Format/Allocate program will preserve this information by first reading it from any existing Format 4 label, and then writing it back in the new label.

R6 Format 5 OS DSCB type label (for compatibility with OS). Label indicates to OS that no space is available on this volume.

FR Is one or more filler records.

Format for CP-Owned FBA Devices

For FBA devices, the unit of formatting or allocation is the page. Fixed block device capacity when formatted for CP use is:

3310	15752 pages/spindle
3370, A1 or B1	69750 pages/spindle
3370, A2 or B2	89094 pages/spindle

Each block is 512 bytes long. The first 16 blocks (pages 0 and 1) are reserved for system use.

Block	0	1	2	3-4	5-12	13-15	16-nn
Contents	IPL REC	VOLID	VTOC	Allocation Extent Map	DMKCKP	Reserved	Pages 2-N

Each block contains:

Block	Contents
0	IPL record. Places the system into the wait state if IPL occurs before the CP nucleus is built.
1	Volume label. Same as DOS/VS1 VOL1 label. On CP system residence volumes, a field in this record contains a pointer to the page where the system directory starts. A label is automatically written when the format function is used, specifying page 0 as the starting page. The owner field of the label record contains "CP370" if there is allocation data in blocks 3 and 4.
2	Volume Table of Contents. The Format 4 and Format 5 DSCB's. These labels are written for compatibility with OS. They indicate that no space is available on this volume.
3-4	Allocation Extent Map. Each entry is 12 bytes long and describes a range of pages on the device as well as the usage of the pages (PERM, TEMP, DRCT, etc.).
5-12	Checkpoint Program. The CP module (DMKCKP).
13-15	Reserved.
16-nn	Contains the pages used by the VM/SP system. The area starts at block 16, which corresponds to page 2 on the volume.

Format/Allocate Program Input

Format/Allocate program control statements may be entered through a card reader or from the system console. All error messages for improperly specified control statements are displayed at the console.

Format/Allocate Program Card Input

Punch control statements for card input start in column 1, and each field is separated from the adjacent field by a comma. Two commas in a row cause the insertion of a default value. Three commas in a row cause the insertion of two default values.

Note: The only default values permitted are those that define the starting and ending cylinders or DASD extents. The defaults are the first and last cylinders of the volume (or pages), respectively.

Comments must be preceded by at least three blanks.

The control card entries for the Format/Allocate program must be in the following order:

- Format function

FORMAT,*devadr,devtype,volser,startadr,endadr*

- Allocate function

  ALLOCATE,*devadr,devtype,volser*
  TEMP,*startadr,endadr*
  PERM,*startadr,endadr*
  TDSK,*startadr,endadr*
  DRCT,*startadr,endadr*
  OVRD,*startadr,endadr*
  PAGE,*startadr,endadr*
  DUMP,*startadr,endadr*
  END

- Label functions

  FORMAT,*devadr,devtype,volser*,LABEL

FORMAT, ALLOCATE, and LABEL are Format/Allocate program control words and may be abbreviated to one letter.

FORMAT Control Statement: The format of the FORMAT control statement is:

FORMAT, *devadr,devtype,volser,startadr,endadr*

devadr
is a three-digit hexadecimal number that identifies the address of the device that the Format/Allocate program is to act upon. Valid device addresses under CMS are X'001' to X'5FF' for ECMODE OFF and X'001' to X'FFF' for ECMODE ON.

Note: To avoid I/O contention when formatting two device types (3375/3380) in two virtual machines, do not use consecutive even/odd pairs of device addresses (e.g., 290,291); rather, it is advisable to use consecutive odd/even pair device addresses (e.g., 291,292).

devtype
is a four-to-seven character field that defines a supported device for the Format/Allocate program.

2305-1	
2305-2	
2314	
2319	
3330	
3330-11	
3330	for a 3333 device
3340-35	
3340-70	
3340-70	for 3340-70F or 3344 devices
3350	
3330	for a 3350 device in 3330-1 compatibility mode
3330-11	for a 3350 in 3330-11 compatibility mode
3375	
3380	refers to all 3380 models
FB-512	for a 3310 or 3370 device

volser
is a one-to-six character field that represents the volume serial number of the volume you are formatting.

startadr
is the starting cylinder address on the DASD on which the format function is to be performed. For fixed-block devices, this is the starting page number. The starting address is entered as decimal digits.

endadr
is the last cylinder address on the DASD on which the format function is to be performed. For fixed-block devices, this is the ending page number. The end address is entered as decimal digits.

ALLOCATE Control Statements: The formats of the ALLOCATE control statements are:

```
ALLOCATE,devadr,devtype,volser
TEMP,startadr,endadr
PERM,startadr,endadr
TDSK,startadr,endadr
DRCT,startadr,endadr
OVRD,startadr,endadr
PAGE,startadr,endadr
DUMP,startadr,endadr
END
```

Note: You must enter the ALLOCATE and END statements to perform some type of allocation. Any allocation type control statement is optional depending on what kind of space you want on the disk. Any space not allocated on the disk will default to 'TEMP' space.

devadr
is a three-digit hexadecimal number that identifies the address of the device that the Format/Allocate program is to act upon. Valid device addresses under CMS are X'001' to X'5FF' for ECMODE OFF and X'001' to X'FFF' for ECMODE ON.

devtype
is a four-to-seven character field that defines a supported device for the Format/Allocate program:

2305-1	
2305-2	
2314	
2319	
3330	
3330-11	
3330	for a 3333 device
3340-35	
3340-70	
3340-70	for 3340-70F or 3344 devices
3350	
3330	for a 3350 device in 3330-1 compatibility mode
3330-11	for a 3350 in 3330-11 compatibility mode
3375	
3380	refers to all 3380 models
FB-512	for a 3310 or 3370 device

Note: There are 8 blocks per page, so any format or allocation specification would be:

Page no. = Block no./8.

volser
is a one-to-six character field that represents the volume serial number of the volume you are formatting.

startadr
is the starting cylinder address on the DASD on which the allocate function is to be performed. For fixed-block devices, this is the starting page number. The starting address is entered as decimal digits.

endadr
is the last cylinder address on the DASD on which the allocate function is to be performed. For fixed-block devices, this is the ending page number. The end address is entered as decimal digits.

TEMP
indicates that the following operands identify temporary storage space reserved for spooling or paging activity. DMKPGT allocates TEMP storage space for spooling.

PERM
defines an area that can contain the logout area, the CP nucleus, and space that is not used by the system but is available for use by virtual machine users (for example, for user minidisks).

TDSK
defines the temporary minidisk space available for virtual machine users during a single terminal session on the VM/SP system.

DRCT
indicates that the following space is reserved for directory files.

Note: If you wish to reallocate the system directory, it may be necessary to rerun the system directory program (DMKDIR) because the active directory pointer can be destroyed.

OVRD
indicates the space reserved for an override file. This file assigns classes to CP commands to override the IBM-defined command structure. For count-key-data devices, the system allocates only the first cylinder that it finds allocated for OVRD. For FBA devices, the system will use the first extent defined as OVRD; the minimum allocation is 2 pages.

PAGE
indicates that the following operands identify DASD storage space for preferred paging activity.

Note: Cylinders and blocks which are to be used for non-preferred paging space (spooling, overflow paging operations, and so forth) should be allocated as 'TEMP'.

DUMP
indicates that the appropriate DASD space be marked reserved for CP allocation of system dumps. This reserved area will not be used for CP paging and spooling.

Note: DUMP allocation must be contiguous since CP will look for contiguous cylinders or blocks when assigning the system dump spool file.

END
ends the ALLOCATE function. This statement causes termination of ALLOCATE functions, and a display of the allocation results. If 'END' is the only control statement specified after the 'ALLOCATE' statement, then the existing disk allocation is displayed but not updated, unless an allocation record mismatch occurs on an FBA device. In the case of a record mismatch (which may have happened if, for example, a 3370 was restored to a 3310 device) ALLOCATE followed by END provides a way of rewriting the allocation data and avoiding future disk errors. For cases where FBA devices have an

allocation record mismatch, the unallocated space will be allocated as PERM.

Specify END after you have entered all desired allocation statements.

TEMP, PERM, TDSK, PAGE, DUMP, DRCT, and OVRD are all functions of ALLOCATE. These cards can follow the ALLOCATE control statement in any sequence. Each card in turn overlays the cylinder (or allocate) table, and any space not reallocated remains the same. If an ALLOCATE function overlays the previous allotment, then the previous cylinder space allotment is truncated to the beginning of the next allotment. The allocation operation is executed after you re-IPL VM/SP. For example:

Disk Storage Allocation	First Cylinder	Last Cylinder
1st Entry PAGE	000	002
2nd Entry TEMP	003	403
3rd Entry PERM	010	050
4th Entry TDSK	040	050
5th Entry DRCT	003	004
6th Entry OVRD	005	005
7th Entry END		

The result of this disk volume allocation is:

Disk Storage Allocation	First Cylinder	Last Cylinder
PAGE	000	002
DRCT	003	004
OVRD	005	005
TEMP	006	009
PERM	010	039
TDSK	040	050
TEMP	051	403

Once an ALLOCATE control statement is encountered, all cards following it until an END card is encountered are assumed to be part of a single allocation. The Format/Allocate service functions cannot be performed on another disk volume until the END card is encountered. Any area not allocated will default to "TEMP" space. (See Figure 12 on page 134.)

Note: Reallocation of an area containing an active VM/SP directory deallocates the directory to allow a new directory to be written in the same area.

After any reallocation, the directory program (DMKDIR) must be executed to reinitialize the directory. If this is not done, CP will abend with an ABENDCPI002 when you initialize at IPL.

LABEL Control Statement: The format of the LABEL control statement is:

FORMAT,*devadr*,*devtype*,*volser*,**LABEL**

devadr
is a three-digit hexadecimal number that identifies the address of the device that the Format/Allocate program is to act upon. Valid device addresses under CMS are X'001' to X'5FF' for ECMODE OFF and X'001' to X'FFF' for ECMODE ON.

devtype
is a four-to-seven character field that defines a supported device for the Format/Allocate program:

2305-1	
2305-2	
2314	
2319	
3330	
3330-11	
3330	for a 3333 device
3340-35	
3340-70	
3340-70	for 3340-70F or 3344 devices
3350	
3330	for a 3350 device in 3330-1 compatibility mode
3330-11	for a 3350 in 3330-11 compatibility mode
3375	
3380	refers to all 3380 models
FB-512	for a 3310 or 3370 device

volser
is a one-to-six character field that represents the volume serial number.

LABEL
is a keyword designating the label function of the Format/Allocate program.

Note: Format of cylinder 0 is a prerequisite for the label only function.

If the volume is to be used for paging, spooling, dump, directory, override, or temp space, 'CP370' is required in record 3 for CKD or block 1 for FB-512. The label option does not write it out; the FORMAT function must be used to format cylinder 0 for CKD or page 0 for FB-512 to insure the inclusion of the 'CP370'.

Examples:

FORMAT

```
FORMAT,232,3330,MYDISK,000,006
FORMAT,232,3330,MYDISK,,,
```

```
FORMAT,232,3330,MYDISK,,00
FORMAT,232,3330,MYDISK,001,,

ALLOCATE

ALLOCATE,232,3330,MYDISK
PAGE,000,020
TEMP,021,150
PERM,055,060
TDSK,100,108
OVRD,109,109
DRCT,110,120
DUMP,121,150
END

LABEL

F,232,3330,MYDISK,label
```

Format/Allocate Console Input

The Format/Allocate program can be controlled by control statements entered into the real or virtual console instead of by a deck of cards containing control statements. If the program finds no control statements at the card reader, it issues a prompting message to the console. The proper response causes the prompting message for the next operand to appear until the Format, Allocate, or Label function is completely defined; then the Format/Allocate program is executed. After execution, the prompting begins again until all DASD allocation requirements are fulfilled.

Follow these steps to run the Format/Allocate program from the console:

1. Load the card reader with a loader, followed by the Format/Allocate deck. Under CMS, the loader and the deck are contained in one file, named IPL FMT S, on the CMS system disk.

2. IPL the card reader.

3. Respond to the prompts.

4. In a virtual machine environment, you must re-IPL CMS to exit from the Format/Allocate program.

Following are examples of Format/Allocate program execution under CP control. Figure 11 on page 134 is an example of the label operation. Figure 12 on page 134 is an example of the allocate operation. Figure 13 on page 135 is an example of the allocate overlap operation. All responses are entered after the colon, as shown, except on 3270-type terminals. After a function is complete, the program returns and again issues the ENTER 'FORMAT' OR 'ALLOCATE': statement.

```
  VM/SP FORMAT/ALLOCATE PROGRAM

  ENTER "FORMAT" OR "ALLOCATE":
format

  FORMAT FUNCTION SELECTED
  ENTER DEVICE ADDRESS (CUU):
131

  ENTER DEVICE TYPE:
3380
  ENTER START CYLINDER (XXX OR XXXX) OR "LABEL":
L

  ENTER DEVICE LABEL:
cpdsk2
```

Figure 11. Using the Format Program Label Function

```
  ENTER "FORMAT" OR "ALLOCATE":
allocate
  ALLOCATE FUNCTION SELECTED
  ENTER DEVICE ADDRESS (CUU):
131
  ENTER DEVICE TYPE:
3380
  ENTER DEVICE LABEL:
cpdsk2
  ENTER ALLOCATION DATA FOR VOLUME CPDSK2
  TYPE CYL CYL
  .... ... ...
  drct 0000 0001
  ovrd 0002 0002
  perm 0004 0008
  page 0009 0070
  dump 0071 0099
  tdsk 0100 0150
  end
  ALLOCATION RESULTS
  DRCT 0000 0001
  OVRD 0002 0002
  TEMP 0003 0003
  PERM 0004 0008
  PAGE 0009 0070
  DUMP 0071 0099
  TDSK 0100 0150
  TEMP 0151 0884
  DEVICE 131 VOLUME CPDSK2 ALLOCATION ENDED
```

Figure 12. Using the Format Program Allocate Function

```
  ENTER "FORMAT" OR "ALLOCATE":
 allocate
  ALLOCATE FUNCTION SELECTED
  ENTER DEVICE ADDRESS (CUU):
 131
  ENTER DEVICE TYPE:
 3350
  ENTER DEVICE LABEL:
cpdsk2
  ENTER ALLOCATION DATA FOR VOLUME CPDSK2
  TYPE CYL CYL
  .... ... ...
  perm 004 004
  temp 000 010
  tdsk 000 010
  perm 010 202
  drct 000 004
  ovrd 005 005
  end
  ALLOCATION RESULTS
  DRCT 000 004
  OVRD 005 005
  TDSK 006 009
  PERM 010 202
  TEMP 203 554
  DEVICE 131 VOLUME CPDSK2 ALLOCATION ENDED
```

Figure 13. Using the Format Program Allocate Overlap Function

Figure 14 shows the label function for FBA devices. Figure 15 on page 136 shows the allocate function for FBA devices. Figure 16 on page 137 shows the allocate overlap function for FBA devices.

```
  VM/SP FORMAT/ALLOCATE PROGRAM

  ENTER "FORMAT" OR "ALLOCATE":
 format
  FORMAT FUNCTION SELECTED
  ENTER DEVICE ADDRESS (CUU):
 131
  ENTER DEVICE TYPE:
FB-512
  ENTER START PAGE NUMBER OR "LABEL":
label
  ENTER DEVICE LABEL:
cpdsk2
```

Figure 14. Using the Format Program Label Function for FBA Devices

```
  ENTER "FORMAT" OR "ALLOCATE":
allocate
  ALLOCATE FUNCTION SELECTED
  ENTER DEVICE ADDRESS (CUU):
131
  ENTER DEVICE TYPE:
FB-512
  ENTER DEVICE LABEL:
cpdsk2
  ENTER ALLOCATION DATA FOR VOLUME CPDSK2
  TYPE PAGE PAGE
  .... ... ...
  perm   2 1000
  temp 1001 5000
  drct 5001 5100
  ovrd 5101 5108
  tdsk 5109 9000
  end
  ALLOCATION RESULTS
  PERM     2  1000
  TEMP 1001  5000
  DRCT 5001  5100
  OVRD 5101  5108
  TDSK 5109  9000
  TEMP 9001 15751
  DEVICE 131 VOLUME CPDSK2 ALLOCATION ENDED
```

Figure 15. Using the Format Program Allocate Overlap Function for FBA Devices. Specifications for 3310 are shown

```
  ENTER "FORMAT" OR "ALLOCATE":
allocate
  ALLOCATE FUNCTION SELECTED
  ENTER DEVICE ADDRESS (CUU):
131
  ENTER DEVICE TYPE:
FB-512
  ENTER DEVICE LABEL:
cpdsk2
  ENTER ALLOCATION DATA FOR VOLUME CPDSK2
  TYPE PAGE PAGE
  .... ..... .....
  temp   2  500
  page 9500 11000
  end
  ALLOCATION RESULTS
  TEMP      2   500
  PERM    501  1000
  TEMP   1001  5000
  DRCT   5001  5100
  OVRD   5101  5108
  TDSK   5109  9000
  TEMP   9001  9499
  PAGE   9500 11000
  TEMP  11001 15751
  DEVICE 131 VOLUME CPDSK2 ALLOCATION ENDED
 ------------------------------------------------------
  ENTER "FORMAT" OR "ALLOCATE":
a
  ALLOCATE FUNCTION SELECTED
  ENTER DEVICE ADDRESS (CUU):
131
  ENTER DEVICE TYPE:
FB-512
  ENTER DEVICE LABEL:
cpdsk2
  ENTER ALLOCATION DATA FOR VOLUME CPDSK2
  TYPE PAGE PAGE
  .... ..... .....
  perm   0 1000
  LOWEST ALLOCATABLE PAGE IS PAGE 2 -- RESPECIFY
  perm   2 17345
  HIGHEST ALLOCATABLE PAGE IS PAGE 15751 -- RESPECIFY
  perm   2 1000
  end
  ALLOCATION RESULTS
  PERM      2  1000
  TEMP   1001  5000
  DRCT   5001  5100
  OVRD   5101  5108
  TDSK   5109  9000
  TEMP   9001  9499
  PAGE   9500 11000
  TEMP  11001 15751
  DEVICE 131 VOLUME CPDSK2 ALLOCATION ENDED
```

Figure 16. Using the Format Program Allocate Overlap Function for FBA Devices. Specifications for 3310 are shown

Note that before the ALLOCATE function was invoked, page 0 was formatted and labeled CPDSK2. The area associated with the first three 4096-byte blocks on page 0 are not used for spooling but contain system information (page allocation map, label, and so on).

These CP-formatted volumes can be made usable by CP in one of two ways:

- They may be attached to the system by the VM/SP operator.
- Their volume serial numbers may appear in the SYSOWN macro in the DMKSYS module. The CP system residence volume's serial number must appear in the SYSOWN macro.

DASD Dump Restore Service Program (DDR)

The DASD Dump Restore (DDR) program dumps, restores, copies, or prints data from a direct access storage device, such as a whole volume of real DASD or a VM/SP user minidisk. The DDR program may run as a stand-alone program, or under CMS via the DDR command.

The DDR program has five functions:

- Dumping part or all of the data from DASD to tape.
- Transferring data from tapes created by the DDR dump function to a direct access device. The direct access device must be of the same type as that which originally contained the data.
- Copying data from one device to another of the same type. Data may be reordered by cylinder (or block) when copied from disk to disk. In order to copy one tape to another, the original tape must have been created by the DDR DUMP function.
- Printing selected parts of DASD and tape records in hexadecimal and EBCDIC on the virtual printer.
- Displaying selected parts of DASD and tape records in hexadecimal and EBCDIC on the terminal.

To generate the VM/SP starter system from the distribution tape, the stand-alone RESTORE function must be used.

Invoking DDR Under CMS

The format of the DDR command is:

DDR	$\left[fn\ ft\ \left[\begin{matrix}fm\\ *\end{matrix}\right]\right]$

fn ft [*fm* | *]

is the identification of the file containing the control statements for the DDR program. If no file identification is provided, the DDR program prompts for control statements from the console. The filemode defaults to an asterisk (*) if a value is not provided.

DDR handles two logical line editing symbols:

- The logical character delete symbol (@), which allows deletion of one or more of the previously entered characters.
- The logical line delete symbol (¢), which deletes the entire previous physical line.

Most often, the default values for these two symbols are defined for each virtual machine at system generation time. When you use the CP TERMINAL command to redefine the CHARDEL and LINEDEL characters, the re-definitions have no affect on DDR line editing. DDR continues to recognize "@" as valid CHARDEL and "¢" as valid LINEDEL symbols.

Comments:

1. In a virtual machine, DDR informational messages are directed by default to the CMS virtual printer, X'00E', unless the SYSPRINT CONS option is specified.
2. When DDR is invoked in CMS, the I/O operation is performed by CP, which has built-in error recovery facilities.

Invoking DDR as a Stand-alone Program

To use DDR as a stand-alone program, the operator should IPL it from a real or virtual IPL device as he would any other stand-alone program. Then indicate where the DDR program is to obtain its control statements by responding to prompting messages at the console.

Comments:

1. When DDR is run as a stand-alone program, it has only the most elementary error recovery support.
2. When running stand-alone, DDR will search for a console at address 009 or 01F. If these consoles are not operational, the program will enter a wait state, waiting for an interrupt to identify the console. If any

nonconsole type device is physically connected to address 009 or 01F, it must be made nonoperational or the results will be unpredictable.

3. The stand-alone DDR utility will not support cylinder faults for MSS virtual volumes when performing the DUMP or COPY functions.

4. It may be necessary to disable channel-to-channel devices in order for the DDR utility to run as a stand-alone program.

5. The DDR utility does not assign 3480 tape drives to itself. So, when a 3480 is in use during stand-alone DDR processing, do not IPL any other processor that has a path to the device. This is because the other processor might steal the 3480 drive by assigning it to itself. This would immediately stop DDR's access to the device. For the same reason, do not vary the 3480 device online to any other processor until DDR processing completes.

6. When you are doing a stand-alone DDR and you have a failure on a tape drive, you can re-IPL the DDR tape and restart at the failed tape without having to go back to the beginning.

DDR 8809 Data Streaming Support

The 8809 operates in either start/stop (12.5 inches per second) or streaming mode (100 inches per second). In streaming mode, the 8809 maintains the tape velocity through the interblock gap, anticipating the next command. This means that the tape motion continues without losing time either starting or stopping if the next command is received in time.

If a command doesn't arrive on time, the tape proceeds past the next record, backs up, and then performs the next command. This process of stopping and re-positioning is called backhitching, and takes more than one second to complete.

So DDR can use 8809 'Data Streaming', the following conditions must be met:

- DDR is Operating in a stand-alone real mode.
- The DDR function is either DUMP or RESTORE.
- The DASD is an FB-512 (FBA) device.
- The input and output devices are on different channels.

DDR Control Statements

DDR control statements describe the intended processing and the needed I/O devices. I/O definition statements must be specified first.

All control statements may be entered from either the console or the card reader. Only columns 1 to 71 are inspected by the program. All data after the last operand in a statement is ignored. An output tape must have the DASD cylinder header records in ascending sequence; therefore, the extents must be entered in sequence by cylinder or by extent. Only one type of function—dump, restore, or copy—may be performed in one execution, but up to 20 statements describing DASD cylinder extents may be entered.

The function statements are delimited by (preceded by) an INPUT and OUTPUT statement. If additional functions are to be performed, the sequence of control cards must be repeated. If you do not use INPUT or OUTPUT control statements to separate the functions you specify when the input is read from a card reader or CMS file, an error message is displayed. However, the remainder of the input stream will be checked for proper syntax, but no further DDR operations will be performed. Only those I/O devices defined by the INPUT statement and the OUTPUT statement must be redefined in subsequent steps. The SYSPRINT I/O definition remains the same.

To return to CMS, enter a null line (carriage return) in response to the prompting message (ENTER:). To return directly to CP, key in #CP.

The PRINT and TYPE statements work differently than other DDR control statements in that they operate on only one data extent at a time and it is not necessary to respecify the INPUT statement ahead of each PRINT or TYPE statement. If the input is from a tape created by the dump function, the tape must be positioned at the header record for each step. The PRINT and TYPE statements have an implied output of either the console (TYPE) or system printer (PRINT), so no OUTPUT statement is required.

The I/O definition statements describe the tape devices, DASD, and printers used while executing the DASD Dump Restore program.

INPUT/OUTPUT Control Statement

An INPUT or OUTPUT statement describes each tape unit and DASD used. The format of the INPUT/OUTPUT statement is:

INput **OUTput**	*cuu type* [*volser* / *altape* / **SCRATCH**] [(Options...)] Options: [**SKip** *nn* / **SKip** 0] [**MOde 6250** / **MOde 1600** / **MOde 800** / **MOde 38K**] [**REWIND** / **UNload** / **LEave**] **COmpact**

INPUT
indicates that the device described is an input device.

OUTPUT
indicates that the device described is an output device.

Note: If the output device is a DASD and DDR is running under CMS, the device is released using the CMS RELEASE command function and DDR processing continues.

cuu
is the unit address of the device.

type
is the one of the following device types:

2305-1	2400	3330-11	3380	3430
2305-2	2401	3340-35	3410	3480
2311	2415	3340-70	3411	8809
2314	2420	3350	3420	FB-512
2319	3330	3375	3422	

There is no 7-track support for any tape devices.

Specify a 3340-70F as a 3340-70, and a 3333 as a 3330. Specify a 3350 that is in 3330-1 or 3330-11 compatibility mode as a 3330 or 3330-11. Specify a 3344 as a 3340-70, and specify 3350 for a 3350 operating in native mode (as opposed to compatibility mode). Specify any 3380 model as a 3380. Both 3310 and 3370 are denoted by specifying FB-512 or FB.

The DASD Dump Restore (DDR) program, executing in a virtual machine, uses I/O DIAGNOSE 20 to perform I/O operations on tape and direct access storage devices.

DDR under CMS requires that the device type entered agree with the device type of the real device as recognized by VM/SP. (DDR uses DIAGNOSE X'24' If there is a conflict with device types, the following message is issued:

```
DMKDDR708E  INVALID INPUT OR OUTPUT DEFINITION
```

However, if DDR executes in a virtual machine, DDR uses DIAGNOSE 24 to determine the real device type. If the device types do not agree, an error message is issued. The speed setting for 8809 tape drives is not under your control. When DDR is running as a command under CMS, the 8809 is supported only in start/stop mode. If DDR is run stand-alone, DDR attempts to run the 8809 in high-speed mode. In this mode, the data transfer time is reduced. However, this does not necessarily mean that the time for a DDR job is reduced; job duration depends on many factors such as processor and device contention.

volser
is the volume serial number of a DASD. If the keyword "SCRATCH" is specified instead of the volume serial number, no label verification is performed.

altape
is the address of an alternate tape drive.

If multiple reels of tape are required and "altape" is not specified, DDR types one of the the following at the end of the reel:

```
END OF VOLUME CYL xxxx HD xx, MOUNT NEXT TAPE
```

or

```
END OF VOLUME BLOCK xxxxxx, MOUNT NEXT TAPE
```

After the new tape is mounted, DDR continues automatically.

SKip $\left[\begin{matrix}nn \\ \underline{0}\end{matrix}\right]$

forward spaces *nn* files on the tape, where *nn* is any number up to 255. The SKIP option is reset to zero after the tape has been positioned.

Use the SKIP option if there is more than one dump on the tape or if other files precede the dump.

$$\left[\begin{matrix}\textbf{MOde 6250} \\ \underline{\textbf{MOde 1600}} \\ \textbf{MOde 800} \\ \textbf{MOde 38K}\end{matrix}\right]$$

Modes 6250, 1600, 800, or 38K causes all output tapes that are opened for the first time and at the load point to be written or read in the specified density. All subsequent tapes mounted are also set to the specified density. If no mode option is specified, then no mode set is performed and the density setting remains as it previously was unless

the tape is positioned at the load point. When this occurs, the density setting resets to 1600 (the default value).

MODE also specifies the recording density for the 3480 magnetic tape subsystem. Since the 3480 records only at a density of 38K, the default for the MODE option of the 3480 is 38K.

Notes:

1. *If a user specifies a density mode that the tape cannot handle, the control unit may not return an error condition. Instead, the mode setting is ignored and the default control unit setting is used.*

2. *800 BPI is an invalid mode option for the 3430 tape drive. An error message is issued if a user specifies 800 as a mode option for a 3430.*

3. *Only modes 1600 and 6250 can be specified for the 3422 magnetic tape subsystem. An invalid mode will generate an error message.*

REWIND
rewinds the tape at the end of a function.

UNLOAD
rewinds and unloads the tape at the end of a function.

LEAVE
leaves the tape positioned at the end of the file at the end of a function.

COMPACT
causes the output tape to be in a compact format, which uses less tape space than standard format. DDR stores data in a compact format by compressing strings of duplicate data into a smaller amount of space and reducing the amount of space necessary to represent the characters in the data. This option is valid only on the OUTPUT control statement for the DUMP functions.

You can use tapes in the compact format as input to the RESTORE, COPY, PRINT, and TYPE functions without any changes. For more information, refer to the following sections about function statements.

Comments:

1. When the wrong input tape is mounted, the message

   ```
   DMKDDR709E  WRONG INPUT TAPE MOUNTED
   ```

   is displayed and the tape will rewind and unload regardless of options REWIND, UNLOAD, or LEAVE being specified.

2. If DDR is executed from CMS, failure to attach the tape drive or the disk device (or both) to your virtual machine prior to invoking the input/output statement causes the following response to be displayed:

```
DMKDDR708E  INVALID INPUT OR OUTPUT DEFINITION
```

SYSPRINT Control Statement

The SYSPRINT control statement describes the device that output is to be sent to. If the SYSPRINT CONS option is specified, the output is directed to the console for both the CMS environment and the DDR virtual machine.

In the CMS environment, all output is directed (by default) to 00E, unless the SYSPRINT CONS option is specified. Any SYSPRINT cuu option specification is ignored.

In the DDR virtual machine, the output is directed to the output device specified by the SYSPRINT cuu option. If the SYSPRINT CONS option is specified, all output is directed to the console. If no options are specified, the output is directed (by default) to 00E.

SYsprint	[*cuu* | **CONS** | 00E]

cuu
specifies the unit address of the device.

CONS
specifies the console as the output device.

Function Statements

The function statements tell the DDR program what action to perform. The function commands also describe the extents to be dumped, copied, or restored. The format of the DUMP/COPY/RESTORE control statement is:

DUmp	**[FTr]** [*block*1 | *cyl*1 **[TO]** [*block*2 | *cyl*2 **[Reorder]** **[To]** [*block*3 | *cyl*3]] | **CPvol** | **ALL** | **NUcleus**]

COpy **REstore**	[*block1* / *cyl1* [TO] [*block2* / *cyl2* [**Reorder**] [To] [*block3* / *cyl3*]]] **CPvol** **ALL** **NUcleus**

DUMP

requests the program to move data from a direct access volume onto a magnetic tape or tapes. After the system dumps the data, if you specified the COMPACT option on the OUTPUT control statement, you will receive these messages:

```
BYTES IN _____________BYTES OUT _____________
TRACKS NOT COMPACTED ON TAPE _____________________
BLOCKS NOT COMPACTED ON TAPE_____________________
```

The format of the tape depends on the type of the direct access volume. **Comments:**

1. The FTr operand is valid only with the DUMP control statement.

2. If you specify dump with the COmpact option on the OUTPUT card, the system dumps the data in an FTR format.

FTr

requests an output tape format of variable unblocked records. The size of the records and the number of records per track written to the tape depend on the density of the tape. The FTr operand is valid only with the DUMP control statement.

The option can be used for those devices supporting the full-track-read (FTR) feature (3330, 3340, 3344, 3350, 3375, and 3380) and for FBA devices (FTR is the default for 3375 and 3380 and, therefore, need not be specified).

If FTr is specified on the DUMP control statement for a count-key-data DASD but the control unit does not support the feature, a message is written and the operation proceeds with data written in the old format.

For count-key-data direct access volumes, the data is moved cylinder by cylinder.

The format of the resulting tape is:

Non-FTR format:

Record 1
A volume header record, consisting of data describing the volumes.

Record 2
A track header record, consisting of a list of count fields to restore the track, and the number of data records written on tape. After the last count field, the record contains key and data records to fill the 4K buffer.

Record 3
Track data records, consisting of key and data records packed into 4K blocks, with the last record truncated.

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

FTR format:

Record 1
The same as described for the non-FTR format.

Record 2
A track header record, consisting of fields containing the length of the track, the density of the tape, and the number of count fields in the track followed by the track contents to fill the record, making it the same length as Record 3.

Record 3
Track data records, consisting of count-key-data records in 8K, 12K, or 48K blocks for 800, 1600, or 6250 BPI respectively, or 48K blocks for the 3480 Tape Subsystem. The last block, in all cases, is a short block.

Record 4
The same as described for the non-FTR format.

For FBA devices, the data is moved in sets of blocks. Each set contains 95 (or less for the last set) blocks of data. Any number of blocks can be moved with one DUMP statement. The format of the resulting tape depends on whether the FTR option is used or not, and/or the density of the output tape.

Record 1
The same as described for the non-FTR format of CKD devices.

Record 2
A data header record. This consists of data that describes the set of blocks that follow (such as block numbers and the number of tape records required to hold these FB-512 blocks). Following the control data are the actual FB-512 blocks filling out the tape record.

Record 3
FB-512 data records. These contain the rest of the blocks making up the set.

Record 4
The same as described for the non-FTR format of CKD devices.

In non-FTR format the record length of record 2 and record 3 is 4K bytes.

For FTR or compact formatted tapes, record length is 8K, 12K, or 48K blocks for 800, 1600, or 6250 BPI respectively, or 48K blocks for the 3480 Tape Subsystem. The last block, in all cases, may be a short one.

COPY
requests the program to copy data from one device to another device of the same or equivalent type. Data may be recorded on a cylinder or block basis from input device to output device. A tape-to-tape copy can be accomplished only with data dumped by this program.

You may use a tape in compact format as input. For a tape-to-tape copy, the output tape will be in the same format (compact or standard) as the input tape. The COMPACT option on the OUTPUT control statement is not valid for the COPY function. If it is specified, the system displays the following message:

```
COMPACT OPTION IGNORED FOR COPY OPERATIONS
```

Note: You cannot copy between FBA and count-key-data devices.

RESTORE
requests the program to return data that has been dumped by this program. Data can be restored only to a DASD volume of the same or equivalent device type from which it was dumped. It is possible to dump from a real disk and restore to a minidisk as long as the device types are the same.

You can use a tape in compact format as input. DDR checks if the input is in compact format, and expands the data back to standard format, if needed. You do not need to specify anything to the program about the tape format. After the system restores the data, you will receive the following message:

```
BYTES RESTORED _______
```

cyl1 **[TO]** **[** *cyl2* **[REORDER]** **[TO]** **[***cyl3***]** **]**
Only those cylinders specified are moved, starting with the first track of the first cylinder (cyl1), and ending with the last track of the second cylinder (cyl2). The REORDER operand causes the output to be reordered, that is, moved to different cylinders, starting at the specified cylinder (cyl3) or at the starting cylinder (cyl1) if cyl3 is not specified. The REORDER operand must not be specified unless specified limits are defined for the operation; the starting and, if

required, ending cylinders (cyl1 and cyl2) must be specified. If the input device cylinder extents exceed the number of cylinders specified on the output device, an error message results.

Note: The DDR program accepts either a 3-digit or 4-digit cylinder specification.

*block1***[To] [** *block2* **[REORDER] [TO] [***block3***]]**
only the specified blocks are moved, starting with the first block, up to and including the last block. The REORDER operand causes the data to be moved to a different DASD location. The REORDER operand must not be specified unless limits are defined for the operation. If the input block extents exceed the capacity of the output device, an error message results.

CAUTION
Use the REORDER operand to move minidisks to new locations, not to re-locate non-minidisk cylinders. To understand the difference, consider a 10-cylinder minidisk. Its cylinders are numbered 0-9 and the count fields of its records refer to cylinders 0-9. If the minidisk contains location-dependent data, then references to cylinders 0-9 will be hidden within that data. When you use REORDER to move the minidisk to a new real location, the minidisk cylinders are still regarded as being cylinders 0-9, and you do not need to change the cylinder numbers in the count field of the records. On the other hand, when moving non-minidisk cylinders, you would want the count fields of the moved records to reflect the new cylinder addresses, but REORDER keeps the original cylinder numbers in the count fields.

CPVOL
specifies that cylinder 0 for count-key-data devices, or blocks 0-16 for FBA devices including all active directory, override space, and permanent disk space are to be copied, dumped, or restored. This indicates that both source and target disk must be in CP format, that is, the CP Format/Allocate program must have formatted them. This includes space allocated as DRCT, PERM, and OVRD. When a tape-input function specifies CPVOL, the system restores or copies all the data.

ALL
specifies that the operation is to be performed on the entire DASD volume, either cylinders or blocks. This operation is not valid for alternate track cylinder assignments on some devices. (See "Restrictions" on page 150.)

Note: The occurrence of message DMKDDR705E (issued upon completion of the copy restore or dump operation) indicates that an attempt was made to copy, restore, or dump the contents of cylinders beyond the extents of the designated minidisk.

When you do a COPY ALL function of the DDR program from an existing 3380 DASD to a 3380 AE4/BE4, and vice versa, you must reallocate all cylinders to reflect the change in device capacity. (Use the ALLOCATE function of the FORMAT/ALLOCATE program.)

When DDR is run under CMS and you specify the ALL option for a minidisk that resides on a CKD DASD volume, you will get message DMKDDR705E when the operation completes.

NUCLEUS
specifies that record 2 on cylinder 0, track 0 and the nucleus cylinders for count-key-data devices are dumped, copied, or restored. These are blocks 5-12 for FBA devices.

Note: To do a 'DDR RESTORE NUC', the tape must be created by 'DDR DUMP NUC'. Otherwise, the system would issue message DMKDDR723E.

Restrictions

1. Each track processed by this utility must have a valid home address on it containing the real cylinder and track location. Even when restoring and copying data to a track it must have a pre-existing home address on it.

2. Each track on an input DASD must have a valid record zero on it, with no more than eight bytes in the key and data fields of the record. Each track on an output DASD must also have a valid record zero on it unless that device is a 2314, 2319, 2305-1, or 2305-2.

3. Flagged tracks are treated just as any other track for all 2314, 2319, and 2305 devices. That is, no attempt is made to substitute the alternate track data when a defective primary track is read. In addition, tracks are not inspected to determine whether they were previously flagged when written. Therefore, volumes containing flagged tracks should be restored to the same cylinders of the volume from which they were dumped. The message DMKDDR715E occurs each time a defective track is dumped, copied or restored, and the operation continues. When ALL is specified for these device types, both the primary cylinders and the high-order cylinders normally reserved for alternate tracks are dumped, copied, or restored.

4. Flagged tracks on 3330, 3340, 3350, 3375, and 3380 devices are handled so that data is transferred to or from the assigned alternate track in place of the defective track. (For 3330, 3375, 3380 and 3350 this is accomplished automatically by the hardware of the control unit while for the 3340 and 3344 it is accomplished through software.)

Note: Alternate track recovery for overflow records is not provided by VM/SP for 3340s and 3344s.

The tape created by dumping one of these types of DASD will appear as if it had been dumped from a defect-free device and the tape can be restored to any device of the same type, even though that device might not have the same tracks flagged defective as the original device had. (The COPY function works this way also.) If a track is flagged as defective, but has no alternate assigned, a warning message is issued and the only data transferred is the home address record and record zero. When ALL is specified for these device types, only the primary cylinders are processed; the cylinders reserved for alternate tracks are not processed except that an assigned alternate track is processed whenever the corresponding defective track is processed. However, by specifying the cylinder range explicitly (cyl1 to cyl2 format), all cylinders, including cylinders in the alternate track area, can be dumped or copied from. But these same cylinders cannot be restored to or copied to explicitly. It is intended that explicitly dumped cylinders in the alternate track area will be restored to another area via the REORDER operand. The only reason the explicit dumping and copying of cylinders from the alternate track area is allowed at all is to facilitate conversion of 3340 and 3344 disks that were written using early releases of VM/SP.

5. You should move any minidisk that extended into the alternate track cylinder to either another area of the disk or to another disk. To do this, use the REORDER option to copy or restore a minidisk to another area.

6. 3375 or 3380 DASD default to full track read mode. For other devices that support full track read processing, you must specify the option. Otherwise, the tape will be produced in the current non-FTR DDR format of 4096 blocks. The 3330/3340 DASD can only take advantage of the full track read feature only when the 3830 or 3880 has microcode supporting either 3344 or 3350.

7. The system cannot use tapes created by the DDR DUMP function, which are in compact format, as input to earlier levels of DDR.

8. If you specify the COMPACT option, do not specify the FTR option with it. The full-track-read feature will be utilized by COMPACT if it is available.

9. The 8809 tape drive may not operate efficiently in streaming mode while DDR is processing data in the compact format.

Some Examples:

Example 1:

```
INPUT 191 3330 SYSRES
OUTPUT 180 3420 181 (MODE 1600
SYSPRINT 00F
DUMP FTR CPVOL
INPUT 130 3330 MINI01
DUMP 1 TO 50 REORDER 51
60 70 101
```

This example sets the tape density to 1600 BPI, then dumps all pertinent data (for CPVOL, this includes label/allocation data, directory space, permanent space, and override space) from the volume labeled SYSRES onto the tape that is mounted on unit 180. The data is dumped in a full-track-read format, so, in this case, the records will be in 8K blocks. If the program runs out of space on the first tape, it continues dumping onto the alternate device (181). A map of the dumped cylinders is printed on unit 00F while the program is dumping.

When the first function is complete, the volume labeled MINI01 is dumped onto the tape. Its cylinder header records are labeled 51 to 100. A map of the dumped cylinders is printed on unit 00F. Next, cylinders 60 to 70 are dumped and labeled 101 to 111. This extent is added to the cylinder map on unit 00F. When the DDR processing is complete, the tapes are unloaded and the program stops.

Example 2:

```
INPUT 150 3375 SYSTEST
OUTPUT 181 3480 182 (MODE 38K LEAVE COMPACT
DUMP ALL
```

This example sets the tape density to 38K BPI (which is required for a 3480 tape device). After the data is dumped, the tape will not rewind or unload due to the LEAVE option. DDR will compact all data dumped before writing it out to tape. When the first tape is full, DDR will begin dumping to the alternate tape (182). DDR will dump all cylinders on the 3375 to the tapes.

Example 3:

```
INPUT 101 3380
OUTPUT 202 3380
COPY NUCLEUS
```

This example copies the following from device 101 to the same location on device 202:

- Record 2 on cylinder 0, track 0
- The nucleus cylinders.

In this case, the input and output device types must be the same. All FBA devices are considered to be of the same device type: FB-512.

Example 4:

```
INPUT 182 3420 (MODE 1600 REWIND
OUTPUT 250 FB-512 SYSTEST
RESTORE 10 TO 20 REORDER TO 30
```

This example restores data previously dumped from an FBA device to another FBA device with address 250. Blocks 10 through 20 on the tape are restored to the DASD starting at block 30 to block 40.

In any case, if you are defining cylinder extents from the console, you only need to enter DUMP, COPY or RESTORE on the command line. The system displays the following:

```
ENTER CYLINDER EXTENTS
ENTER:
```

For any extent after the first extent, the message

```
ENTER NEXT EXTENT OR NULL LINE
ENTER:
```

is displayed. The user may then enter additional extents to be dumped, restored, or copied. A null line causes the job step to start.

Comments:

1. When a cylinder map is printed on the virtual printer (00F as in the previous example) a heading precedes the map information. Module DMKDDR controls the disk, time and zone printed in the heading. Your installation must apply a local modification to DMKDDR to ensure that local time, rather than GMT (Greenwich Mean Time), is printed in the heading.

2. Attempts to restore cylinders or blocks beyond the capacity that had been recorded on the tape produces a successful EOJ, but the printout only indicates the last cylinder or block found on the tape.

3. An end-of-file mark is written to the tape following each DUMP function. Multiple extents can be written within one DUMP function. Multiple DUMP functions can be performed within one DDR session. If an extent table is built within one DUMP function, then a tape mark is written after all of these extents have been dumped, to indicate end-of-file.

PRINT/TYPE Function Statement

Use the PRINT and TYPE function statement to print or type (display) a hexadecimal and EBCDIC translation of each record specified. The first of a group of PRINT or TYPE statements must be preceded by an INPUT statement defining either a direct access device or a tape. The output is directed to the system console for the TYPE function, or to the SYSPRINT device for the PRINT function. (This does not cause redefinition of the output unit definition.) PRINT and TYPE may be used to display the

contents of any track including those in the alternate track cylinders. For 3330, 3340, and 3350 devices, the following is displayed when alternate tracks are involved:

- When displaying a defective track that has a properly assigned alternate, the home address record displayed is taken from the defective track while record zero and all other records are taken from the alternate. The "defective" flag, visible in the displayed home address, is the only hint that this is not a normal track.
- When displaying a flagged defective track which does not have a proper alternate, only the home address record and record zero are displayed, and they are both taken from the defective track.
- When displaying an alternate track explicitly, all data displayed is from that track.

The input may be any valid device as described under "INPUT/OUTPUT Control Statement" on page 141. If the input device is tape, it must be a tape created by the DDR service program. You do not need to specify whether the tape is in compact or standard format, as the input format does not affect the printed or displayed output.

The format of the PRINT/TYPE control statement is:

PRint **TYpe**	*cyl1* [*hh1* [*rr1*]]	[**To** *cyl2* [*hh2* [*rr2*]]]	[(Options...)]]
	block1	[**To** *block2*]	[(Options...)]
	Options **[Hex] [Graphic] [Count]**		

cyl1
is the starting cylinder.

hh1
is the starting track. If present, it must follow the cyl1 operand. The default is track zero.

rr1
is the starting record. If present, it must follow the hh1 operand. The default is home address and record zero.

TO *cyl2*
is the ending cylinder. If more than one cylinder is to be printed or typed, "TO cyl2" must be specified.

hh2
is the ending track. If present, it must follow the cyl2 operand. The default is the last track on the ending cylinder.

rr2
is the record ID of the last record to print. The default is the last record on the ending track.

block1
is the starting FB-512 block number.

TO *block2*
is the ending block number. If more than one block is to be printed or typed, "To block2" must be specified.

Options:

HEX
prints or displays a hexadecimal representation of each record specified.

GRAPHIC
prints or displays an EBCDIC translation of each record specified.

COUNT
prints or displays only the count field for each record specified. The COUNT option is ignored for FBA data.

Usage:

If the TYPE statement follows the occurrence of error message DMKDDR705E and specifies the same cylinder, track, and record extents indicated in the error message, the contents of the printed record must be interpreted in the context of the I/O error information given in the initial message.

Examples:

PRINT 0 TO 3
Prints all of the records from cylinders 0, 1, 2, and 3.

PRINT 0 1 3
Prints only one record, from cylinder 0, track 1, record 3. Count-key-data devices print only one record; from cylinder 0, track 1, record 3; FBA devices print blocks 0 to 3.

PRINT 1 10 3 TO 1 15 4
Prints all records starting with cylinder 1, track 10, record 3, and ending with cylinder 1, track 15, record 4.

The example in Figure 17 on page 159 shows the information displayed at the console (TYPE function) or system printer (PRINT function) by the DDR program. The listing is annotated to describe some of the data fields.

The printed output for FBA data is also provided by the DDR program. The program first prints a heading that lists the block number and then prints the 512 bytes of data in the block.

Responses

Message (DMKDDR711R) indicates that the volume serial number read from the device at cuu is not the same as that specified on the INPUT or OUTPUT control card:

```
DMKDDR711R VOLID READ IS  volid2 NOT volid1
           DO YOU WISH TO CONTINUE?  RESPOND YES NO OR
           REREAD:
```

volid2
: is the volume serial number from the VOL1 label on the DASD unit.

volid1
: is the volume serial number from the INPUT or OUTPUT control card.

Message (DMKDDR716R) indicates that the device at cuu (as specified in the INPUT or OUTPUT control card) contains no volume serial number:

```
DMKDDR716R NO VOL1 LABEL FOUND FOR volser
           DO YOU WISH TO CONTINUE?  RESPOND YES  NO  OR
           REREAD:
```

volser
: is the volume serial number of the DASD from the INPUT or the OUTPUT control card.

Message (DMKDDR717R) requests verification of the input parameters:

```
DMKDDR717R DATA DUMPED FROM volid1 TO BE RESTORED TO volid2
           DO YOU WISH TO CONTINUE? RESPOND YES  NO  OR
           REREAD:
```

volid1
: is the volume serial number from the input tape header record (volume dumped).

volid2
: is the volume serial number from the output device.

Message (DMKDDR725R) indicates that the input device has more storage units than the output device:

```
DMKDDR725R DASD INPUT DEVICE WAS (IS) LARGER THAN OUTPUT
           DEVICE.
           DO YOU WISH TO CONTINUE? RESPOND YES NO OR
           REREAD:
```

Explanation: RESTORE function - The number of cylinders or blocks on the original DASD input unit is compared with the number on the output device.

COPY function - The input device contains more cylinders or blocks than the output device.

Operator Action: The operator must determine if the COPY or RESTORE function is to continue. The response is either yes, no, or reread.

Other messages from DDR prompt for input and report progress in processing.

The following two messages prompt for input from the terminal:

```
ENTER CYLINDER EXTENTS
ENTER:

ENTER BLOCK EXTENTS
ENTER:
```

The following two messages prompt for the next tape reel:

- `END OF VOLUME CYL xxxx HD xx, MOUNT NEXT TAPE`
- `END OF VOLUME BLOCK xxxxxx, MOUNT NEXT TAPE`

After the tape is mounted, DDR continues processing.

This message reports that the RESTORE operation has begun:

`RESTORING volser`

volser is the volume serial number of the disk dumped.

This message reports that the COPY operation has begun:

`COPYING volser`

volser is the volume serial number described by the input unit.

This message reports that the dumping operation has begun:

`DUMPING volser`

volser is the volume serial number described by the input unit.

This message reports that the PRINT operation has begun:

```
PRINTING volser
```

volser is the volume serial number described by the input unit.

This message reports that the DUMP operation has ended:

```
END OF DUMP
```

This message reports that the RESTORE operation has ended:

```
END OF RESTORE
```

This message reports that the COPY operation has ended:

```
END OF COPY
```

This message reports that the PRINT operation has ended:

```
END OF PRINT
```

This message reports that all specified operations have completed:

```
END OF JOB
```

When this message

```
Enter:
```

prompts for input from the terminal, pressing the ENTER key (or equivalent) causes control to return to CMS, if the virtual machine is in the CMS environment.

In addition to these responses, other messages that require the operator to continue, terminate, or reinitiate the current operation are described in *VM/SP System Messages and Codes*.

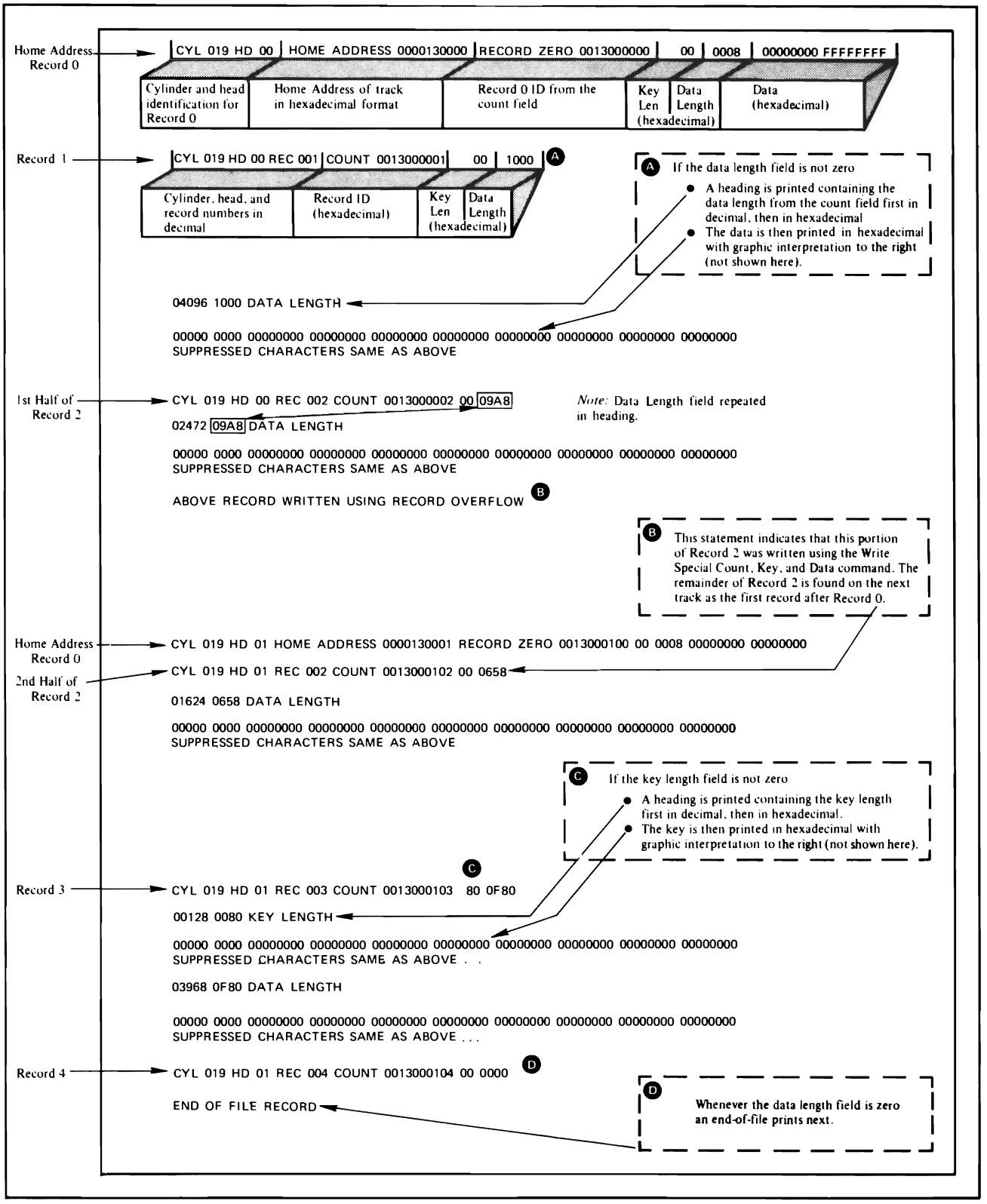


Figure 17. Annotated Sample of Output from the TYPE and PRINT Functions of the DDR Program

Chapter 16. VM/SP Use of the IBM 3850 MSS

Virtual machines operating CMS, OS/VS1, or OS/VS2 (MVS) may access mass storage volumes containing VM/SP minidisks or entire mass storage volumes dedicated to the virtual machine. These volumes appear to the virtual machine as 3330 volumes and are accessed using 3330 device support in the virtual machine. VM/SP controls allocation, volume mounting, and volume demounting. Virtual machines that run OS/VS1 or OS/VS2 (MVS) with MSS support can also access mass storage volumes using dedicated device support.

VM/SP Access to the MASS Storage Control

Whenever an MSS 3330V volume must be mounted or demounted, the VM/SP control program first selects an appropriate device address. If a volume mount is required, the device is selected from the pool of available 3330V devices created at system generation time. If a volume must be demounted, CP selects the address of the device on which the volume is currently mounted.

To pass mount and demount orders, the virtual machine must have an MSC port dedicated to it via the ATTACH command or the DEDICATE directory statement. An application program named DMKMSS is distributed as part of VM/SP; it acts as an interface between CP and the MSC. After DMKMSS is started in an OS/VS1 or OS/VS2 (MVS) virtual machine, it uses a special virtual I/O device and the VM/SP DIAGNOSE interface to communicate with the VM/SP control program.

If the MSC request was for a volume mount, the MSC ending status indicated that the MSC was processed. If the MSC accepts the mount order, the MSC orders the staging adapter to generate a pack change interrupt (an unsolicited device end) on the device when that device has been mounted. CP receives the pack change interrupt, the RDEVBLOK is set to indicate that the volume is mounted, and any VM/SP task waiting for the volume is marked dispatchable. If the mount order was rejected, no further processing of the mount occurs. The previously allocated RDEVBLOK is marked free and processing continues with the next MSS request.

Asynchronous MSS Mount Processing

When an MSS volume mount is required to satisfy a LINK or ATTACH command or an MDISK or DED directory statement, CP returns control to the virtual machine as soon as MSC accepts the mount request[6]. The virtual machine may continue to execute before the virtual device specified on the MDISK, DED, LINK, or ATTACH is available.

The reasons for asynchronous MSS mount processing are the relatively long time required to complete the mount, and the chance that an error may occur in the MSS after the mount order is accepted. The virtual device to be mounted may not be vital to the specific task to be accomplished. Also, if an error occurs in the MSS (such as a permanent read error on a cartridge) after the mount is accepted, the error indication is passed from the MSC to the virtual machine. VM/SP cannot determine that an error has occurred and that the mount will not complete. If the virtual machine were not dispatchable until the mount completed, it would be locked out until the MSS error was corrected.

With asynchronous mount processing, the virtual machine has the flexibility to either continue processing without the affected virtual device, or wait until the MSS mount completes. If the virtual machine issues an SIO instruction to a virtual device that is defined on the volume being mounted, VM/SP queues the I/O request until the mount completes. The virtual machine is marked I/O wait nondispatchable until the mount completes and the SIO is started.

VM/SP Processing of MSS Cylinder Faults

VM/SP supports 3330V cylinder fault processing in two ways: real channel programs directed to 3330Vs are constructed so that cylinder faults can be recognized and the channel program restarted; and the attention interruption (indicating that the cylinder fault has been satisfied) is recognized and any I/O for that device restarted.

When the VM/SP processor issues a seek CCW to a 3330V device, the staging adapter must translate the seek argument to the correct cylinder of staging space. If the cylinder is staged, then the SIO is passed to the associated staging DASD drive. If the cylinder is not staged, the staging adapter initiates cylinder fault processing. The staging adapter first passes a cylinder fault indication to the MSC, requesting the cylinder of data to be staged. It then returns a status modifier to the channel in response to the seek, which causes the channel to skip one CCW in its CCW fetch

---

6 However, the central server cannot issue these CP commands. The central server is the MSS communicator virtual machine which acts as an interface between CP and the MSC. CP commands issued to the central server are ignored and a message is issued.

processing. That is, the channel does not fetch the next CCW after the seek.

As a result of the cylinder fault, the MSC allocates staging space for the requested data and causes it to be staged. The staging adapter then generates a channel end/device end interruption to indicate that the cylinder has been staged.

It is possible in error situations that the attention interruption may not be received. Each time an I/O request is queued by VM/SP as a result of a cylinder fault, a timer is set. If the timer expires before the interruption is received, a message (DMKSSS074I) is written to the VM/SP system operator and the request is retried.

Backup and Recovery of MSS Volumes

The process of creating backup copies of MSS volumes, and restoring from those backup copies, can be controlled through the OS/VS access methods services COPYV command. This command can operate without system operator intervention.

For each active volume in the MSS, there may be one or more copy volumes. At any time, the active volume may be copied to a copy volume with the access method services COPYV command. All volume mounts and data transfer are controlled by this command. If at any time it is necessary to restore the level of a volume to that of a copy, the OS/VS access methods services RECOVERV command is used.

All the OS/VS access methods services commands can be run from either a real processor or a VS virtual machine. If the MSS communicator virtual machine is in operation, these commands can be run from that virtual machine while it is acting as the communicator.

Chapter 17. Timers in a Virtual Machine

This chapter describes the timing facilities available to a virtual machine created by CP.

Interval Timer

The interval timer at virtual location 80 (X'50') does not behave exactly like a real machine's. On a real machine, the interval timer is updated 300 times per second when enabled and when the real machine is not in manual state. The interval timer on a real machine thus reflects system time and wait state time.

In a virtual machine, the interval timer reflects virtual processor time but not wait time. (This behavior can be controlled with the virtual timer assist feature. See the REALTIMER option in the next subsection.) CP adds the virtual processor time used to the virtual interval timer when it regains control, and this one updating reflects the entire time the virtual machine had control. During the time a virtual machine has control, the virtual interval timer does not change.

For some privileged instructions, CP can to simulate the instruction and still return control to the virtual machine before the end of that virtual machine's queue slice. In such cases, CP updates the virtual interval timer. This happens only with those privileged instructions that require normal or fast reflect entry into the dispatcher. For those privileged instructions that do not require entry into the dispatcher, the virtual interval timer is not updated until CP gets control at the end of the queue slice.

If the virtual machine assist feature or Extended Control Program Support (ECPS) is ON, more time is charged to the virtual interval timer than if the feature is OFF. When the virtual machine assist feature is OFF, the time spent by CP to simulate privileged instructions *is not* charged to the virtual interval timer; whereas, with the feature ON, the time spent *is* charged to the virtual interval timer.

Virtual Interval Timer Assist

The virtual interval timer assist is a hardware feature that updates the virtual interval timer and presents timer interrupts to the virtual machine. When the software simulates the interval timer, updating occurs only when CP takes over control. This usually results in an update frequency of once per queue slice and the intervals between updates are irregular. When the virtual interval timer assist feature is active, both virtual and real interval timers are updated 300 times per second.

For the virtual interval timer assist feature to be active, the following conditions must be met:

- VM/SP must be running on a Model 135-3, 138, 145-3, 148, 3031, 3031AP, 4321, 4331, 4341, 4361, or 4381.
- The assist feature must have been enabled for the system by the class A command SET SASSIST ON.
- The virtual machine must have enabled the virtual interval timer assist option by the class G command SET ASSIST ON TMR.

VM/SP provides an option, called the REALTIMER option, which causes the virtual interval timer to be updated during virtual wait state as well. With the REALTIMER option in effect, a virtual interval timer reflects virtual processor time and virtual wait time, but not CP time used for services for that virtual machine, such as privileged instruction execution. The more services a virtual machine requires from CP, the greater the difference between the time represented by the interval timer and the actual time used by and for the virtual machine. The larger the number of active virtual machines contending for system resources, the greater the difference between virtual machine time and actual elapsed (wall clock) time.

The REALTIMER option is turned on by the class G command SET TIMER REAL or by the REALTIMER parameter on the OPTION control statement of the user's directory.

Processor Timers

A virtual machine must have the ECMODE directory option to use the System/370 processor (CPU) timer.

The CPU timer is decremented when the virtual machine is in the running state, and in a virtual PSW wait state. It is not decremented when runnable but undispatched, or when in a CP function. The CPU timer is unaffected by the setting of SET TIMER (whether OFF, ON, or REAL).

The interval timer is decremented when the virtual machine is in the running state if SET TIMER is ON or REAL. It is decremented in virtual PSW wait state if and only if SET TIMER is REAL. It is not decremented when runnable but undispatched, or when in a CP function, regardless of the setting of SET TIMER. It is never decremented if TIMER is OFF.

The method of sampling the value in the CPU timer causes it to appear to a virtual machine to be updated more often than an interval timer. The privileged instructions Set Processor Timer (SPT) and STore Processor Timer (STPT) are used to set a doubleword value in the CPU timer and to store it in a doubleword location of virtual storage. When a virtual machine samples the value in the virtual processor timer by issuing a STPT instruction, CP regains control to execute the privileged instruction, and

updates the time. The act of sampling the CPU timer from a virtual machine causes it to be brought up to date.

TOD Clock

The System/370 time-of-day (TOD) clock does not require simulation in a virtual machine. The System/370 in which CP is operating may have one real TOD clock for each processor, and all virtual machines can interrogate the real TOD clock. The Store Clock (STCK) instruction is nonprivileged; any virtual machine can execute it to store the current value of the TOD clock in its virtual storage. The Set Clock (SCK) instruction, which is used to set the TOD Clock value, can be issued from a virtual machine, but CP always returns a condition code of zero and does not actually set the clock. Note that the TOD clock is the only true source of actual elapsed time information for a virtual machine. The base value for the TOD clock in VM/SP is 00:00:00 GMT, January 1, 1900.

4361 processors are offered with an Auto Start feature. Using a battery operated clock, this feature maintains the time while the power is off. For these processors, if you shut down the system using the SHUTDOWN command with the POWEROFF parameter, then during the next IPL, you will not be prompted to set the time-of-day clock.

In an attached processor or multiprocessor environment, the TOD clocks are synchronized using the procedure described in the *IBM System/370: Principles of Operation*.

Clock Comparator

The clock comparator associated with the TOD clock is used in virtual machines for generating interrupts based on actual elapsed time. The ECMODE option must be specified for a virtual machine to use the clock comparator feature. The Set Clock Comparator (SCKC) instruction specifies a doubleword value that is placed in the clock comparator. When the TOD clock passes that value, an interrupt is generated.

Pseudo Timer

The pseudo timer is a special VM/SP timing facility. It provides 24 or 32 bytes of time and date information in the format shown in Figure 18 on page 168.

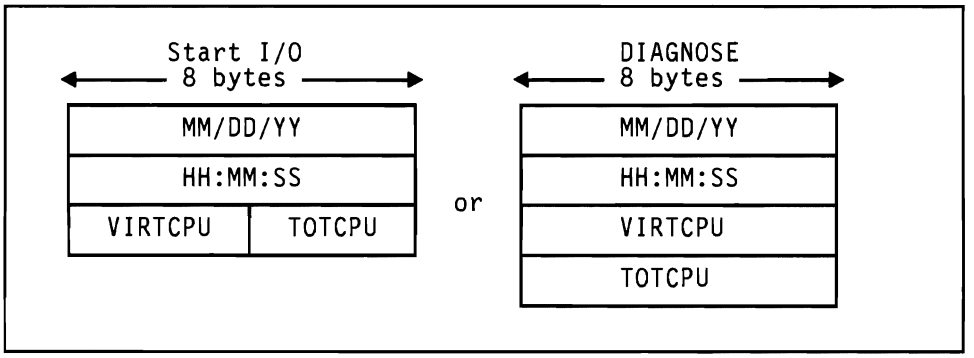


Figure 18. Formats of Pseudo Timer Information

The first eight-byte field is the date, in EBCDIC, in the form Month/Date/Year. The next eight-byte field is the Time of Day in Hours:Minutes:Seconds. The VIRTCPU and TOTCPU fields contain virtual processor and total processor time used. The units in which the processor times are expressed and the length of the fields depend upon which of two methods is used for interrogating the pseudo timer.

Pseudo Timer Start I/O

The pseudo timer can be interrogated by issuing a START I/O to the pseudo timer device, which is device type TIMER, and is usually at device address 0FF. No I/O interrupt is returned from the SIO. The address in virtual storage where the timer information is to be placed is specified in the data address portion of the CCW associated with the SIO. This address must not cross a page boundary in the user's address space. If this method is used, the virtual processor and the total processor times are expressed as fullwords in high resolution interval timer units. One unit is 13 microseconds.

Pseudo Timer DIAGNOSE

The pseudo timer can also be interrogated by issuing DIAGNOSE with an operation code of C, as described in *VM System Facilities for Programming*. If this method is used, the virtual and total processor times are expressed as doublewords in microseconds.

Chapter 18. CP in Attached Processor and Multiprocessor Modes

This chapter describes how to:

- Define attached processor (AP) mode
- Define multiprocessor (MP) mode
- Understand the use of the channel set switching instructions
- Understand the use of the privileged instructions that set and inspect the processor's prefix register
- Understand the use of the privileged instruction that determines the address of the processor that is executing
- Understand the use of hardware signaling to communicate between processors
- Understand the use of a TOD clock synchronization check
- Code fetch and store sequences that can be safely used in the AP/MP environment
- Use locks for serialization of functions
- Set processor affinity
- Change processors using the SWTCHVM macro
- Configure I/O devices to obtain maximum availability and recovery potential

Multiprocessor Environment

In a tightly coupled multiprocessor (MP) environment, two processors share real storage under the control of a single control program. Both processors have I/O capability in an MP environment. See "Configuring I/O Devices for an MP System" on page 182 for a discussion on how to configure I/O devices for maximum availability and recovery potential.

In a dyadic environment, two processors share real storage under the control of a single control program. Both processors have I/O capability. However, unlike an MP complex, a dyadic processor cannot be partitioned into two distinct uniprocessor systems.

Attached Processor Environment

In an attached processor (AP) environment, two processors share real storage under the control of a single control program. However, unlike a multiprocessing environment, only one processor in an AP environment has I/O capability. If you are running on a 3033 or a 3081, the channel set switching feature is available. If a severe hardware error occurs on the first processor in an AP environment, the control program may be able to use the channel set switching feature to switch the channels of one processor to the other processor. The channel set switching instructions that the control program can use to connect and disconnect a channel set to a processor are:

```
CONCS   connect channel set
DISCS   disconnect channel set
```

Note: When you generate VM/SP as an MP system it does not use the channel set switching facility even if the facility is installed on the hardware.

Advantages of the AP/MP Environment

An AP/MP environment provides additional processing capability when compared to a uniprocessor environment. An AP/MP environment also provides increased availability. In case of hardware malfunction on one processor, the other processor can frequently continue operating. Serviceability is enhanced because it is possible to use the VARY ON/OFF PROCESSOR command to vary a processor off-line for system repair or to upgrade the system.

Facilitating an AP/MP Environment

In an AP or MP environment, two processors share main storage. To facilitate this sharing, VM/SP provides for the unique features and requirements of this environment: prefixing, processor address identification, processor signaling, time-of-day clock synchronization, interlocks on certain fetch and store instructions, locks, and affinity setting. The system programmer should be familiar with the instructions used to accomplish these tasks.

Prefixing

When VM/SP is executing in an AP/MP environment both processors cannot use absolute page zero for status information. Instead, each processor has its own prefixed storage area (PSA) in the high end of real storage. However, if the system operator varies a processor on-line after CP initialization completes, the processor's PSA may be located in any page of

the dynamic paging area. See Figure 19 for a storage map of the V=R machine after CP initialization.

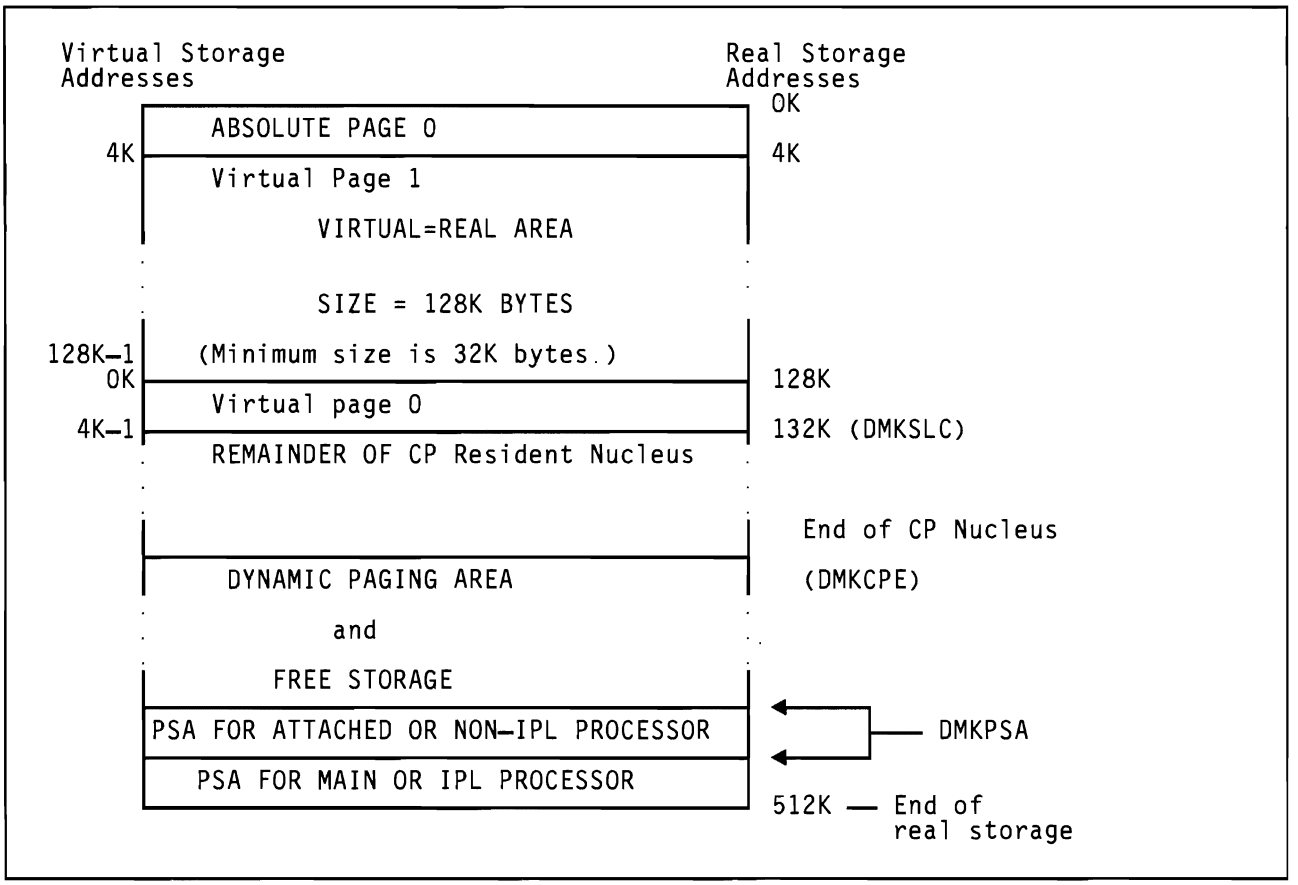


Figure 19. Storage Layout in a Virtual=Real Machine

The control program puts the addresses of the PSAs in the prefix registers of the two processors during system initialization. The control program can set and inspect the contents of the processor's prefix register by using these privileged instructions:

- SPX - set prefix
- STPX - store prefix.

If you are operating in AP/MP mode, VM/SP uses the prefix registers. When code executing on either processor refers to an address from 0 to 4095, the address is added to the contents of the prefix register for that processor to produce the absolute address that will be accessed. In this way, each processor can independently control its operations with separate channel address words and channel status words. Prefixing is described in detail in *System/370 Principles of Operation.*

Identifying a Processor Address

The hardware assigns the processor address during system installation. To determine the address of the processor that is executing, the control program issues the privileged instruction:

- STAP store CPU address.

VM/SP stores both processor addresses in both PSAs in the following fields:

- IPUADDR CPU address of this processor
- IPUADDRX CPU address of the other processor.

The system uses this information for interprocessor communication.

Signaling with the SIGNAL Macro

During certain critical periods, such as when a processor malfunctions or when a processor synchronization must occurs, one processor must signal the other processor. There are three types of program-controlled signals possible under VM/SP. They are:

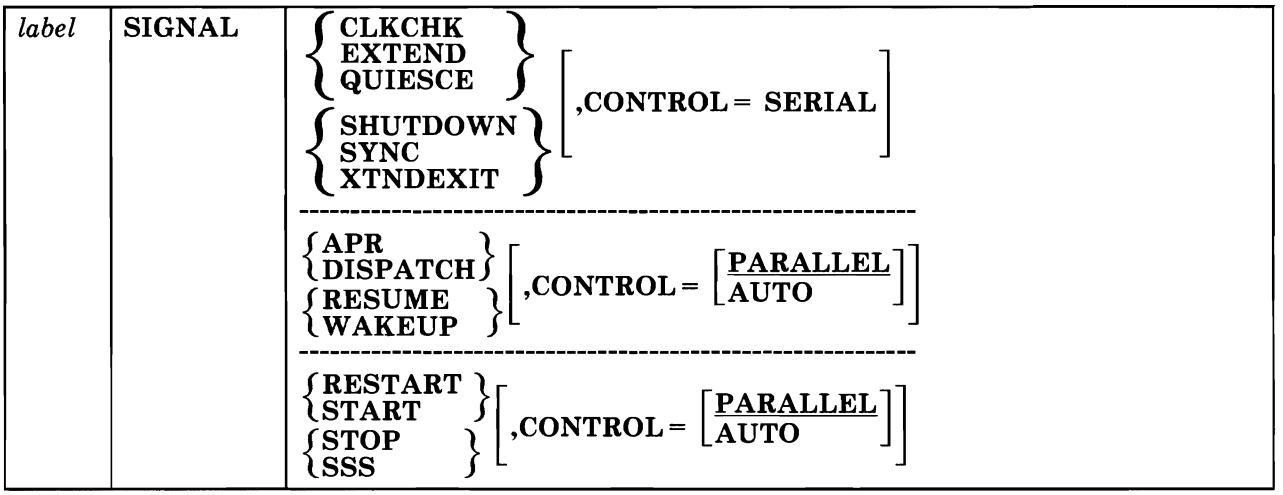
- Emergency signals
- Direct signals
- External call signals.

Use the SIGNAL macro to issue the signal processor (SIGP) instruction. If you have generated the system as an AP/MP system, the control program expands the macro. The macro expansion code destroys the contents of registers 0, 1, 14, and 15. The macro expansion loads register 0 with the signalled processor address, loads register 1 with the function code, and uses registers 14 and 15 for linkage.

Note: If you have not generated the system as an AP/MP system, the control program treats the SIGNAL macro as a no-operation.

The SIGNAL macro causes all signaling requests to be sent to the external interruption handler so that error analysis and recovery attempts are centralized.

The format of the SIGNAL macro and the functions that you can perform using each type of signal are:



where:

label
is any desired label.

first operand
is the function to be performed and is a required positional parameter. This parameter can be an emergency signal, an external call signal, or a direct signal.

Emergency Signals
When one processor wants the other processor to perform an action immediately, it executes an emergency signal instruction. Since emergency signals can only be serial, control is not returned to the issuing processor until the other processor performs the function. The emergency signals are:

CLKCHK
indicates that the high order bits of the time-of-day clocks are not synchronized.

EXTEND
indicates that free storage extend processing is to take place.

QUIESCE
indicates that the receiving processor is to halt all execution until it receives a RESUME signal.

SHUTDOWN
indicates that the system is about to shutdown.

SYNC
indicates that the low order bits of the time of day clocks are no longer synchronized.

XTNDEXIT
indicates that the free storage extension is complete and virtual machines can be dispatched again.

External Call Signals
When one processor wants to call the other processor's attention to an event or condition, it executes an external call order. The external call functions are:

APR
causes automatic processor recovery to be invoked to try to remove a failing processor from the configuration

DISPATCH
indicates that a CPEXBLOX is on the dispatcher's queue for the receiving processor

RESUME
cancels a previous QUIESCE signal

WAKEUP
indicates that the processor is to resume operations after having stopped processing

Direct Signals
Direct signals correspond to physical buttons on the real processor. These signals are controlled by the hardware, and cannot be masked off. The direct signals are:

RESTART

START

STOP

SSS
stop and store status.

,CONTROL=
is the second operand.

,CONTROL=SERIAL
specifies that control returns to the sender after the function is complete. CONTROL=SERIAL is the only parameter that you can use with the emergency signals. You cannot specify CONTROL=SERIAL for the external calls and direct signals.

,CONTROL=PARALLEL
specifies that control returns to the sender even though the function may not be complete. You can use CONTROL=PARALLEL with the external calls and direct signals; it is the default for these signals.

,CONTROL=AUTO
specifies that the signal is sent to the issuing processor. You can use CONTROL=AUTO with the external calls and direct signals.

Time-of-Day (TOD) Clock Synchronization Check

If more than one TOD clock exists in a tightly coupled configuration, the clocks must be synchronized. If TOD clocks are not in high order synchronization during initialization of an AP/MP system, the system issues a message to the system operator to enable the TOD clock set key. If the clocks are out of low order synchronization, that is bits 32 to 63 of the two clocks do not match, the system receives a time-of-day-clock-sync-check when external interruptions are enabled. Then the system synchronizes the clocks.

Fetching and Storing

Since main storage is shared, there is a possibility that both processors may be accessing the same location in storage simultaneously. The control program must prevent simultaneous updates to the same storage location. In a tightly-coupled multiprocessor environment certain instructions cannot safely execute if there is a chance that their execution might change storage that the other processor is also using. Fetch and store instructions such as OI, NI, and NC could cause one processor to update storage that the other processor is also using. To prevent this type of error in a multiprocessing environment, the following fetch and store instructions have interlocks:

- CDS - compare double and swap
- CS - compare and swap
- TS - test and set.

The following example shows how you could use the compare and swap instruction to set a flag in a multiprocessing environment.

```
        Processor A

        LA   Rx,FLAGS      load Rx with address of FLAGS byte
        LA   Ry,X'80'      load Ry with byte to set FLAGS
        SLL  Ry,24         line up fields
        L    Rz,0(Rx)      load Rz with FLAGS byte
 RETRY  LR   Rw,Rz         load Rw with contents of Rz
        OR   Rw,Ry         load Rw with reset value of FLAGS
        CS   Rz,Rw,0(Rx)   reset FLAGS byte if =; otherwise load Rz from
FLAGS
        BNE  RETRY         if contents of Rz ≠ FLAGS, branch to
RETRY
        .
        .
        .
 FLAGS  DC   X'20'         initial setting of field



        Processor B

        LA   Ra,FLAGS      load Ra with address of FLAGS byte
        LA   Rb,X'40'      load Rb with byte to set FLAGS
        SLL  Rb,24         line up fields
        L    Rc,0(Ra)      load Rc with FLAGS byte
 RETRY  LR   Rd,Rc         load Rd with contents of Rc
        OR   Rd,Rb         load Ra with reset value of FLAGS
        CS   Rc,Rd,0(Ra)   reset FLAGS byte if =; otherwise load Rc from
FLAGS
        BNE  RETRY         if contents of Rc ≠ FLAGS, branch to
RETRY
        .
        .
        .
 FLAGS  DC   X'20'
```

Figure 20. Sample of the Correct Way to Set a Flag in an AP/MP Environment

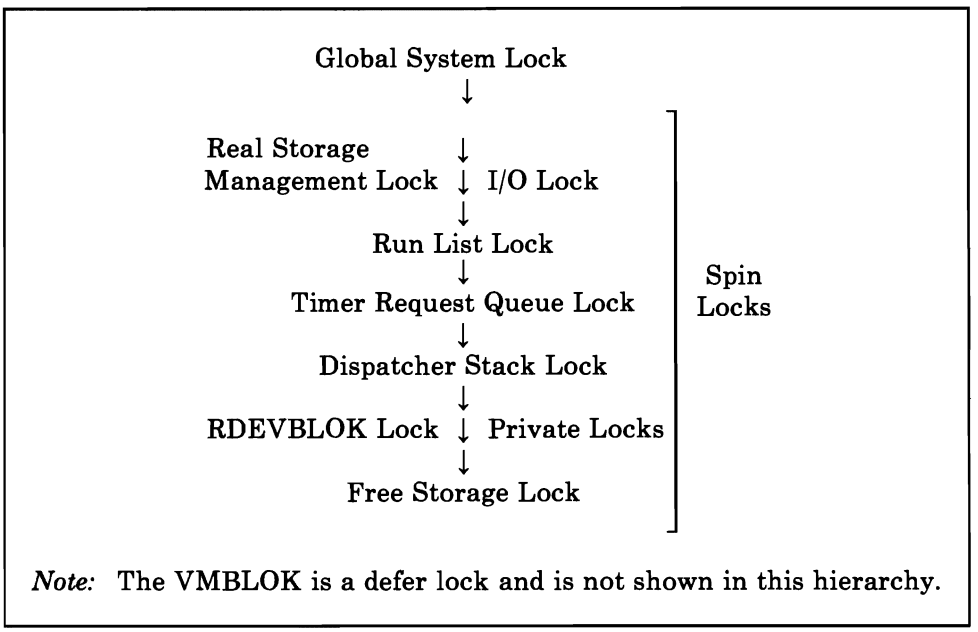
Locks and Serialization of Functions

If VM/SP is executing in AP/MP mode, critical sections of code must be serialized. A critical section of code is code that is executing on one processor and must appear as one indivisible operation to the other processor. An example of a critical section of code is code that updates a queue. The other processor should not have access to the queue until the element is either added or deleted and all pointers are updated. VM/SP uses locks to accomplish serialization of critical functions. A lock is an area of storage. It is initialized to a value, usually zero, to signify that the lock is not held. Before entering a critical section of code, the processor requests the lock to serialize the operation. The operating system determines if a lock is free and gives it to the processor requesting the lock by means of a hardware interlocked update operation such as compare and swap (CS). When the critical section of code has been executed, the system *releases* the lock by changing its value back to zero.

Locking Hierarchy

The introduction of a locking structure makes the avoidance of processor deadlock a prime concern. A deadlock occurs if the processors have different locks and want to obtain the lock that the other processor holds. VM/SP uses a locking hierarchy to avoid these deadlock situations. A locking hierarchy provides for the ordering of the set of locks. If a processor holds a given lock, it can only request a lock that is lower in the locking hierarchy. For example if a processor holds the free storage lock, the processor cannot perform input/output. On the other hand, if a processor holds the I/O lock, the processor can obtain free storage.

Figure 21 shows the hierarchy of locks under VM/SP where the global system lock is the highest lock. The real storage management lock and the I/O lock are on the same level, since there is no situation that requires simultaneous ownership of the I/O lock and the real storage management lock. If a conflict arises, the system will define a hierarchy between these locks.



Note: The VMBLOK is a defer lock and is not shown in this hierarchy.

Figure 21. Hierarchy of VM/SP Locks

Locking Structure

There are two basic types of locks:

- Defer locks
- Spin locks.

If a function requests a defer lock and it is not available, control is returned to the caller with a condition code that indicates that the lock is not available. However, if a function requests a spin lock and it is not available, the lock manager loops until the lock becomes available.

To provide system integrity, VM/SP attached processor and multiprocessor support is designed around one global lock, a VMBLOK local lock, and several system local locks for specifically identified queues or modules.

Global System Lock: Much of CP runs under the global system lock, which is a defer lock. For example, all command processing requires the global system lock. Also, all code executed via an IOBLOK, TRQBLOK, or CPEXBLOK is protected by the global system lock. Certain basic system functions, however, are able to execute without the global system lock on the mainline, non-error paths. These functions include virtual page fault processing, the simulation of virtual I/O requests and some other privileged operations, and the processing of a real I/O interruption.

If a processor needs the global system lock and cannot obtain it, the processor must defer the function until the global system lock is available. The function is deferred by either stacking the VMBLOK appendage (called the deferred interrupt block) or a CPEXBLOK for later processing. The processor that could not obtain the global system lock then uses the unlocked dispatcher entry to dispatch a new virtual machine.

In some situations, a function cannot be deferred even though the global system lock is not available. In these cases, the *dispatcher* spins on the global system lock until it becomes available. The dispatcher requires the system lock to unstack CPEXBLOKs, IOBLOKs, and TRQBLOKs.

To ensure system integrity along the paths that do not require the global system lock, other local locks have been defined. With the exception of the VMBLOK lock, these locks are all spin locks and are held for relatively short periods of time.

VMBLOK Lock: The VMBLOK lock, which is a defer lock, is obtained by the dispatcher before dispatching a virtual machine in problem program state or before performing any system service for that virtual machine. This lock prevents a virtual machine from being serviced by CP while it is running in problem program state.

Real Storage Management Lock (RM Lock): The real storage management lock (called the RM lock) is a spin lock that serializes functions within the paging subsystem. This lock controls all accesses to the free and flush lists, the page read and write request queues, the deferred allocation queue, the active paging queue, CPEXBLOKs chained via CPEXMISC, and certain nonreentrant fields within DMKPTR and DMKPAG.

I/O Lock: The I/O lock is a spin lock that serializes access to I/O devices by serializing access to fields in the real I/O control blocks: RCHBLOK, RCUBLOK, and RDEVBLOK.

Run List Lock: The run list lock is a spin lock that controls all additions to and deletions from the run list.

Timer Request Queue Lock: The timer request queue lock is a spin lock that allows the external first-level interruption handler to process a timer interruption without the global system lock.

Dispatcher Stack Lock: The dispatcher stack lock is a spin lock that controls all additions to or deletions from the IOBLOK/TRQBLOK queue or the CPEXBLOK queue.

RDEVBLOK Lock: The RDEVBLOK lock is a private spin lock that the paging subsystem uses to serialize the IOBLOK queue.

Free Storage Lock: The free storage lock is a spin lock obtained by DMKFRE and DMKFRT for FREE and FRET requests for free storage. All of the locks that CP uses are described in further detail in *VM/SP System Logic and Problem Determination Guide Volume 1 (CP).*

LOCK Macro

Use the LOCK macro to obtain or release a lock. The format of the LOCK macro is:

label	**LOCK**	**{OBTAIN / RELEASE},TYPE={SYS / VMBLOK / FREE / RL / TR / DS / IO / RM / PRIVATE} [,SPIN={YES / NO}[,SAVE] ,OPTion = NOUPDT]**

where

label
is any desired label.

{OBTAIN / RELEASE}
is a required positional operand indicating whether the lock is to be obtained or released.

TYPE=
is a required keyword parameter. The possible values are:

SYS	for the global system lock
VMBLOK	for the VMBLOK
FREE	for the free storage lock
RL	for the runlist lock

TR for the timer request queue lock

DS for the dispatch lock

IO for the I/O lock

RM for the real storage management lock

PRIVATE for a private user-defined lock. If you have user-defined areas that are used by more than one virtual machine, you will need to define your own locking conventions. You can use the LOCK macro to obtain and release a private lock.

You must specify the address of the lockword in register 1 and the lockword must be a fullword aligned on a fullword boundary. Spin time for private locks is kept in the DMKLOKSI timer value for all non−DMKLOK locks.

,SPIN = {**YES** / **NO**}

specifies whether control is to be returned without the lock being held. The default is SPIN=YES.

,SAVE

is an optional keyword that indicates register 0, 1, 14, and 15 are to be saved before the rest of the macro expansion. These registers are saved in the PSA of the processor that is executing this macro. The registers are restored before exit from the macro expansion.

,OPTion= NOUPDT

indicates that the VMBLOK should be locked without checking for shared segments.

Condition Codes

The condition code (cc) is set by the LOCK macro:.

Condition Code	**Parameter**	**Meaning**
cc=0	OBTAIN	lock obtained
	RELEASE	lock released
cc=1	OBTAIN,SPIN=NO	lock owned by another processor

For various abend codes related to lock use, see *VM/SP System Messages and Codes*.

Affinity

When you specify the affinity option for a virtual machine, the program of that virtual machine is executed only on the specified processor. You might want to specify affinity in the following cases:

- If one processor has a special hardware feature or a special RPQ that is required for a particular program, set affinity to this processor.
- If a virtual machine has a high I/O-to-compute ratio, you might want to set affinity to the I/O processor. On the other hand, if a virtual machine has a high compute-to-I/O ratio, you could set affinity to the attached processor.

How to Set Affinity

You request affinity either in the directory or with a SET AFFINITY command. See the *VM/SP CP Command Reference* for details on the class G SET AFFINITY command. See the *VM/SP Operator's Guide* for other privilege classes of the SET AFFINITY command.

Shared Segments in an AP/MP Environment

When two processors are executing simultaneously, it is necessary to know when a user changes a shared page. In attached processor or multiprocessor mode, there are two sets of page tables and swap tables maintained for each shared segment unless a user is running unprotected. If a user is running unprotected shared segments, there is only one copy.

SWTCHVM Macro

Routines that must lock a virtual machine other than the current virtual machine use the SWTCHVM macro. The SWTCHVM macro unlocks the VMBLOK specified in register 11 and locks the VMBLOK specified in register 1. Time charging is also switched. The format of the SWTCHVM macro is:

label	**SWTCHVM**	**OPT** = { **[STAY] [NOUPDT]** / **[UNLOCK]** }

where:

label
: is any desired label

STAY
: indicates that if the VMBLOK lock is not available, a CPEXBLOK will be stacked for the current processor.

NOUPDT
indicates that the VMBLOK should be locked without checking for shared segments.

UNLOCK
indicates that the current VMBLOK is unlocked, register 11 is updated to point to VMBLOK specified in register 1, the timer is switched to start charging supervisor time to the new VMBLOK, but the new VMBLOK is not locked. Note: The UNLOCK option cannot be specified with either of the other options.

Configuring I/O Devices for an MP System

When you configure I/O devices, you should consider the following:

- The possibility of a hardware failure
- Smooth transition when you reconfigure between MP and uniprocessor (UP) modes for maintenance.

In either of these cases to ensure maximum system availability, you should provide paths from both processors to I/O devices. You can do this in several ways:

- Configure symmetrically as many channels and I/O devices as possible.
- Install channel-switching and string-switching features on control units where possible. A channel switch is a feature on a control unit that enables two real processors to share a symmetric device. A symmetric device is a device that can be accessed by both processors, while an asymmetric device cannot be shared. A string switch enables you to attach a symmetric I/O device to two separate control units. These features provide access to I/O devices from both processors. This increased access reduces the possible loss of access to critical I/O devices because of hardware malfunctioning.
- Symmetric devices are also defined as alternate path devices. Reserve/release support is mutually exclusive with alternate path support.
- Configure asymmetric devices through a manual switching unit. Then the operator can physically attach these devices to either processor, one processor at a time. Asymmetric devices include printers, card readers, punches, and information display systems.
- Provide redundant control units for critical I/O devices.

Chapter 19. Print Buffers and Forms Control

The 3203, 3262, 3289 Model 4, 4245, and 4248 use the same type Forms Control Buffer (FCB) as the 3211 Printer. The 4248 can also use the extended FCB. The FCB loaded in a virtual 3203, 3211, or 3262 should be compatible with the FCB loaded in the real counterpart. Otherwise, the results can be unpredictable. The 3203 and 3262 use the Universal Character Set (UCS) used by the 1403 Printer.

The 3203 and 3262 attach a 64-byte associative field to the end of the UCS to check, during print line buffer (PLB) loading, that each character loaded into the PLB for printing is also on the print train. The 3203 associative field is exactly like the 3211 associative field described in Figure 22 on page 191, with the exception that the addresses begin at 240. You also need an associative field when making use of the UCS buffer. Refer to your printer's components description manual for a detailed layout of the associative field.

For the 4245 and 4248, the UCS image is loaded by the printer.

Buffer images are supplied for the Universal Character Set (UCS) buffer, the Universal Character Set Buffer (UCSB), the Font Offset Buffer (FOB), and the Forms Control Buffer (FCB). The VM/SP-supplied buffer images are:

- UCS - for the 1403 and 3203 Printers

Name	Meaning
AN	Normal AN arrangement
HN	Normal HN arrangement
PCAN	Preferred character set, AN
PCHN	Preferred character set, HN
QN	PL/I - 60 graphics
QNC	PL/I - 60 graphics
RN	FORTRAN, COBOL commercial
YN	High speed alphanumeric
TN	Text printing 120 graphics
PN	PL/I - 60 graphics
SN	Text printing 84 graphics

- UCSB - for the 3211 Printer

Name	Meaning
A11	Standard Commercial
H11	Standard Scientific
G11	ASCII

P11 PLI
T11 Text Printing

- UCSB - for the 3262 Printer

Name	Meaning
P48	48 character print image
P52	52 character print image (Austria/Germany)
P63	63 character print image, optimized
P64	64 character print image
P96	96 character print image
P116	116 character print image (French/Canadian)
P128	128 character print image (Katakana)

- FOB - for the 3289 Model 4 printer

Name	Meaning
F48	Font Offset Buffer for the 48-character print belt
F64	Font Offset Buffer for the 64-character print belt
F94	Font Offset Buffer for the 94-character print belt
F127	Font Offset Buffer for the 127-character print belt

- FCB - for the 3203, 3211, 3262, 3289 Model 4, 4245, and 4248 Printers

  Two names are provided for an FCB image.

Name	Meaning
FCB1	Space 6 lines/inch Length of page -- 66 lines Paper size -- 11 inches

Line Represented	Channel Skip Specification
1	1
3	2
5	3
7	4
9	5
11	6
13	7
15	8
19	10
21	11
23	12
64	9

Name **Meaning**

FCB8 · Space 8 lines/inch
Length of page 68 lines
Paper size -- 8.5 inches

Line Represented	Channel Skip Specification
1	1
4	2
8	3
12	4
16	5
20	6
24	7
28	8
32	10
36	11
63	12
66	9

For the exact contents of the buffer images, see:

IBM 2821 Control Unit Component Description

IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide

IBM 3289 Line Printer Model 4 Component Description

IBM 3262 Printers 1 and 11 Component Description.

The following table indicates in which module you must code the images for each printer:

Data Module	Printer
DMKFCB	All 3211 type printers
DMKUCS	UCS image for the 1403 printer
DMKUCC	UCS image for the 3203 printer
DMKPIA	UCS image for the 3289E printer
DMKPIB	UCS image for the 3262-1/11 printer
DMKUCB	UCS image for the 3211 printer

For further information refer to the Component Description of the printer for which the image is being coded.

If you find that the supplied buffer images do not meet your needs, you can alter a buffer image or create a new buffer image. Be careful not to violate the VM/SP coding conventions if you add a new buffer image; buffer images must not cross page boundaries.

Adding New Print Buffer Images

To add a new print buffer image to VM/SP, you must:

1. Provide a buffer image name and 12-byte header for the buffer load.
2. Provide the exact image of the print chain.
3. Provide a means to print the buffer image if VER is specified on the LOADBUF command.
4. Rebuild the CP nucleus so that it includes the changed modules.

The following subsections describe how to use macros to make the process of adding buffer images relatively easy. They should be used to avoid errors.

UCS Buffer Images for the 1403 Printer

The Universal Character Set (UCS) buffer contains up to 240 characters and supports the 1403 printer. To add a new UCS buffer image:

1. Code the UCS macro. This creates a 12-byte header for the buffer load that is used by CP. The format of the UCS macro is:

label	UCS	ucsname

where:

label is an optional label or a blank space.

ucsname is a 1- to 4-character name that is assigned to the buffer load.

2. Supply the exact print image. The print image is supplied by coding DCs in hexadecimal or character format. The print image may consist of several DCs, the total length of the print image cannot exceed 240 characters.
3. The UCSCCW macro must immediately follow the print image. This macro creates a CCW string to print the buffer load image when VER is specified by the operator on the LOADBUF command. The format of the UCSCCW macro is:

label	UCSCCW	ucsname[,(print1,print2,...,print12)]

where:

label is an optional label or a blank space.

ucsname is a 1- to 4-character name that is assigned to the buffer load by the UCS macro.

[(print1,...,print12)]
is the line length (or number of characters to be printed by the corresponding CCW) for the verify operation. Each count specified must be between 1 and 132 (the length of the print line on a 1403 printer) and the default line length is 48 characters. Up to 12 print fields may be specified. However, the total number of characters to be printed may not exceed 240.

4. Finally, insert the macros just coded, UCS and UCSCCW, into the DMKUCS module. This module must be reloaded. DMKUCS is a pageable module with no executable code. DMKUCS must be on a page boundary and cannot exceed a full page in size. If DMKUCS exceeds a page boundary (4K), an error message is issued.

Example 1: You do not have to specify the line length for verification of the buffer load. Insert the following code in DMKUCS:

```
UCS     EX01
DC      5C'1234567890A...Z1234567890*/'
UCSCCW EX01
```

The buffer image is 5 representations of a 48-character string containing:

- The alphabetic characters
- The numeric digits, twice
- The special characters: * and /

Since the line length for the print verification is not specified on the UCSCCW macro, it defaults to 48 characters per line for 5 lines.

Example 2: Insert the following code in DMKUCS:

```
UCS     NUM1
DC      24C'1234567890'
UCSCCW NUM1,(60,60,60,60)
```

The NUM1 print buffer consists of twenty-four 10-character entries. If, after DMKUCS is reloaded, the command

```
LOADBUF  00E  UCS  NUM1  VER
```

is specified, 4 lines of 60 characters (the 10-character string repeated 6 times) are printed to verify the buffer load).

Example 3: The print image can be specified in character or hexadecimal notation, or a combination of the two. The code in DMKUCS to support the preferred character set, AN, is as follows:

```
UCS    PCAN
DC     C'1234567890,-PQR#$@/STUVWXYZ',X'9C'    LOZENGE(9C)
DC     C'.*1234567890,-JKLMNOABCDEFGHI+.*'
DC     C'1234567890,-PQR&&$%/STUVWXYZ',X'9C'   LOZENGE(9C)
DC     C'.*1234567890,-JKLMNOABCDEFGHI+.*'
DC     C'1234567890,-PQR#$@/STUVWXYZ',X'9C'    LOZENGE(9C)
DC     C'.*1234567890,-JKLMNOABCDEFGHI+.*'
DC     C'1234567890,-PQR&&$%/STUVWXYZ',X'9C'   LOZENGE(9C)
DC     C'.*1234567890,-JKLMNOABCDEFGHI+.*'
UCSCCW PCAN,(60,60,60,60)
```

The DCs are coded in both character and hexadecimal notation. The hexadecimal code for the lozenge ('9C') follows the character notation on 4 of the DCs. The DCs, when taken in pairs, represent 60 characters. When print verification of a buffer load is requested, 4 lines of 60 characters are printed.

UCSB Buffer Images for the 3211 Printer

The Universal Character Set Buffer (UCSB) contains up to 512 characters and supports the 3211 printer. To add a new UCSB image:

1. Code the UCB macro. This macro creates a 12-byte header record for the buffer load that is used by CP. The format of the UCB macro is:

label	UCB	ucbname

where:

label is an optional label or a blank space.

ucbname is a 1- to 4-character name that is assigned to the buffer load.

2. Supply the exact print image. The print image is supplied by coding DCs in hexadecimal or character notation. The total length of the print image cannot exceed 512 characters.

The format of the UCSB is:

Position	Contents
1-432	Print train image.
433-447	Reserved for IBM use. Must be all zeroes.
448-511	Associative field. See Figure 22 on page 191 for an explanation of the contents of this field. The associative field is used to check (during print line buffer (PLB) loading) that each character loaded into the PLB for printing also appears in the train image field of the UCSB and, therefore, is on the print train. Any character loaded into the PLB without its associated code in the train image field of the UCSB is nonprintable and causes a **print data**

check to be set immediately. The associative field also contains dual control bits.

For more information on the coding of the associative field, see the Component Description and Operator's Guide for your printer.

512 Reserved for IBM use. Must be zero.

3. The UCBCCW macro must immediately follow the print image. This macro creates a CCW string to print the buffer load image when the operator specifies VER on the LOADBUF command. The format of the UCBCCW macro is:

label	UCBCCW	ucbname[,(print1,print2,...print12)]

where:

label is an optional label or a blank space.

ucbname is 1- to 4-character name that is assigned to the buffer load by the UCB macro.

[(print1,...,print12)]
specifies the line length of each line (up to 12) printed to verify the buffer load. The line length must be between 1 and 150 (the length of a print line on a 3211 printer). The default specification for verification is 48 characters per line for nine lines. The total number of characters to be printed must not exceed the size of the print train image, 432 characters.

4. Finally, insert the two macros just coded, UCB and UCBCCW, into the DMKUCB module. This module must be reloaded before the new buffer image can be used. DMKUCB is a pageable module with no executable code. DMKUCB must be on a page boundary and cannot exceed a full page in size. If DMKUCB exceeds a page boundary (4K), an error message is issued.

For example, The code for the A11 UCSB is as follows:

```
*     A11 STANDARD COMMERCIAL 48 GRAPHICS 3211
      UCB    A11
      DC     9C'1<.+IHGFEDCBA*$-RPQONMLKJ%,&&ZYXWVUTS/@098765432'
      DC     X'000000' 433-435
      DC     X'00000000000000000000000101010' 436-450
      DC     X'10101010101010004040424000 4010' 451-465
      DC     X'10101010101010100040404100 0040' 466-480
      DC     X'40101010101010101000404000 0000' 481-495
      DC     X'10101010101010101010004040 4448' 496-510
      DC     X'0000' 511-512
      UCBCCW A11,(48,48,48,48,48,48,48,48,48)
      EJECT
```

Note that the DC specification contains 49 characters and the UCBCCW macro specifies 48 characters. The ampersand (&) must be coded twice to

be accepted by the assembler. The single quote (') must also be specified twice to be accepted.

It would have been acceptable to code the UCBCCW as:

```
UCBCCW A11
```

since the default is what was coded.

	Bit 0		Bit 1		Bit 2		Bit 3	
UCSB Address	Hexa-decimal	Graphic & Control Symbols EBCDIC	Hexa-decimal	Graphic & Control Symbols EBCDIC	Hexa-decimal	Graphic & Control Symbols EBCDIC	Hexa-decimal	Graphic & Control Symbols EBCDIC
448	00	NUL	40	SP	80		C0	{
449	01		41		81	a	C1	A
450	02		42		82	b	C2	B
451	03		43		83	c	C3	C
452	04	PF	44		84	d	C4	D
453	05	HT	45		85	e	C5	E
454	06	LC	46		86	f	C6	F
455	07	DEL	47		87	g	C7	G
456	08		48		88	h	C8	H
457	09		49		89	i	C9	I
458	0A		4A	¢	8A	{	CA	
459	0B		4B	.	8B	{	CB	
460	0C		4C	<	8C	≤	CC	⌐
461	0D		4D	(	8D	(	CD	
462	0E		4E	+	8E	+	CE	⌐
463	0F	CU1	4F	|	8F		CF	}
464	10		50	&	90		D0	}
465	11		51		91	j	D1	J
466	12		52		92	k	D2	K
467	13		53		93	l	D3	L
468	14	RES	54		94	m	D4	M
469	15	NL	55		95	n	D5	N
470	16	BS	56		96	o	D6	O
471	17	IL	57		97	p	D7	P
472	18		58		98	q	D8	Q
473	19		59		99	r	D9	R
474	1A	CC	5A	!	9A	}	DA	
475	1B		5B	$	9B	}	DB	
476	1C		5C	*	9C	□	DC	
477	1D		5D	)	9D	)	DD	
478	1E		5E	;	9E	±	DE	
479	1F	CU2	5F	¬	9F	■	DF	
480	20		60	-	A0	~	E0	\
481	21		61	/	A1	~	E1	
482	22		62		A2	s	E2	S
483	23		63		A3	t	E3	T
484	24	BYP	64		A4	u	E4	U
485	25	LF	65		A5	v	E5	V
486	26	EOB	66		A6	w	E6	W
487	27	PRE	67		A7	x	E7	X
488	28		68		A8	y	E8	Y
489	29		69		A9	z	E9	Z
490	2A	SM	6A	¦	AA		EA	
491	2B		6B	,	AB	⌊	EB	
492	2C		6C	%	AC	⌈	EC	H
493	2D		6D	>	AD	[	ED	
494	2E		6E		AE	≥	EE	
495	2F	CU3	6F	?	AF	●	EF	
496	30		70		B0	0	F0	0
497	31		71		B1	1	F1	1
498	32		72		B2	2	F2	2
499	33		73		B3	3	F3	3
500	34	PN	74		B4	4	F4	4
501	35	RS	75		B5	5	F5	5
502	36	UC	76		B6	6	F6	6
503	37	EOT	77		B7	7	F7	7
504	38		78		B8	8	F8	8
505	39		79	\	B9	9	F9	9
506	3A		7A	:	BA		FA	
507	3B		7B	#	BB	⌋	FB	
508	3C		7C	@	BC	⌐	FC	
509	3D		7D	'	BD	]	FD	
510	3E		7E	=	BE	≠	FE	
511	3F		7F	"	BF	¯	FF	

Figure 22. UCSB Associative Field Chart

FOB Buffer Images for the 3289 Model 4 Printer

The Font Offset Buffer (FOB) contains 256 font offset bytes and supports the 3289 Model 4 printer. A font offset byte defines a character by specifying its location on a print belt. The location of the character is specified in terms of its distance (offset) from a fixed reference point.

To add a new FOB, create a header for the buffer, supply the contents of the new buffer, and provide a means to print the buffer image if the operator must verify its contents:

1. Code the FOB macro instruction to create a 12-byte header record to be used by the CP. The format of the FOB macro instruction is:

label	FOB	fobname

where:

label is an optional label or a blank space.

fobname is a 1- to 4-character name that is assigned to the buffer.

2. Supply the exact contents of the Font Offset Buffer by coding DCs in hexadecimal format. Several DCs can be coded, but the total buffer length must be 256 bytes. (If the buffer does not contain 256 bytes, the load check bit is set.)

3. The FOBCCW macro instruction must be coded immediately following the DCs that define the buffer contents. This macro instruction creates a CCW string to print the buffer data when the operator specifies VER on the LOADBUF command. The format of the FOBCCW macro instruction is:

label	FOBCCW	fobname [(print1,print2,...print12)]

where:

label is an optional label or a blank space.

fobname is the 1- to 4-character name assigned to the buffer by the FOB macro instruction.

[(print1,...print12)]
specifies the length of each line (up to 12 lines) printed to verify the buffer contents. The line length must be between 1 and 132 (the line length of a 3289 Model 4 printer). The default specification for verification is eight 64-byte lines of hexadecimal formatted data. The total number of hexadecimal bytes to be printed must not exceed 512. (There are two printed bytes for each of the 256 bytes of data in the buffer.)

4. Finally, insert the two macros just coded, FOB and FOBCCW, into the DMKPIA module. This module must be reloaded before the new buffer image can be used. DMKPIA is a pageable module with no executable code. DMKPIA must be on a page boundary and cannot exceed a full page in size. If DMKPIA exceeds a page boundary (4K), an error message is issued.

Example: The F64 Buffer

```
FOB     F64
DC      X'7F',63X'80'
DC      X'7F',9X'80',X'302E31322D3315'
DC      9X'80',X'342223353637210C',9X'80'
DC      X'1617393A3B',9X'80','3F3C0A0B2F303E'
DC      65X'80',X'2425262728292A2B2C'
DC      7X'80',X'18191A1B1C1D1E1F20'
DC      6X'80',X'38800D0E0F1011121314'
DC      6X'80',X'09000102030405060708'
DC      6X'80'
FOBCCW F64,(64,64,64,64,64,64,64,64)
```

UCC Buffer Images for the 3203 Printer

The UCC buffer contains up to 240 characters and supports the 3203 printer. To add a new UCC buffer image:

1. Code the UCC macro. This creates a 12-byte header for the buffer load that is used by CP. The format of the UCC macro is:

label	UCC	uccname

where:

label is an optional label or a blank space.

uccname is a 1- to 4-character name that is assigned to the buffer load.

2. Supply the exact print image. The print image is supplied by coding DCs in hexadecimal or character format. The print image may consist of several DCs, the total length of the print image cannot exceed 240 characters.

3. The UCCCCW macro must immediately follow the 64-byte associative field, which must follow the print image. (See Note 3.) This macro creates a CCW string to print the buffer load image when VER is specified by the operator on the LOADBUF command. The format of the UCCCCW macro is:

label	UCCCCW	uccname[,(print1,print2,...,print12)]

where:

label is an optional label or a blank space.

uccname is a 1- to 4-character name that is assigned to the buffer load by the UCC macro.

[(print1,...,print12)]
is the line length (or number of characters to be printed by the corresponding CCW) for the verify operation. Each count specified must be between 1 and 132 (the length of the print line on a 3203 printer) and the default line length is 48 characters. Up to 12 print fields may be specified. However, the total number of characters to be printed may not exceed 240.

4. Finally, insert the macros just coded, UCC and UCCCCW, into the DMKUCC module. This module must be reloaded. DMKUCC is a pageable module with no executable code. DMKUCC must be on a page boundary and cannot exceed a full page in size. If DMKUCC exceeds a page boundary (4K), an error message is issued.

Example 1: You do not have to specify the line length for verification of the buffer load.

Note: This example is only a representation, and not an actual buffer image.
Insert the following code in DMKUCC:

```
UCC     EX01
DC      5C'1234567890A...Z1234567890*/'
DC      X'C0101010101010101010004000404000'  240-255
DC      X'4010101010101010101004040400000'  256-271
DC      X'4040101010101010101004000000000'  272-287
DC      X'1010101010101010101000000404000'  288-303
UCCCCW EX01
```

The buffer image is 5 representations of a 48-character string containing:

- The alphabetic characters
- The numeric digits, twice
- The special characters: * and /

Since the line length for the print verification is not specified on the UCCCCW macro, it defaults to 48 characters per line for 5 lines.

Example 2: Insert the following code in DMKUCC:

Note: This example is only a representation, and not an actual buffer image.

```
UCC     NUM1
DC      24C'1234567890'
DC      X'C0101010101010101010004000404000'  240-255
DC      X'4010101010101010101004040400000'  256-271
DC      X'4040101010101010101004000000000'  272-287
DC      X'1010101010101010101000000404000'  288-303
UCCCCW NUM1,(60,60,60,60)
```

The NUM1 print buffer consists of twenty-four 10-character entries. If, after DMKUCC is reloaded, the command

```
LOADBUF  00E  UCS  NUM1  VER
```

is specified, 4 lines of 60 characters (the 10-character string repeated 6 times) are printed to verify the buffer load.

Note: The UCS buffer for the 3203 Model 4 and Model 5 printers is programmed essentially the same as for the 1403 printer. You should follow the same procedures for programming this buffer, noting the differences listed below.

1. Instead of the UCS macro, code the UCC macro (which is equivalent to the UCS macro for the 1403).

2. The print train image should be 240 bytes long.

3. Immediately after the print image, the DUAL and UNCOMPARABLE TABLE (DUCT) should appear. The DUCT should be 64 bytes long and start at byte 240 in the UCS buffer.

4. Following the DUCT should be the UCCCCW macro (which is equivalent to the UCSCCW macro).

5. Finally, the completed macros and UCS data should be inserted into the DMKUCC module.

Note that when the UCCCCW macro is coded, you may specify that a maximum of 240 bytes is to be printed.

Also, the UCS buffer for the 3203 Model 5 printer must be 304 bytes long, while the UCS buffer for the 3203 Model 4 printer may be less (however, if the LOADBUF command is issued, the buffer must be 304 bytes long). For information on the effects of the lower UCS buffer length, you should consult the *3203 Component Description and Operator's Guide.*

For more information on coding the DUCT, consult the *IBM 3203 Model 5 Component Description and Operator's Guide*, or the *IBM 3203 Component Description and Operator's Guide*,

PIB Buffer Images for the 3262 Model I and II Printers

The PIB buffer contains up to 132 characters and supports the 3262 printer. To add a new PIB buffer image:

1. Code the PIB macro. This creates a 12-byte header for the buffer load that is used by CP. The format of the PIB macro is:

label	PIB	pibname

where:

label is an optional label or a blank space.

pibname is a 1- to 4-byte character name that is assigned to the buffer load.

2. Supply the exact print image. The print image is supplied by coding DCs in hexadecimal or character format. The print image may consist of several DCs, the total length of the print image cannot exceed 288 characters.

3. The PIBCCW macro must immediately follow the print image. This macro creates a CCW string to print the buffer load image when VER is specified by the operator on the LOADBUF command. The format of the PIBCCW macro is:

label	PIBCCW	pibname [,(print1,print2, ..., print12)]

where:

label is an optional label or a blank space.

pibname is a 1- to 4-character name that is assigned to the buffer load.

[,(print1,print2, ...,print12)]
specifies the length of each line (up to 12 lines) printer to verify the buffer contents. The line length must be between 1 and 132 (the length of the print line on a 3262 printer) and the default line length is 24 characters. Up to 12 print fields can be specified. However, the total number of characters to be printed may not exceed 288.

4. Finally, insert the macros just coded, PIB and PIBCCW, into the DMKPIB module. This module must be reloaded. DMKPIB is a pageable module with no executable code. DMKPIB must not exceed a full page in size. If DMKPIB exceeds a page boundary (4K), an error message is issued.

Examples of New PIB Buffer Images

Example 1: You do not have to specify the line length for verification of the buffer load. Insert the following in DMKPIB:

```
PIB     EX01
DC      8CL36'1234567890...WXYZ'
PIBCCW EX01
```

The buffer image is 8 representations of a 24-character string containing:

- The numeric digits
- The alphabetic characters

Since the line length for the print verification is not specified on the PIBCCW macro, it defaults to 24 characters per line for 8 lines.

Example 2: Insert the following code in DMKPIB:

```
PIB     EX02
DC      8CL36'1234567890...WXYZ'
PIBCCW EX02,(36,36,36,36,36,36,36,36)
```

The EX02 print buffer consists of eight 36-character entries. If, after DMKPIB is reloaded, the command:

```
LOADBUF   00E  UCS   EX02  VER
```

is specified, 8 lines of 36 characters are printed to verify the buffer load.

Forms Control Buffer

It is possible to have a forms control buffer with both a virtual and real 3211-type (3203, 3211, 3262-1/5/11, 3289 Model 4, 4245, 4248) printer. A virtual 3211-type printer file can be printed on a real 1403; in fact, one way to provide forms control for a 1403 is to define it as a virtual 3211. The 4248 can also use the extended forms control buffer.

There is an FCB macro to support 3211-type forms control. The format of the FCB macro is:

label	FCB	fcbname,space,length,(line,channel...),index

where:

label is an optional label or a blank space.

fcbname is the name of the forms control buffer. It can be one to four alphameric characters.

space is the number of lines/inch. Valid specifications are 6 or 8. This operand may be omitted: the default is 6 lines/inch. When the space operand is omitted, a comma (,) must be coded. Spacing has no meaning for a virtual printer.

length is the number of print lines per page or carriage tape (1 to 255).

(line,channel...)
shows which print line (line) prints in each channel (channel...). Channel values range from 1 to 12. VM/SP-supplied buffer images for the FOB are listed at the beginning of this chapter. The entries can be specified in any order.

index is an index value (from 1 to 31). `index` specifies the print position that is to be the first printed position. `index` is valid only for the 3211 printer. The `index` specification is not accepted for other printers. (The `index` specification can be overridden with the LOADBUF command).

VM/SP provides two real FCB images, FCB1 and FCB8. These FCBs are in pageable module DMKFCB. Installations may add additional FCB images to DMKFCB as long as the size of DMKFCB does not exceed the size of two pages.

A default virtual FCB image is provided for virtual 3211-type printers (3203, 3211, 3262-1/5/11, 3289 Model 4, 4245, and 4248). The image is used for the virtual printer if no FCB has been previously loaded for that virtual device (by the LOADVFCB or LOAD FCB commands). It is not stored in the spool file but used only for virtual processing of the print commands. The default FCB image is 66 bytes long; channel 1 is defined on line 1, channel 2 on line 2, . . . channel 12 on line 12.

Notes:

1. *The Forms Control Buffers must have compatibility of channel one; that is, channel one and line one must be the same physical line for all FCBs that are built, or forms misalignment results.*

2. *If the FCB macro is coded to have more than one channel designated for one print line, the macro includes only the last channel in the buffer for that print line. This is because a buffer byte can only be loaded with one channel code.*

3. *When an operator loads a default FCB, it is recommended that all channels be defined to prevent an undefined channel error.*

Example 1: If you wanted your printer to print:

- 8 lines/inch
- 60 lines/page
- print line 3 in channel 1
- print line 60 in channel 9

- print line 40 in channel 12
- print position 10 the first print position

you would code the FCB macro (with a name, SPEC) as:

```
FCB SPEC,8,60,(3,1,40,12,60,9),10
```

If you want another forms control buffer, called LONG, to be exactly the same as SPEC (except that only 6 lines print per inch) you could code either of the following:

```
FCB LONG,6,60,(3,1,40,12,60,9),10
```

```
FCB LONG,,60,(3,1,40,12,60,9),10
```

Example 2: You could have your special forms control buffer (SPEC) loaded for either a virtual or real 3203, 3211, 3262, 3289 Model 4, 4245, or 4248 printer. The LOADVFCB command is for the virtual printer and the LOADBUF command is for the real printer.

The INDEX parameter is only valid for a 3211 printer. If INDEX is not specified for the 3211 printer, no indexing is done. If INDEX is specified without a value, the value coded in the FCB macro is used and if INDEX is specified with a value, the specified value overrides the value coded in the FCB macro.

If you specify INDEX for the virtual 3211 printer and again for the real 3211 printer, the output is indexed using the sum of the two specifications minus 1. For example, the command

```
LOADVFCB 00F FCB SPEC INDEX
```

indexes the virtual print file 10 positions because 10 was specified in the FCB macro for the SPEC forms control buffer. When this file is sent to the real printer, the operator issues the command

```
LOADBUF 00E FCB SPEC INDEX 20
```

which indexes the file an additional 20 positions. The value specified on the command line (20) overrides the value in the FCB macro (10). The output starts printing in print position 29 (10 + 20-1 = 29).

Because the 3203, 3262, 3289 Model 4, 4245, and 4248 printers do not have indexing capabilities, the LOADVFCB and LOADBUF commands with the INDEX option cause an invalid option error message from CP.

The 4248 printer is supported in extended function mode as a virtual, dedicated, or system printer.

The 4248 accepts the FCB (Forms Control Buffer) format for the 3211 printer, and also accepts a new FCB format that supports the printer when it is running in extended function mode. Maximum output line lengths of 132 or 168 bytes are supported depending upon the number of hammer positions installed on the printer.

A spool file containing the extended FCB can be printed only on:

- a printer that supports the FCB type
- a 1403 printer
- a 3800 printer
- a 3211-type printer started with the DEFFCB option.

A 4248 printer should have one of the system output spools defined only to it. This will allow a user to send spool files containing extended FCBs to a printer that supports the FCB.

Extended FCB Macro Instruction

The 4248 accepts either a 3211-type FCB or the new extended FCB. If you want the new features of the extended FCB, code the FCB macro as follows:

Note: The 4248 allows you to specify the print speed and stacker rate.

label	FCB	fcbname,speed,length,fcbdata,offset,levels,rates

where:

label is an optional label or a blank space.

fcbname is a 4-character alphameric name to be used when issuing the LOADBUF or LOADVFCB commands.

speed specifies the print speed of your choice.

U - unchanged
L - low (2200 lines per minute)
M - medium (3000 lines per minute)
H - high (3600 lines per minute)

length is the length of the FCB data; number of print lines on the page (between 2 and 256 inclusive).

fcbdata is the data defining the FCB. The data must be in the following format:

(line, channel, spacing ...)

where:

line represents the numeric position in FCB data. (May be omitted or specified as a number between 1 and the FCB data length.)

channel specifies channel codes 1 through 12, or omitted.

spacing is specified as:

6	6 lines per inch
8	8 lines per inch
(omitted)	no change

Note: Spacing can be changed on every line, not just once per page.

offset is the position where the duplicate copy (if desired) should begin.

levels specifies the stacker level control:

A	automatic stacker level control
1	tray lowered 1 inch below automatic position
2	tray lowered 2 inches below automatic position
3	tray lowered 3 inches below automatic position
(omitted)	automatic stacker level control.

rates specifies the stacker drop rate.

7	drop tray after 7 sheets
13	drop tray after 13 sheets
19	drop tray after 19 sheets
25	drop tray after 25 sheets
(omitted)	drop tray after 7 sheets

Note: If automatic stacker level control is specified, the stacker drop rate is ignored.

Example: If you coded this FCB macro as:

```
FCB EXTE,M,179,(3,1,8,40,2,,179,12,),90,1,25
```

You would be requesting

Extended FCB

printer speed - medium
page length - 179

at line 3:

channel 1 code
spacing changed to 8 lines per inch

at line 40:

channel 2 code
spacing unchanged

at line 179:

channel 12 code
spacing unchanged

Duplicate copy function -- duplicate copy is to begin at print position 90

stacker level -- tray lowered 1 inch below automatic position

stacker rate -- drop tray after 25 sheets

Chapter 20. IBM 3800 Printing Subsystem

The IBM 3800 Printing Subsystem is a high-speed, non-impact printer that combines electro-photographic and laser technology. The 3800 printer can achieve speeds of up to 20,040 lines/minute, while several unique features give the user the ability to control the characteristics of printed output.

VM/SP users have access to the 3800 Model 1 and Model 3 Printing Subsystems. Existing programs designed to produce 3800 Model 1 printer output may produce output for the 3800 Model 3 printer with little or no program change. This is also true of programs designed for a 3800 Model 3 printer.

Use of a 3800 Model 3 printer allows for improved print quality (240 x 240 pel resolution compared with the 3800 Model 1 pel resolution of 180 x 144) and the addition of a 10 lines/inch vertical space option.

The features of the 3800 printer include:

- Forms control buffer, which controls the amount of vertical space between printed lines. The user can specify vertical spacing of 6, 8, or 12 lines/inch. Users formatting on the 3800 Model 3 printer have an additional option of 10 lines/inch.

- Multiple copy printing, which allows the user to request multiple copies without the use of carbon paper. The 3800 uses its high speed to print the specified number of originals.

- Copy modification, which allows the user to print or suppress predefined information on specified copies of a page. For example, a different name and address can be printed on each copy of a page.

- Forms overlay, which allows the user to specify a form or grid to be printed (flashed) from a negative while output is being printed inside the form.

- Character arrangement tables, which allow the user to specify which predefined character set to use for printing a data set. Each character set contains up to 64 printable characters.

- Character modification, which allows character sets to be modified or extended to meet the user's needs.

The 3800 modules control the specified printer control information. These modules are:

- Character Arrangement Tables (CAT)
- Library Character Sets (LCS)
- Graphic Modification Modules (GRAPHMOD)
- Forms Control Buffers (FCB)
- Copy Modification Modules (COPYMOD).

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

For detailed information on the 3800 Printing Subsystems, Model 1 and 3, see:

- *Introducing the IBM 3800 Printing Subsystem and Its Programming*
- *Concepts of the IBM 3800 Printing Subsystem*
- *IBM 3800 Printing Subsystem Programmer's Guide*
- *IBM 3800 Printing Subsystem Model 3 Programmer's Guide: Compatibility*
- *Reference Manual for the IBM 3800 Printing Subsystem*
- *Reference Manual for the IBM 3800 Printing Subsystem Model 3.*

VM/SP supports the 3800 Model 1 and Model 3 printer as a dedicated device, as a virtual spooling device, and as a real spooling device.

Using the 3800 Printer as a Dedicated Device

A virtual machine configured to support a real 3800 printer can attach a 3800 printer for its exclusive use. VM/SP supports all of the facilities of the dedicated 3800 printer.

Using the 3800 Printer as Virtual Spooling Devices

VM/SP enables a user to create printer spool files on a virtual 3800 printer defined for his virtual machine. VM/SP provides full support for the copy modifications, forms overlay, character modifications, and multiple copy features. The user can specify the character arrangement tables, copy modifications, and forms control buffers used to print a file.

VM/SP makes these 3800 features available both to CMS users and to virtual machines running an operating system with full 3800 printer support. When a virtual machine running an operating system with full 3800 printer support issues commands that generate 3800 load commands, the virtual machine operating system passes these load commands and their associated data to CP. CP includes these load commands in the virtual spool file.

The CP SPOOL and CHANGE commands and the CMS SETPRT command specify 3800 control information. The class G QUERY command displays characteristics of the virtual printer including 3800 control information. It also displays control information associated with specific spool files.

Defining a Virtual 3800 Printer

To use the features of the 3800 printers as virtual spooling devices, the installation or user must define a virtual 3800 Model 1 or a virtual 3800 Model 3 for the user's virtual machine. With the class G DEFINE command a user can specify a virtual 3800 model number and its characteristics, including how many Writable Character Generation Modules (WCGM) the virtual printer has, and whether CP reflects all data checks to the virtual machine. (See the *VM/SP CP Command Reference* for details on the DEFINE command.)

The SPOOL control statement allows an installation to define a virtual 3800 Model 1 or a virtual 3800 Model 3 in the directory. (See the *VM/SP Planning Guide and Reference* for details on the SPOOL control statement.)

Using the SPOOL and CHANGE Commands

Use the SPOOL and CHANGE commands to specify 3800 control information. When a file is printed on a real 3800, necessary modules are located in the image library and loaded into the printer.

Five parameters on the SPOOL and CHANGE commands support the 3800 printers.

FLASH identifies the form overlay, if any, to be used when printing the file

CHARS names the character arrangement table to be used to print the file

MODIFY indicates the copy modification module, if any, to be used when printing the file

FCB specifies the name of the forms control buffer to be used for the file

COPY indicates the number of copies to be printed.

(See the *VM/SP CP Command Reference* for detailed command descriptions.)

Using the SETPRT Command

The CMS SETPRT command imbeds 3800 modules in a spool file. When the file prints on a real 3800 Model 1 or Model 3, these modules are loaded into the printer. The SETPRT command also allows a user to specify copy groups and the number of copies to print. (See the *VM/SP CMS Command Reference* for details on the SETPRT command.)

The TRC option of the PRINT command lets you print each line with a different character set. The TRC option causes the second byte in each line to select a character arrangement table.

Note that files created on a virtual 3800 can print on any real spooling device supported by VM/SP. However, if a file that was created on a virtual 3800 is printed on another real printer (for example, a 1403 or a 3211) all 3800 unique control information is ignored by the real printer. When files created on a 3800 Model 1 (or 3) and printed on the 3800 Model 3 (or 1, respectively) character set control information is ignored and the default specified by the operator on the START command is used.

While the 3800 printer allows print lines of 204 bytes, lines are truncated if printed on a real printer with a smaller maximum line length (for example, a 1403 or a 3211).

For CMS users running OS programs, coding the OPTCD J parameter of the FILEDEF command indicates to the QSAM PUT macro and the BSAM WRITE macro that each output line contains a TRC byte.

Using the 3800 Printer as a Real Spooling Device

Users of spool files can print their files on an IBM 3800 Model 1 or Model 3 Printing Subsystem. The copy modification modules, forms overlay, graphic character modification modules, and multiple copy features are fully supported.

The operator specifies control information for separator pages using the START command. The control information acts as the default for a user's print files if the user doesn't specify CHARS or FCB.

The GENIMAGE command constructs the 3800 modules— the character arrangement tables, graphic modifications, copy modifications, and forms control buffers—used by the 3800. The IMAGELIB and IMAGEMOD commands load this control information into the image library.

The NAME3800 macro instruction allows the system programmer to create the image library that contains the control information needed to print a spool file.

Specifying Printer Options

The START command includes parameters that enable the VM/SP operator to name the character arrangement table and the forms control buffer to be used for the separator page. The operator can also identify, via the FLASH operand of the START command, the forms overlay currently loaded in the 3800. In addition, the operator uses the IMAGE parameter to specify which image library is to be used for that 3800. Finally, by specifying the PURGE parameter, the operator can purge all spool files that cause errors when loaded into the 3800.

The class B QUERY command displays this 3800 printer information. See the *VM/SP CP Command Reference* for further information on the class B QUERY command and the START command.

The GENIMAGE Command

GENIMAGE The GENIMAGE command, which uses the OS/VS utility IEBIMAGE, creates library character sets, copy modifications, graphic modifications, and forms control buffers.

Note: Because of the change with pel density, customized 3800 Model 1 character sets are not interchangeable with the 3800 Model 3 character sets. Users may recode customized 3800 Model 1 character sets and build new modules using the IEBIMAGE utility program. The MVS Character Conversion Aid may also be used to convert existing customized character sets to the 3800 Model 3 pel density.

See *IBM 3800 Printing Subsystem Programmer's Guide* and *IBM 3800 Printing Subsystem Model 3 Programmer's Guide: Compatibility* for more information on coding 3800 modules.

Maintaining the 3800 Image Library

The 3800 image library contains library character sets (LCSs), graphic character modification modules (GRAPHMODs), character arrangement tables (CATs), forms control buffers (FCBs), and copy modification modules (COPYMODs). These modules are selected and loaded into the 3800 printer based on the parameters specified on the CP START command and control information associated with each spool file. The 3800 modules are created by the GENIMAGE command and are stored in the 3800 image library by the IMAGELIB and IMAGEMOD commands.

Both 3800 model 1 and 3800 model 3 modules may be stored in the same 3800 image library. The 3800 image library must contain the CAT and any GRAPHMODs referenced by that CAT for each character set that is to be used. For the 3800 model 3 only, any LCSs referenced by the CAT must also be loaded in the 3800 image library.

The GENIMAGE Service Program

The GENIMAGE command invokes the IEBIMAGE program to create the CATs, LCSs, GRAPHMODs, FCBs, and COPYMODs. These are created on a CMS disk with a filetype of TEXT.

GENIMAGE	[*fn*] [*ft*] [*fm*] [*sfn*] [*sft*] [*sfm*] [**SYSIN**] [**FILE**] [*****] [**SYSPRINT**] [**LISTING**] [**A1**]

fn
specifies the filename of the input control file. The name may be up to eight alphameric characters in length. If not specified, the default filename is SYSIN.

ft
specifies the filetype of the control file. Filetype may be up to eight alphameric characters in length. If not specified, the default filetype is FILE.

fm
specifies the filemode of the control file. Filemode may be up to two alphameric characters in length. If not specified, the default filemode is *.

sfn
specifies the filename into which the messages listing is placed. If not specified, the default file name is SYSPRINT.

sft
specifies the filetype for the messages listing. If not specified, the default file type is LISTING.

sfm
specifies the filemode for the messages listing. If not specified, the default file mode is A1.

Figure 23 on page 209 and Figure 24 on page 209 list the "starter set" of character sets supplied by IBM. They are supplied with a filename and filetype of XTB1xxxx TEXT.

AN	FM12	GU10	ODA	RN
AOA	FM15	GU12	ONA	SN
AOD	GF10	GU15	ONB	TN
AON	GF12	G11	PCAN	TU10
A11	GF15	HN	PCHN	T11
BOA	GN	H11	PN	XN
BON	GS10	KN1	P11	YN
DUMP	GS12	OAA	QN	2773
FM10	GS15	OAB	QNC	2774

Figure 23. VM-Supplied Character Arrangement Tables for the 3800 Model 1 and Model 3 Printers

AE10	EITR	GP12	LR12	SB12
BITR	ESTR	GR10	OB10	SI10
BRTR	GB10	GSC	OR10	SI12
CE10	GB12	GT10	PB12	SO12
CE12	GF10	GT12	PI12	SR12
CO10	GF12	GT15	PR10	ST10
CR10	GF15	GUC	PR12	ST12
DOTR	GFC	LB12	RT10	ST15
EBTR	GI12			

Figure 24. VM-Supplied Character Arrangement Tables for the 3800 Model 3 Only

Responses from GENIMAGE Command

Completion messages and codes are returned to the user in the file indicated by the sfn and sft operands of the GENIMAGE control statement. The GENIMAGE command produces two outputs: the listing file and a TEXT deck for each NAME statement in the input deck. Figure 25 contains the filename prefixes of these TEXT decks.

Module Type	3800 Model 1	3800 Model 3
LCS	LCS1	LCS2
CAT	XTB1	XTB1
GRAPHMOD	GRAF	GRF2
FCB	FCB3	FCB3
COPYMOD	MOD1	MOD1

Figure 25. Prefixes of TEXT Deck Names, Output by the GENIMAGE Command

GENIMAGE uses:

XTB1xxxx for a Character Arrangement Table

FCB3xxxx for a Forms Control Buffer
MOD1xxxx for a Copy Modification Module
GRAFxxxx for a Graphic Character Modification Module (3800 Model 1)
GRF2xxxx for a Graphic Character Modification Module (3800 Model 3)
LCS1xxxx for an Image Library Character Set Module (3800 Model 1)
LCS2xxxx for an Image Library Character Set Module (3800 Model 3)

For a description of control statements in SYSIN FILE, messages and codes in SYSPRINT LISTING, and the character sets listed above, see the following documents:

- *IBM 3800 Printing Subsystem Programmer's Guide* for the 3800 Model 1
- *IBM Printing Subsystem Model 3 Programmer's Guide: Compatability* for the the 3800 Model 3.

The IMAGELIB Service Routine

Before issuing the IMAGELIB command, you must create a control file whose filename is the same as the IMAGELIB you want to build, and whose filetype is CNTRL. The format of this file is one statement per TEXT deck you want included, with the names in columns one through eight.

IMAGELIB reads the control file containing the list of text images. The files are then loaded into the specified named system. The format of the IMAGELIB control statement follows.

IMAGELIB	*namesys*

namesys
specifies the named system that is being created or replaced.

The IMAGEMOD Command

The IMAGEMOD command allows an installation to make changes to 3800 named systems without the need for generating a completely new named system from scratch. It uses the current contents of the 3800 named system and adds, deletes, or replaces only those members that you change. The format of the IMAGEMOD command is as follows:

IMAGEMOD	{GEN / ADD} *libname*[*modname*[*modname*]...] {REP / DEL} *libname*[*modname*[*modname*]...] MAP *libname*[(Options)] Options: [TERM / PRINT / DISK]

GEN
specifies the creation from scratch of the new 3800 named system including the specified module names.

ADD
specifies the addition of the indicated module to an already existing 3800 named system.

REP
specifies the replacement of the indicated member with a new version of that member in an already existing 3800 named system.

DEL
specifies the deletion of an existing member from an already existing 3800 named system. The named system is compressed so that no blank space is left.

MAP
specifies the creation of a map of the 3800 named system on the user's terminal, virtual printer, or CMS disk. This map gives the member name, relative byte displacement in both decimal and hexadecimal, and the byte size of the member in both decimal and hexadecimal.

Note: The MAP function does not require the specification of any module names.

TERM
specifies that the map of the 3800 named system be written to the user's terminal.

PRINT
specifies that the map of the 3800 named system be written to the user's virtual printer.

DISK
specifies that the map of the 3800 named system be written to a CMS disk file called "libname MAP A5".

libname
specifies the name of the 3800 named system to be operated upon. Unless the GEN operand is specified, this named system must have been created with at least one member by the IMAGELIB command and must be defined by the installation administrators via the NAME3800 macro in the DMKSNT module.

modname
specifies the name of a module (CAT, LCS, GRAPHMOD, FCB, COPYMOD) that is to be added, replaced or deleted. One to 10 modules may be specified on one IMAGEMOD command.

Recovering from I/O Errors

Because the actual printing of lines on the page is slower than the output of lines from the processor, spool files are placed into a delayed purge queue to await printing by the 3800. Certain errors on the 3800 cause the print file to be requeued. Only when the maximum number of files are in the queue does the first one actually get purged. The size of the queue can be specified at system generation time via the DPMSIZE parameter on the RDEVICE macro instruction. DPMSIZE can have a maximum value of nine.

Chapter 21. 3088 Multichannel Control Unit

The IBM 3088 Multisystem Channel Communication Unit (MCU), an input/output device, interconnects as many as eight systems using block multiplexer channels. The 3088 Model 1 interconnects up to four systems, while the 3088 Model 2 interconnects up to eight systems.

With the 3088, you can use the PREPARE channel command to prevent attention interrupts on the side issuing the PREPARE command. For a description of 3088 channel command words, see "Channel Command Words" on page 215.

The 3088 is compatible with existing channel-to-channel usage. Also, 3088 support extends existing CTCA addressing and scheduling by:

- Allowing multiple unit addresses per control unit
- Implementing block multiplexer channel scheduling for both real and virtual CTCAs and 3088.

System Programmer Considerations

At system generation time, code parameters in the RDEVICE macro and the RCTLUNIT macro to define the 3088 to the control program. See the *VM/SP Planning Guide and Reference* for the format of these macros.

RDEVICE MACRO

When you code the RDEVICE macro, specify the address and device type. For example, to define a maximum of 32 sequential unit addresses at A00, code the RDEVICE macro as follows:

```
RDEVICE ADDRESS=(A00,32),DEVTYPE=3088
```

RCTLUNIT MACRO

When you code the RCTLUNIT macro, specify the address and the control unit type. Also, since the 3088 supports a maximum of 32 or 64 devices, specify the number of sequential unit addresses using the FEATURE=xxx-DEVICE operand. For example, if you want to generate 32 devices at channel address A00, code the RCTLUNIT macro as follows:

```
RCTLUNIT ADDRESS=A00,CUTYPE=3088,FEATURE=32-DEVICE
```

Special Directory Control Statement

The 3088 is a valid device for the SPECIAL directory control statement. For example, to specify a 3088 at virtual address A00, code the SPECIAL directory control statement as follows:

```
SPEcial A00 3088
```

Virtual 3088 Support

Use the class G DEFINE command to define a virtual 3088 device, with or without a real equivalent. The system simulates all functions of the real 3088, except for the online testing functions, for each virtual 3088 that you define. Define each virtual 3088 unit address with a single DEFINE command. Defining each virtual unit address is different from the dedicated 3088 support where you can define multiple unit addresses using a single RDEVICE macro. Refer to *VM Running Guest Operating Systems* for examples of virtual machine usage of channel-to-channel devices.

Command Use and 3088 Support

Support for the 3088 recognizes the 3088 as a valid device. Figure 26 outlines commands affected by 3088 support. See the *VM/SP CP Command Reference* for details on these commands.

Command	Class	3088 Support
DEFINE	G	The 3088 is a valid device type on this command. The control unit address for a CTCA and a 3088 need not end in zero. Once you define the control unit, you may define other virtual devices for the same CTCA or 3088.
ATTACH COUPLE DETACH QUERY	B G G, B B	The response to these commands is the same for channel-to-channel adapters (CTCAs) and 3088s.

Figure 26. CP commands and 3088 Support

Channel Command Words

In addition to the channel commands supported in System/360 and System/370 modes, the 3088 supports the following two channel commands:

PREPARE
Use the PREPARE channel command to receive a channel program without causing an attention interrupt to the side issuing the command.

SENSE ID
Use the SENSE ID channel command to transfer model and control unit identification to the system issuing the command.

Diagnostic Aids

3088 support offers online testing facilities, and messages and MNOTES as diagnostic aids when using the support. See *VM/SP System Messages and Codes* for the complete text of the messages.

Online Testing

The last address in the group of 32 or 64 addresses for each interface attached to the 3088 is available as a dedicated diagnostic unit address. The diagnostic unit address provides a communication path between diagnostic programs and the 3088 microprocessor for online testing. For example, a system attached to the 3088 may use the diagnostic unit address to read the 3088 logout and error information.

Messages and MNOTES To Support 3088 Devices

The system issues a message or MNOTE if:

- You try to define a 3088 for a unit address previously defined
- The virtual channel-to-channel device specified in the COUPLE command is busy on the receiving userid's virtual machine
- You try to couple a 3088 to a channel-to-channel adapter
- You specify a model on the RDEVICE macro.

Appendix. VM/SP Monitor Tape Format and Content

Each time a monitor call interrupt occurs, VM/SP Monitor receives control and collects data appropriate for the particular class and code of MONITOR CALL. (Or, for USER, PERFORM, or DASTAP classes, VM/SP Monitor gets control at periodic intervals to collect data.) The information is formatted into records that are collected sequentially in the order that each interrupt occurred. The tape data format is standard Variable Blocked (VB) format. The default tape drive density is used to write data. Maximum block and record lengths are 4096 bytes. The formats and contents of all the kinds of data records for the currently implemented classes and codes of MONITOR CALL are listed below.

All values described in the following records are binary unless otherwise noted:

[1] Indicates that the field is EBCDIC.

[2] Indicates that the field is in special timer format described below.

[3] See *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* for field format definition.

Header Record

Every data record is preceded by the following 12-byte header:

Data Item	Number of Bytes	DSECT Variable Name
Total bytes in record	2	MNHRECSZ
Zeroes (standard V format record)	2	
MONITOR CALL class number	1	MNHCLASS
MONITOR CALL code number	2	MNHCODE
Time of Day	5	MNHTOD

Note: Time of day occupies 2 fullwords in storage, with the rightmost 12 bits zeroes. The rightmost 2 bytes and the leftmost byte are ignored, giving 16-microsecond accuracy instead of 1-microsecond.

The first 4 bytes of this header are the standard variable-format record field.

Monitor Class Zero

Data Records

Class Zero - Codes for Tape Header, Trailer, and Data Suspension Records

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
97	Tape header record			
	CPU serial/model number	8	CPUID	MN097CPU
	Software version number[1]	8	DMKCPEID	MN097LEV
	Date of data collection session[1]	8	TOD clock	MN097DAT
	Time of data collection session[1]	8	TOD clock	MN097TIM
	Userid of monitor controller[1]	8	VMUSER	MN097UID
	CR8 mask of enabled classes	4	DMKPRGC8	MN097CR8
	Size of CP nucleus	4	Derived by CP	MN097NUC
	Size of Free/Fret pools	4	Derived by CP	MN097FSS
	Size of dynamic paging area	4	Derived by CP	MN097DPA
	Size of trace table	4	Derived by CP	MN097TTS
	Size of V=R area (if any)	4	Derived by CP	MN097VR
	CPU logical address	2	LPUADDR	MN097CPL
	APU logical address	2	LPAUDDRX	MN097APL
	Generated system mode	2	DMKSYSAP	MN097MOD
	Unused	2		
	PP map	8	DMKCPEPP	MN097CPP
98	Tape trailer record			
	Userid of user shutting down monitor[1]	8	VMUSER	MN098UID
99	Tape write suspension record			
	TOD at suspension[2]	5	---	MN099TOD
	Count of write suspensions	4	---	MN099CNT

[1] Indicates that the field is EBCDIC.

[2] Indicates that the field is in special timer format described below.

Class Zero - PERFORM

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
00	Interval statistics			
	Total main processor idle time[2]	8	IDLEWAIT	MN000WID
	Total main processor page wait[2]	8	PAGEWAIT	MN000WPG
	Total main processor time I/O wait[2]	8	IONTWAIT	MN000WIO
	Total main processor problem time[2]	8	PROBTIME	MN000PRB
	Total paging start I/Os	4	DMKPAGPS	MN000PSI
	Total page I/O requests	4	DMKPAGCC	MN000CPA
	Current page frames on free list	4	DMKPTRFN	MN000NFL
	Pages being written, due for free list	4	DMKPTRSW	MN000PSN
	Total pages flushed, but reclaimed	4	DMKPTRPR	MN000PRC
	Number of reserved pages	4	DMKPTRRC	MN000RPC
	Number of shared system pages	4	DMKPTRSC	MN000SPC
	Total number of times free list empty	4	DMKPTRF0	MN000FLF
	Total number of calls to DMKPTRFR	4	DMKPTRFC	MN000CPT
	Total pages stolen from in-queue users	4	DMKPTRSS	MN000SS
	Number of pages swapped from the flush list	4	DMKPTRFF	MN000PFF
	Number of pages examined in stealing pages	4	DMKPTRRF	MN000PRF
	Number of full scans done in stealing pages	4	DMKPTRCS	MN000PCS
	Total real external interrupts to main processor	4	DMKPSANX	MN000NXR
	Total calls to DMKPRVLG	4	DMKPRVNC	MN000CPR
	Total calls to DMKVIOEX	4	DMKVSICT	MN000CVI
	Total calls to CCWTRANS from DMKVIO	4	DMKVSICW	MN000CCW
	Total virtual interval timer interrupts reflected	4	DMKDSPIT	MN000ITI
	Total virtual CPU timer interrupts reflected	4	DMKDSPPT	MN000PTI
	Total virtual clock comparator interrupts reflected	4	DMKDSPCK	MN000CKI
	Total virtual SVC interrupts simulated by main processor	4	PSASVCCT	MN000CSV
	Total virtual program interrupts handled	4	DMKPRGCT	MN000CPG
	Total I/O interrupts handled	4	DMKIOSCT	MN000CIO
	Total calls to dispatch (main)	4	DMKDSPCC	MN000CDS
	Total fast reflects in dispatch	4	DMKDSPAC	MN000CDA
	Total dispatches for new PSW	4	DMKDSPBC	MN000CDB
	Total calls to schedule	4	DMKSCHCT	MN000CSC
	Count of virtual machine SSK simulated	4	DMKPRVEK	MN000EK
	Count of virtual machine ISK simulated	4	DMKPRVIK	MN000IK
	Count of virtual machine SSM simulated	4	DMKPRVMS	MN000MS
	Count of virtual machine LPSW simulated	4	DMKPRVLP	MN000LP
	Count of virtual machine DIAGNOSE instructions	4	DMKPRVDI	MN000DI
	Count of virtual machine SIO simulated	4	DMKVSISI	MN000SI

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
	Count of virtual machine SIOF simulated	4	DMKVSISF	MN000SF
	Count of virtual machine TIO simulated	4	DMKVSITI	MN000TI

[2] See *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* for field format definition.

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
	Count of virtual machine CLRIO simulated	4	DMKVSICI	MN000CI
	Count of virtual machine HIO simulated	4	DMKVSIHI	MN000HI
	Count of virtual machine HDV simulated	4	DMKVSIHD	MN000HD
	Count of virtual machine TCH simulated	4	DMKVSITC	MN000TC
	Count of virtual machine STNSM simulated	4	DMKPRVMN	MN000MN
	Count of virtual machine STOSM simulated	4	DMKPRVMO	MN000MO
	Count of virtual machine LRA simulated	4	DMKPRVLR	MN000LR
	Count of virtual machine STIDP simulated	4	DMKPRVCP	MN000CP
	Count of virtual machine STIDC simulated	4	DMKPRVCH	MN000CH
	Count of virtual machine SCK simulated	4	DMKPRVTE	MN000TE
	Count of virtual machine SCKC simulated	4	DMKPRVCE	MN000CE
	Count of virtual machine STCKC simulated	4	DMKPRVCT	MN000CT
	Count of virtual machine SPT simulated	4	DMKPRVPE	MN000PE
	Count of virtual machine STPT simulated	4	DMKPRVPT	MN000PT
	Count of virtual machine SPKA simulated	4	DMKPRVEP	MN000EP
	Count of virtual machine IPK simulated	4	DMKPRVIP	MN000IP
	Count of virtual machine PTLB simulated	4	DMKPRVPB	MN000PB
	Count of virtual machine RRB simulated	4	DMKPRVRR	MN000RR
	Count of virtual machine STCTL simulated	4	DMKPRVTC	MN000TCL
	Count of virtual machine LCTL simulated	4	DMKPRVLC	MN000LCL
	Count of virtual machine CS simulated	4	DMKPRVCS	MN000CS
	Count of virtual machine CDS simulated	4	DMKPRVCD	MN000CD
	Count of virtual machine diagnose disk I/O	4	DMKHVCDI	MN000HDI
	Number of users dialed to virtual machines	4	DMKSYSND	MN000NDU
	Number of users logged on	4	DMKSYSNM	MN000NAU
	Number of page reads by main processor	4	PGREAD	MN000PRD
	Number of page writes by main processor	4	PGWRITE	MN000PWR
	Number of system pageable pages	4	DMKDSPNP	MN000NPP
	Sum of working sets of in-queue users	4	DMKSCNPU	MN000SWS
	Number of users in interactive queue (Q1)	4	DMKSCHN1	MN000Q1N
	Number users in compute-bound queue (Q2)	4	DMKSCHN2	MN000Q2N

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
	Number of users eligible to enter Q1	2	DMKSCHW1	MN000Q1E
	Number of users eligible to enter Q2	2	DMKSCHW2	MN000Q2E
	Monitor sampling interval (seconds)	2	DMKPRGTI	MN000INT
	Count of cylinders allocated on primary paging device	2	ALOCUSED	MN000PPA
	Cylinder capacity of primary paging device	2	ALOCMAX	MN000PPC
	Reserved	2	---	MN0RSV1
	Count of mini IOB stack depletes	2	DMKIOSNM	MN000ISD
	Count of mini IOB enqueues	4		MN000GTM
	Count of mini IOB dequeues	4		MN000DQM
	Count of SIOs on alternate paths	4		MN000SWP
	Count of FREE/FRET extends	4	DMKFRENP	MN000EXT
	Count of FREE/FRET unextends	4		MN000NXT
	Count of attempts to split subpool	4		MN000ATT
	Count of SUBPOOL SPLITS	4		MN000CNT

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
01	Internal statistics for attached processor			
	Total attached processor idle wait time	8	IDLEWAIT	MN001WID
	Total attached processor page wait time	8	PAGEWAIT	MN001WPG
	Total attached processor I/O wait time	8	IONTWAIT	MN001WIO
	Total attached processor problem time	8	PROBTIME	MN001PRB
	Total real external interrupts for attached processor	4	DMKPSANX	MN001NXR
	Total SVCs reflected by attached processor	4	PSASVCCT	MN001CSV
	Page reads by attached processor	4	PGREAD	MN001PRD
	Page writes by attached processor	4	PGWRITE	MN001PWR
	Total time spin on system lock	4	DMKLOKSY+8	MN001SSY
	Number of spins on system lock	4	DMKLOKSY+12	MN001NSY
	Total time spin on DMKFRE lock	4	DMKLOKFR+8	MN001SFR
	Number of spins on DMKFRE lock	4	DMKLOKFR+12	MN001NFR
	Total time spin on RUNLIST lock	4	DMKLOKRL+8	MN001SRN
	Number of spins on RUNLIST lock	4	DMKLOKRL+12	MN001NRN
	Total time spin on timer request lock	4	DMKLOKTR+8	MN001STM
	Number of spins on timer request lock	4	DMKLOKTR+12	MN001NTM
	Total time spin on dispatcher queue lock	4	DMKLOKDS+8	MN001SDP
	Number of spins on dispatcher queue lock	4	DMKLOKDS+12	MN001NDP
	Number of times CPFRELK set	4		MN001NFL
	Number of times CPFRESW set	4		MN001NFS
	Number of times system lock deferred	4	LOKSYSCT	MN001NSD
	Number of times VMBLOK lock deferred	4	LOKVMCT	MN001NVD
	Number of DMKDSPRU entries	4		MN001NRU
	Total time spin on I/O lock	4	DMKLOKIO	MN001SIO
	Total no. spins for I/O lock	4	DMKLOKIO	MN001NIO
	Total time spin on RM lock	4	DMKLOKRM	MN001SRM
	Total no. spins for RM lock	4	DMKLOKRM	MN001NRM
	Number quiesce ems on IPL proc	4	DMKEMSCT	MN001NQ1
	Number quiesce ems on non-IPL proc	4	DMKEMSCT	MN001NQ2
	Number extend ems on IPL proc	4	DMKEMSCT	MN001NE1
	Number extend ems on non-IPL proc	4	DMKEMSCT	MN001NE2
	Number resume XC on IPL proc	4	DMKXCCTS	MN001NR1
	Number resume XC on non-IPL proc	4	DMKXCCTS	MN001NR2
	Number dispatch XC on IPL proc	4	DMKXCCTS	MN001ND1
	Number dispatch XC on non-IPL proc	4	DMKXCCTS	MN001N02
	Number dispatch XC on non-IPL proc	4	DMKXCCTS	MN001ND2
	Number wakeup XC on IPL proc	4	DMKXCCTS	MN001NW1
	Number wakeup XC on non-IPL proc	4	DMKXCCTS	MN001NW2

Monitor Class Zero

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
02	Average queue delay	4	DMKSCHQT	MN002SQT
	Average eligible list time	4	DMKSCHET	MN002SET
	Average utilization	4	DMKSCHFS	MN002SFS
	Average resident page request	4	DMKSCHAP	MN002SAP
	Average desired processor/page read	4	DMKSCHKA	MN002SKA
	Average processor overhead/page read	4	DMKSCHUC	MN002SUC
	Calculated paging bias	4	DMKSCHPB	MN002SPB
	Paging bias limit	4		
	Interactive bias	4	DMKSCHIB	MN002SIB
	Count of Q3 users	4	DMKSCHQ3	MN002SQ3
	Q1 in-queue count	8	VMQTOD	MN002Q11
	Q1 in-queue time	8	VMQELP	MN002Q12
	Q1 eligible list time	8	VMQWT	MN002Q13
	Q1 in-queue processor time	8	VMQCPU	MN002Q14
	Q1 estimated average pages per second	8	VMQPGS	MN002Q15
	Q1 count of queue drops	4	VMQCNT	MN002Q16
	Q1 in-queue page reads	4	VMQPRD	MN002Q17
	Q1 in-queue page steals	4	VMQSTL	MN002Q18
	Reserved	4	---	MN00RSV1
	Q2 in-queue count	8	VMQTOD	MN002Q21
	Q2 in-queue time	8	VMQELP	MN002Q22
	Q2 eligible list time	8	VMQWT	MN002Q23
	Q2 in-queue processor time	8	VMQCPU	MN002Q24
	Q2 estimated average pages per second	8	VMQPGS	MN002Q25
	Q2 count of queue drops	4	VMQCNT	MN002Q26
	Q2 in-queue page reads	4	VMQPRD	MN002Q27
	Q2 in-queue page steals	4	VMQSTL	MN002Q28
	Reserved	4	---	MN00RSV2

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
03	Number calls for migration	4	DMKSCHQ1	MN003CMG
	Times migration limit halved	4	DMKSCHQ1	MN003TLH
	Times limit was quartered	4	DMKSCHN1	MN003TLQ
	Times a user was selected	4	DMKSCHW1	MN003TUS
	Number migrations by command	4		MN003MBC
	Number calls resulting in migration	4		MN003CRM
	Number users moved	4		MN003NUM
	Number segments moved	4		MN003NSM
	Number pages moved	4		MN003NPM
	Number full disks moved	4		MN003NDM
	Calls to restore swaptable	4		MN003CSR
	Calls for swaptable migration	4		MN003CSM
	Number tables moved	4		MN003NTM
	Number tables restored	4		MN003NTR
	Calls to pseudo translator	4		MN003CPT
	Reserved	4		MN003RSV
	Total test protect ins simulated	4	DMKPRVTP	MN003CTP
	Total IPTE instructions simulated	4	DMKPIPTE	MN003CIP
	Number preferred FH pages available	4	DMKPGTDM	MN003CDM
	Number preferred MH pages available	4	DMKPGTDK	MN003CDK
	Number preferred MH pages allocated	4	DMKPGTPC	MN003CPC
	Limit of preferred MH pages	4	DMKPGTPL	MN003CPL
	% value for SET SRM MHFULL	4	DMKPGTPN	MN003CPN
	Unused	4		MN003CUN

Note: Privileged instructions simulated by the fast path simulation routines (DMKFSP) are not recorded.

Monitor Class One

Class One - RESPONSE

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
00	Read command sent to terminal			
	Userid	8	VMUSER	MN10XUID
	Line address	2		MN10XADD
01	Terminal output line			
	Userid	8	VMUSER	MN10XUID
	Line address	2		MN10YADD
	Byte count	2		MN10YCNT
	Line of data	Variable		MN10YIO
02	Edited terminal input line			
	Userid	8	VMUSER	MN10XUID
	Line address	2		MN10XADD
	Byte count	1		MN10YCNT
	Line of data[1]	Variable		MN10YIO
03	Sleep issued with time out			
	Userid	8	VMUSER	MN10XUID
	Line address	2		MN10XADD
04	Terminal logged on			
	Userid	8	VMUSER	MN10XUID
	Line address	2		MN10XADD
05	Terminal logged off			
	Userid	8	VMUSER	MN10XUID
	Line address	2		MN10XADD

Note that the line addresses for the 370X in NCP mode appear as the base address.

These records are created at the time that DMKQCN handles the console I/O request. This may reflect a slightly different time than that of the SIO or the I/O interrupt. If DMKQCN is called to write a line that is longer than Terminal line size, more than one MC is issued, resulting in more than one record. Input and output terminal data collected is limited to 128 bytes. Longer lines are truncated.

[1] Indicates that the field is EBCDIC.

Class Two - SCHEDULE

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
02	User dropped from run list			
	Userid[1]	8	VMUSER	MN20XUID
	Number of system pageable pages	4	DMKDSPNP	MN20XNPP
	Sum of working sets of in-queue users	4	DMKSCHPU	MN20XSWS
	Number users in interactive queue (Q1)	4	DMKSCHN1	MN20XQ1N
	Number users in compute-bound queue (Q2)	4	DMKSCHN2	MN20XQ2N
	Number of users eligible for Q1	2	DMKSCHW1	MN20XQ1E
	Number of users eligible for Q2	2	DMKSCHW2	MN20XQ2E
	User new projected working set size	2	VMWSPROJ	MN20XWSS
	Queue being dropped from	1	Q1DROP	MN20XQNM
	Processor address	1	---	MN20XPRC
	Accumulated user CP simulation time[2]	8	VMTTIME	MN20YTTI
	Accumulated user virtual time[2]	8	VMVTIME	MN20YVTI
	Eligible list priority	2	VMQPRIOR	MN204PRI
	Pages read while in queue	2	VMPGREAD	MN202PGR
	Sum of pages resident at all reads	2	VMPGRINQ	MN202APR
	Number pages referenced while in queue	2	---	MN202REF
	Current number of pages resident	2	VMPAGES	MN202RES
	Number of pages stolen while in queue	2	VMSTEALS	MN202PST
	User total virt non-spool device SIO count	4	VMIOCNT	MN202IOC
	User total virtual cards punched	4	VMPNCH	MN202PNC
	User total virtual lines printed	4	VMLINS	MN202LIN
	User total virtual cards read	4	VMCRDS	MN202CRD
	User last executed on this processor	1	VMLSTPRC	MN202LPR

[1] Indicates that the field is EBCDIC.

[2] See *VM/SP Data Areas and Control Block Logic Volume 1 (CP)* for field format definition.

Monitor Class Two

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
03	User added to run list			
	Userid	8	VMUSER	MN20XUID
	Number of system pageable pages	4	DMKDSPNP	MN20XNPP
	Sum of working sets of in-queue users	4	DMKSCHPU	MN20XSWS
	Number of users in interactive queue (Q1)	4	DMKSCHN1	MN20XQ1N
	Number users in compute-bound queue (Q2)	4	DMKSCHN2	MN20XQ2N
	Number of users eligible for Q1	2	DMKSCHW1	MN20XQ1E
	Number of users eligible for Q2	2	DMKSCHW2	MN20XQ2E
	User's projected working set size	2	VMWSPROJ	MN20XWSS
	Queue being added to	1	GPR 15	MN20XQNM
	Processor address (main or attached)	1	---	MN20XPRC
04	User added to eligible list			
	Userid	8	VMUSER	MN20XUID
	Number of system pageable pages	4	DMKDSPNP	MN20XNPP
	Sum of working sets of in-queue users	4	DMKSCHPU	MN20XSWS
	Number of users in interactive queue (Q1)	4	DMKSCHN1	MN20XQ1N
	Number users in compute-bound queue (Q2)	4	DMKSCHN2	MN20XQ2N
	Number of users eligible for Q1	2	DMKSCHW1	MN20XQ1E
	Number of users eligible for Q2	2	DMKSCHW2	MN20XQ2E
	User's projected working set size	2	VMWSPROJ	MN20XWSS
	Queue being added to	1	VMQ1	MN20XQNM
	Processor address (main or attached)	1	---	MN20XPRC
	Accumulated user CP simulation time	8	VMTTIME	MN20YTTI
	Accumulated user virtual time	8	VMVTIME	MN20YVTI
	Eligible list priority	2	VMEPRIOR	MN20YPRI

Class Four - USER

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
00	Interval user resource utilization statistics			
	Userid[1]	8	VMUSER	MN400UID
	Accumulated user CP simulation time	8	VMTTIME	MN400TTI
	Accumulated user virtual time	8	VMVTIME	MN400VTI
	Total page reads	4	VMPGREAD	MN400PGR
	Total page writes	4	VMPGWRIT	MN400PGW
	Total non-spooled I/O requests	4	VMIOCNT	MN400IOC
	Total cards punched	4	VMPNCH	MN400PNC
	Total lines printed	4	VMLINS	MN400LIN
	Total cards read	4	VMCRDS	MN400CRD
	User running status	1	VMRSTAT	MN400RST
	User dispatch status	1	VMDSTAT	MN400DST
	User operating status	1	VMOSTAT	MN400OST
	User queuing status	1	VMQSTAT	MN400QST
	User processing status	1	VMPSTAT	MN400PST
	User control status	1	VMESTAT	MN400EST
	User tracing control	1	VMTRCTL	MN400TST
	User message level	1	VMMLEVEL	MN400MLV
	User queue level	1	VMQLEVEL	MN400QLV
	User command level	1	VMCLEVEL	MN400CLV
	User timer level	1	VMTLEVEL	MN400TLV
	Interrupt pending summary	1	VMPEND	MN400PND
	User's externally assigned priority	1	VMUPRIOR	MN400UPR
	Reserved	1	---	MN4RSV1
	Current number of pages resident	2	VMPAGES	MN400RES
	Current working set size estimate	2	VMWSPROJ	MN400WSS
	Page frames allocated on drum	2	VMPDRUM	MN400PDR
	Page frames allocated on disk	2	VMPDISK	MN400PDK
	Monitor sampling interval (seconds)	2	DMKPRGTI	MN400INT
	User last executed on this processor	1	VMLSTPRC	MN400LPR
01	Total number of segment table origin (STO) steals	4	EXTSTOST	MN410SST
	Total number of page table steals	4	EXTUPTST	MN410PTS
	Address of high water mark	4	EXTHWMRK	MN410HWM
	Maximum number of STO blocks	1	EXTSTOMX	MN410NSB
	Actual number of STO blocks in use	1	EXTSTOCT	MN410SBU
	Reserved	2		MN410RSV

[1] Indicates that the field is EBCDIC.

Monitor Class Five

Class Five - INSTSIM

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
00	Start of PRIVOP simulation			
	Userid[1]	8	VMUSER	MN500UID
	The privileged instruction	4	VMINST	MN500INS
	Virtual storage address of PRIVOP	4	VMPSW	MN500VAD
	Total user CP simulation time at start of simulation	8	CPU timer	MN500OVH

Note: Privileged instructions simulated by the fast path simulation routines (DMKFSP) are not recorded.

[1] Indicates that the field is EBCDIC.

Class Six - DASTAP

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
00,01	Device activity data for all Tape and DASD devices			
	Number of device blocks recorded	2		MN600NUM
	For each device --			
	Device address	2	RDEVADDR + RCUADDR + RCHADDR	MN600ADD
	Type codes	2	RDEVTYPC	MN600TY
	Volume serial number[1]	6	RDEVSER	MN600SER
	Device accumulated I/O count	4	RDEVIOCT	MN600CNT

Note: The monitor code 0 record is collected when the MONITOR START TAPE command is entered. Thereafter, all DASTAP records are collected with a monitor code of 1.

[1] Indicates that the field is EBCDIC.

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
02	Number samples for interval IPL proc	2	MNCHSAM1	MN602SAM
	Number samples for interval non-IPL proc	2	MNCHSAM2	MN602SA2
	Device address	2	RDEVADD	MN602ADD
	Number times control unit busy	2	MNCBUSY	MN602CUB
	Number times device busy IPL proc	2	MNDVBSY	MN602DVB
	I/O tasks queued on control unit	2	RCUQCNT	MN602CUQ
	I/O tasks queued on device	1	RDEVQCNT	MN602DVQ
	Number times device busy non-IPL proc	1	MDVBSY2	MN602DV2
03	Channel busy counts IPL proc	32	MNCHDAT1	MN603CBI
	I/O tasks queued on channel IPL proc	32	RCHQCNT	MN603CQI
	Channel busy counts non-IPL proc	32	MCHDAT2	MN603CB2
	I/O tasks queued on channel non-IPL proc	32	RCHQCNT	MN603CQ2

Monitor Class Seven

Class Seven - SEEKS

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
00	DASD I/O request record			
	Userid[1]	8	VMUSER	MN700UID
	Device address		RDEVADDR + RCUADDR +	
		2	RCHADDR	MN700ADD
	Seek cylinder address	2	IOBCYL	MN700CYL
	Current arm position	2	RDEVCYL	MN700CCY
	Number of queued I/O tasks on device	1	RDEVQCNT	MN700QDV
	Number of queued I/O tasks on control unit	1	RCUQCNT	MN700QCU
	Number of queued I/O tasks on channel	1	RCHQCNT	MN700QCH
	Current seek direction	1	RDEVFLAG	MN700DIR
	Processor address	2	RCHPROC	MN700PRO

Note: Current seek direction value is:

- X'00' seeking to lower cylinder address
- X'01' seeking to higher cylinder address

[1] Indicates that the field is EBCDIC.

Class Eight - SYSPROF -- Additional data for system profile class

Monitor Code	Data Item	Number of Bytes	CP Variable Name	DSECT Variable Name
02	Additional data at add queue, drop queue times			
	Number of 4-byte device block counts which follow	2	---	MN802NUM
	For each device ...count of I/O's	4	RDEVIOCT	---
	After device counts ...			
	Current number of users logged on	4	DMKSYSNM	MN802NAU
	Total system page reads	4	PGREAD	MN802PGR
	Total system page writes	4	PGWRITE	MN802PGW
	Current number of pageable pages	4	DMKDSPNP	MN802NPP
	Total system idle time	8	IDLEWAIT	MN802WID
	Total system page wait time	8	PAGEWAIT	MN802WPG
	Total system I/O wait time	8	IONTWAIT	MN802WIO
	Total system problem time	8	PROBTIME	MN802PRB

Summary of Changes

This book contains material formerly presented in the *VM/SP System Programmer's Guide*, SC19-6203-3, Release 4. Figure 27 on page 236 illustrates the reorganization of the SPG and some other Release 4 books it made obsolete. Order the *VM/SP System Programmer's Guide* by pseudo-number. For:

Release 4, order ST00-1578

Release 3, order ST00-1352

Release 2, order SQ19-6203

Release 1, order ST19-6203.

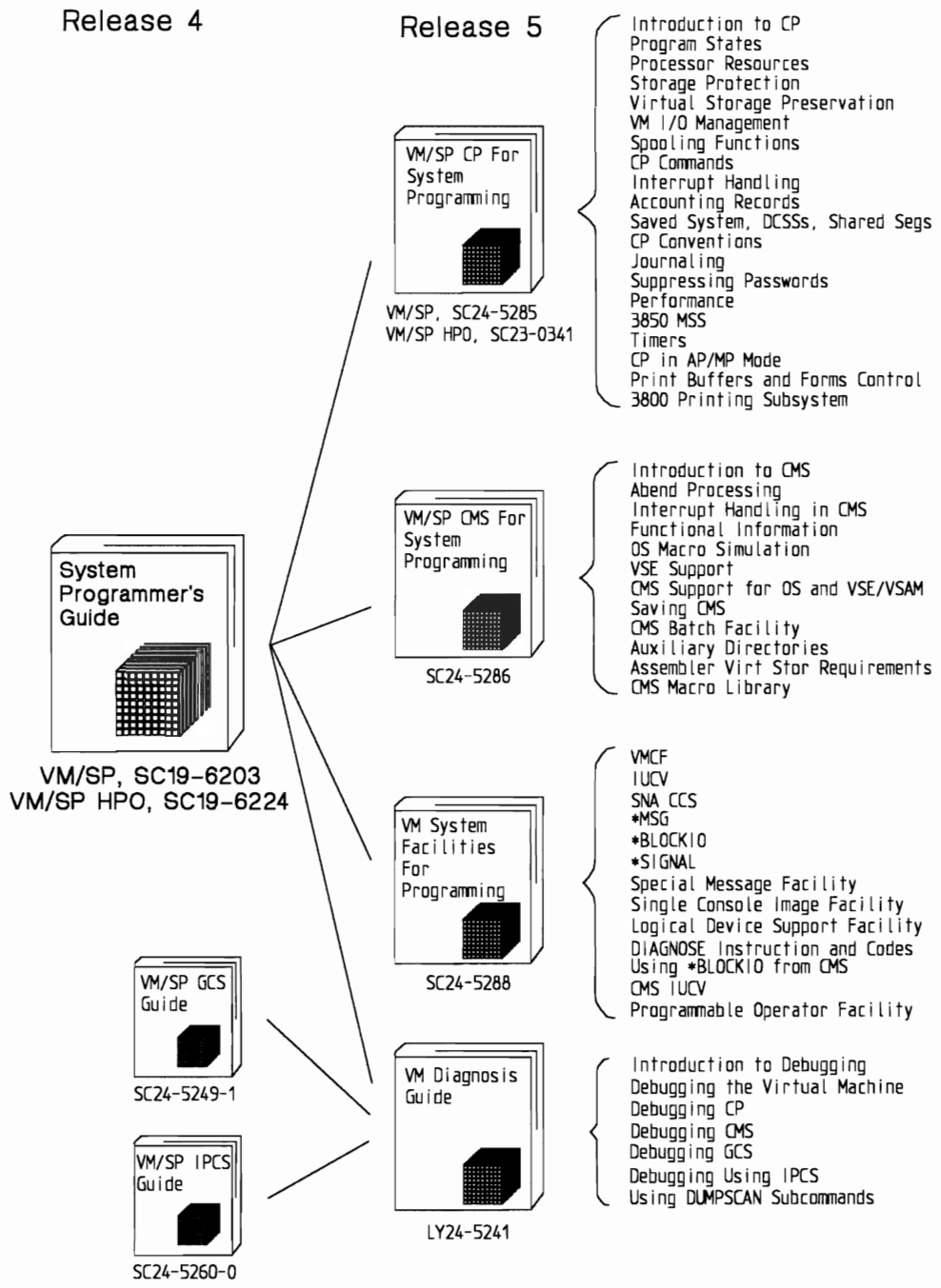


Figure 27. VM/SP System Programming Manuals for VM/SP Release 5.

Summary of Changes for VM/SP CP for System Programming

Summary of Changes
for SC24-5285
for VM/SP Release 5

Transparent Services Access Facility: Transparent Services Access Facility allows users to connect to and to communicate with local or remote virtual machines within a group of systems. With the Transparent Services Access Facility, a user can connect to a program by specifying a name that the program has made known, instead of specifying a userid and a nodeid. For more information, see *VM/SP Transparent Services Access Facility Reference.*

VM/SP Security SPE – Changes to Accounting Records: A new accounting record has been added, others have been changed, and the records have been renumbered.

Alternate Nucleus Support: Alternate nucleus support can improve system availability by allowing you to save multiple versions of your system nucleus. Then, the operator can IPL an alternate nucleus if the primary nucleus is damaged.

Access Verification Routines: Added support for access verification routines provides a standard interface to the Resource Access Control Facility (RACF)/VM support PRPQ.

Hardware Considerations: Some material has been transferred from the Release 4 *VM/SP Operator's Guide*:

- 3800 Image Library
- Device Support Facility
- DASD Dump Restore (DDR).

Numerous editorial and small technical changes have been made throughout.

Summary of Changes for the VM/SP System Programmer's Guide

The following summaries indicate the changes to the *VM/SP System Programmer's Guide*s for Release 4 and Release 3. Topics marked with an asterisk (*) are discussed in this book. To find where the other topics are discussed, see the references in the "Index" on page 255 to other VM and VM/SP publications.

Summary of Changes
for SC19-6203-3
for VM/SP Release 4

Group Control System (VM/SP GCS)

This new component of VM/SP is a virtual machine supervisor that provides simulated MVS services and supports a multitasking environment. For more information on the Group Control System (GCS), refer to the *VM/SP Group Control System Guide - (Discontinued)*, SC24-5249.

Signal System Service

This new CP system service allows virtual machines in a Virtual Machine Group to signal each other. The Signal System Service can only be used by virtual machines in a Virtual Machine Group.

****Saved System 8M Byte Limit Removal***

With the addition of this support, the SAVESYS, VMSAVE, and IPL functions have been enhanced to allow a page image copy of up to a 16M byte virtual machine to be saved and restored.

CP FRET Trap

The CP FRET Trap can be used as an aid in solving problems caused by improper use of CP storage and to solve many storage overlay problems.

VMDUMP Enhancements

DIAGNOSE Code X'94' is available to allow a virtual machine to request dumping of its virtual storage. Also, the three address range restriction has been removed from the VMDUMP command.

DIAGNOSE Code X'98'

Using DIAGNOSE Code X'98', a virtual machine can lock and unlock virtual pages, and execute its own real channel programs.

The Programmable Operator Facility

The Programmable Operator Facility has been enhanced to support distributed operations in an SNA network through an interface, the Programmable Operator/NCCF Message Exchange (PMX), with the Network Communications Control Facility (NCCF). The VM/SP Release 4 programmable operator:

- Allows an NCCF operator to be identified to the programmable operator so that any messages intended for the logical operator may be routed to that NCCF operator.
- Allows an NCCF operator to issue programmable operator commands and receive responses.
- Provides the LGLOPR command for assigning, releasing and replacing the logical operator during operation.

CPTRAP Enhancements

CPTRAP is a major service aid used in problem determination. Enhancements to the CPTRAP command provide two additional functions, GROUPID and WRAP, and one additional entry type, X'3D'.

Enhancements to TRAPRED make reviewing the trap data easier by providing more selectivity for X'3D', X'3E', and X'3F' entries and by providing a way to display formatted output of the trapped data.

Information on CPTRAP has been rewritten and reorganized for ease-of-use. It has also been moved to the Part 3, the debugging section, since it is a debugging tool.

Interactive Problem Control System (VM/SP IPCS)

VM/SP Release 4 has been enhanced to include IPCS as a component of VM/SP. VM/SP IPCS is equivalent to the VM/Interactive Problem Control System Extension (VM/IPCS/E) Licensed Program Product (5748-SA1).

Inter-User Communications Vehicle (IUCV) Enhancements

IUCV now supports the movement of data on the SEND, RECEIVE, and REPLY functions from discontiguous buffers. The modified IUCV macro handles the new BUFLIST= parameter on SEND and RECEIVE functions and the new ANSLIST= parameter on the SEND and REPLY functions.

****Expansion of User Classes***

The DIRECT command has been enhanced and the OVERRIDE command has been added to provide the user with more than the seven IBM defined user classes. You can now choose from 32 user classes, A - Z, and 1 - 6.

****Remote Spooling Communications Subsystem Networking Version 2***

With the release of the Remote Spooling Communications Subsystem Networking Version 2 Program Product (5664-188), any reference to RSCS in this manual applies to RSCS Version 2. Information pertaining to RSCS can be found in the *VM/SP Remote Spooling Communications Subsystem Version 2 General Information*, GH24-5055.

Miscellaneous changes

IOCP Support Enhancements

This support adds new MSSF command words to DIAGNOSE code X'80'.

Integration of Functional Enhancements to VM/SP Release 3

Information has been added to support:

- The 3290 Information Panel
- The 3370 Direct Access Storage Model
- The 4248 Printer
- The 4361 Model Groups 3, 4, and 5 Processor
- The 4381 Model Groups 1 and 2 Processor
- *VM/SP 3800 Model 3 Compatibility Support

  Compatibility support allows VM/SP users to access the 3800 Model 3 Printing Subsystem. Existing programs designed to produce 3800 Model 1 printer output may produce output for the 3800 Model 3 printer with little or no program change. Use of this support provides improved print quality (240 x 240 pel resolution) and the addition of a 10 lines-per-inch (LPI) vertical space option.

DIAGNOSE Code X'8C'

DIAGNOSE code X'8C' has been enhanced to allow a user to access all of the data returned by CP's WRITE STRUCTURED FIELD QUERY.

DMKFRE/DMKFRT Split

The module DMKFRE has been split into two modules, DMKFRE and DMKFRT. DMKFRE handles all requests for free storage as well as calls to DMKFRET to release free storage. DMKFRT handles all requests to return free storage that cannot be handled by the microcoded CP assist FRET function.

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

Summary of Changes
for SC19-6203-2
for VM/SP Release 3

Programmable Operator Facility

Several enhancements to the programmable operator facility added are:

- Message routing with nicknames
- Remote node availability
- Enhanced text comparison
- EXEC action routines
- LOG recording and error handling

PER

Problem determination capability is greatly extended and enhanced by the new CP command, PER.

DASD Block I/O System Service

The DASD Block I/O System Service allows a virtual machine fast, device-independent asynchronous access to fixed size blocks on CMS formatted virtual DASD I/O devices.

IUCV

Inter-User Communication Vehicle (IUCV) extensions provide:

- SEND and REPLY extensions
- An extended mask capability for control interrupts
- An expanded trace capability to record all IUCV operations
- A macro option to initialize the parameter list
- Support for the DASD block I/O system service.

**The IBM 3088 Multisystem Communications Unit*

The IBM 3088 Multisystem Communications Unit interconnects multiple systems using block multiplexer channels. The 3088 uses an unshared subchannel for each unique address and is fully compatible with existing channel-to-channel adapter protocol.

CMS IUCV support

Support for IUCV communication has been introduced into CMS. This support allows multiple programs within a virtual machine to use IUCV functions. Included is the ability to initialize a CMS machine for IUCV communication and to invoke IUCV functions via new CMS macros. These macros also allow the user to specify path-specific exits for IUCV external interrupts.

CMS abend exits

A general CMS abnormal exit capability is provided so that user programs may specify the address of a routine to get control before CMS abend recovery begins. An exit is established and cleared through a new CMS macro.

Enhanced immediate command support

The immediate command capability of CMS is extended by allowing users to define their own immediate commands.

Enhanced VSAM support

CMS supports VSE/VSAM Release 3, which includes significant enhancements designed to improve catalog reliability and integrity while providing additional serviceability and usability. VSE/VSAM Release 2 is not supported.

Miscellaneous

Changes to the DIAGNOSE code X'00' interface provide the time zone differential from Greenwich Mean Time.

DIAGNOSE code X'8C' allows a virtual machine to access device dependent information without having to issue a WRITE STRUCTURE FIELD QUERY REPLY.

CMSSEG has been eliminated and the code was merged into the CMS Nucleus.

The Remote Spooling Communications Subsystem (RSCS) section of this manual has been removed as it pertained to RSCS as a component of VM/370. Now, any reference to RSCS in this manual applies to the RSCS Networking Programming Product, and information can be found in the *VM/SP Remote Spooling Communications Subsystem Networking Program Reference and Operations Manual*, SH24-5005.

A newly added appendix lists and describes the CMS macros applicable to VM/SP.

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

Glossary

This glossary defines terms and abbreviations related to VM/SP. It is intended for readers of the *VM/SP CP for System Programming*. Therefore, some terms already defined in the *VM/SP Library Guide, Glossary, and Master Index*, SC19-6207, do not appear here or may be defined slightly differently. You may also want to refer to the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*

A

Advanced Program-to-Program Communication/VM (APPC/VM). An application program interface for communicating between two virtual machines that is mappable to the SNA LU 6.2 APPC interface and is based on IUCV functions. Along with the TSAF virtual machine, APPC/VM provides this communication within a single system and throughout a collection of systems.

AP/MP mode. A mode of VM/SP used when running in an attached processor or multiprocessor system.

APPC/VM. Advanced Program-to-Program Communications/VM

attached processor. A processor with no I/O capability. An attached processor is always linked to the processor initialized for I/O handling.

auxiliary storage. Data storage other than main storage; in VM/SP, auxiliary storage is usually a direct access device.

B

basic control (BC) mode. A mode in which a virtual machine resumes execution after an I/O interrupt, a page fault, or a DIAGNOSE code X'18'.

C

CAW. channel address word

CCW. channel command word

channel address word (CAW). An area in storage that specifies the location in main storage at which a channel program begins.

channel command word (CCW). A doubleword at the location in main storage specified by the channel address word. One or more CCWs make up the channel program that directs data channel operations.

channel status word (CSW). An area in storage that provides information about the termination of input/output operations.

Channel-to-Channel Adapter (CTCA). A hardware device that can be used to connect two channels on the same computing system or on different systems.

CKD. count-key-data

concurrently. Concerning a mode of operation that includes doing work on two or more activities within a given (short) interval of time.

CMS. refers to the VM/370 Conversational Monitor System component enhanced by the functions included in the VM/SP package.

CMS system disk. The virtual disk (S-disk) that contains the CMS nucleus and the disk-resident CMS commands. The CMS system disk can have extensions, usually the Y-disk.

count-key-data (CKD). A kind of disk storage device formatted in variable size records consisting of a count field, usually followed by a key field, followed by data fields. The count field contains the cylinder number, head number, and the record number of the of the record and the length of the data. The key field contains the record's key (search argument).

CP. refers to the VM/370 Control Program component enhanced by the functions included in the VM/SP package.

CSW. channel status word

CTCA. channel-to-channel adapter

D

DAT. dynamic address translation

DCSS. discontiguous saved segments

deadline priority. A number that is used to sort a virtual machine into the eligible lists and the run list. It is based upon an estimate of when the virtual machine should complete its next queue slice.

directory. For VM/SP, a CP disk file that defines each virtual machine's normal configuration: the userid, password, normal and maximum allowable virtual storage, CP command privilege class or classes allowed, dispatching priority, logical editing symbols to be used, account number, and CP options desired.

discontiguous saved segments (DCSS). Areas of virtual storage outside the address range of a virtual machine. These segments contain read-only or reentrant code and can be connected to a virtual machine's address space. See *shared segment.*

discontiguous segment. Synonymous with *discontiguous saved segment.*

dispatch list. See *run list*

dispatcher. The program in CP that places virtual machines or CP tasks into execution. The dispatcher selects the next virtual machine to run and prepares the virtual machine for problem state execution.

dispatch request queue. A queue of executable CP tasks, I/O tasks, and timer requests that are ready to be dispatched.

dispatcher/scheduler favoring scheme. A set of criteria used by the dispatcher and scheduler to create a bias in favor of queue 1 (Q1) users. Q1 users are usually highly interactive users.

DPA. dynamic paging area

dyadic. A system having two processors that cannot be configured into two independent uniprocessors that use separate control programs. For example, the 3081 Processor Complex contains two processing units that share central storage.

dynamic address translation. In System/370 virtual storage systems, the change of a virtual address to a real storage address during execution of an instruction.

dynamic paging area (DPA). An area of real storage that CP uses for virtual machine pages and pageable CP modules.

E

extended control (EC) mode. Extended control mode, a System/370 mode for formatting and use of control and status information. Contrast with "basic control (BC) mode."

eligible list. One of two queues of virtual machines that are waiting to get into the run list. They are runnable but cannot fit into the run list because of the current system load.

exec. refers to programs with file type EXEC, using the System Product Interpreter (REXX), EXEC 2, or CMS EXEC languages.

F

FBA. Fixed-block architecture.

Fixed-Block Architecture (FBA). Those DASD devices whose architecture uses fixed blocks or records of 512 bytes.

flush list. A set of pages available to replenish the free list.

free list. A list maintained by CP that points to a set of pages that can be allocated to satisfy both virtual machine and system page requests.

G

global system lock. A defer lock that provides system integrity for AP and MP support of command processing and code executed via IOBLOK, TRQBLOK, or CPEXBLOK.

guest virtual machine. A virtual machine in which an operating system is running.

I

in-queue virtual machine. A virtual machine on the run list waiting to be dispatched.

interactive. (1) An application in which each user entry calls forth a response from a system or program. (2) The classification given to a virtual machine depending on this virtual machine's processing characteristics. When a virtual machine uses less than its allocated queue slice because of terminal I/O, the virtual machine is classified as being interactive. See also non-interactive.

IPCS. refers to the VM/370 Interactive Problem Control System component enhanced by the functions included in the VM/SP package.

The IPCS component of VM/SP replaces the unmodified VM/370 interactive problem control system. Details of this component are found in the *VM/SP Interactive Problem Control System Guide*, SC24-5260.

Inter-User Communication Vehicle (IUCV). A VM/SP generalized CP interface that aids the transfer of messages either among virtual machines or between CP and a virtual machine.

IUCV. Inter-User Communication Vehicle.

L

log on. The procedure by which a user begins a terminal session.

log off. The procedure by which a user ends a terminal session.

M

MIH. Missing interrupt handler

minidisk. Synonym for virtual disk.

missing interrupt handler (MIH). A facility of VM/SP that detects incomplete I/O conditions by monitoring I/O activity. It also tries to correct incomplete I/O conditions without operator intervention.

MSSF. Monitoring and service support facility

N

named system. A collection of saved pages a user can IPL or load by name.

native mode. A mode in which an operating system is run stand-alone on the real machine instead of under VM/SP.

noninteractive. The classification given to a virtual machine depending on this virtual machine's processing characteristics. When a virtual machine usually uses all its allocated queue slice, it is classified as being noninteractive or compute bound. See also interactive.

non-resident pages. Pages whose contents are on DASD but not in real storage. A page is considered nonresident when an attempt to load its real address returns a nonzero condition code.

P

page. A 4K byte-long block of data that has a virtual address and can be transferred between real storage and virtual storage.

page frame. An area of real storage that can store a page.

page table. A table in CP that indicates whether a page is in real storage and matches virtual addresses with real storage addresses.

preferred paging area. A special area of auxiliary storage where frequently used pages are paged out. It provides high speed paging.

prefix storage area (PSA). a page zero of real storage that contains machine-used data areas and CP global data.

program status word (PSW). An area in storage used to indicate the order in which instructions are executed, and to hold and indicate the status of the computer system. Synonymous with processor status word.

projected working set. An estimate of the number of pages of real storage that must be allocated to an in-queue virtual machine if it is to avoid excessive paging. It is used to determine whether the virtual machine can be added to the run list from an eligible list.

PSA. Prefix storage area.

PSW. Program status word, or processor status word.

Q

queue-add. The action by the system scheduler, DMKSCH, of placing a runnable virtual machine on the run list.

queue-drop. The action by the system scheduler, DMKSCH, of removing a virtual machine from the run list.

queue slice. The maximum amount of time that a virtual machine may stay in the run list.

R

real machine. The actual processor, channels, storage, and I/O devices required for operation of VM/SP.

RSCS. unless otherwise noted, refers to the Remote Spooling Communications Subsystem Networking Version 2 Program Product (5664-188).

When you install and use VM/SP in conjunction with the VM/370 Release 6 System Control Program (SCP), it becomes a functional operating system that provides extended features to the Control Program (CP) and Conversational Monitor System (CMS) components of VM/370 Release 6. VM/SP adds *no* additional functions to the Remote Spooling Communications Subsystem (RSCS) component of VM/370. However, you can appreciably expand the capabilities of this component in a VM/SP system by installing RSCS Networking Version 2 (5664-188).

run list. A queue of virtual machines that are receiving for processor resources. Virtual machines take turns being dispatched for short periods of time (time slices) until they either complete a queue slice or go into a long wait state. Virtual machines in the run list can be briefly non-runnable—for instance, waiting for a page swap— without being dropped from the run list. These virtual machines in the run list are sorted by deadline priority.

S

S-disk. See CMS system disk.

segment. A contiguous 64K area of virtual storage (not necessarily contiguous in real storage) that is allocated to virtual machine or CP.

segment table. A table used in dynamic address translation to control user access to virtual storage segments. Each entry indicates the length, location, and availability of a corresponding page table.

shadow page table. A table that maps real storage allocations (first level storage) to a virtual machine's virtual storage (third level storage) for use by the real machine in its paging operations.

shared segment. A 64K segment of storage within a *saved system*or a *discontiguous saved segment.* Read-only and reentrant code in the segment can be shared by multiple virtual machines.

spool, spooled, spooling. Relates to the reading of input data streams and the writing of output data streams on auxiliary storage devices.

standalone dump. A program used to print the contents of storage that runs in a virtual machine not under control of an operating system such as CMS.

System/370. applies to the 4300 and 303X series of processors.

T

time sharing. Sharing of computer time and resources.

Transparent Services Access Facility (TSAF). A facility that lets users connect to and communicate with local or remote virtual machines within a collection of systems. With Transparent Services Access Facility, a user can connect to a program by specifying a name that the program has made known, instead of specifying a userid and nodeid.

TSAF. Transparent Services Access Facility

V

virtual address. An address that refers to virtual storage or a virtual I/O device address. It must, therefore, be translated into a real storage or I/O device address when it is used.

virtual disk. A logical subdivision (or all) of a physical disk storage device that has its own address, consecutive storage space for data, and an index or description of the stored data so that the data can be accessed. A virtual disk is also called a minidisk.

virtual machine. A functional simulation of a computer and its associated devices.

virtual machine assist (VMA). A hardware feature available on certain VM/SP-supported processors that causes a significant reduction in the real supervisor state time used to control the operation of virtual machine systems such as VSE, DOS/VS, and OS/VS and to a lesser extent CMS, DOS, and OS when executing under VM/SP.

virtual storage. Storage space that can be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.

VM/VCNA. VM/VTAM Communication Network Application

VM/VTAM Communication Network Application (VM/VCNA). A VTAM application which allows a SNA terminal user to logon to VM/SP though OS/VS1 or VSE.

VSCS. VTAM SNA Console Support component

VSE. refers to the combination of the DOS/VSE system control program and the VSE/Advanced Functions program product. "DOS," in certain cases, is still used as a generic term. For example, disk packs initialized for use with VSE or any predecessor DOS or DOS/VSE system may be referred to as DOS disks.

VM/SP. refers to the VM/SP program package when you use it in conjunction with VM/370 Release 6.

VTAM SNA Console Support component (VSCS). A VTAM application which allows a SNA terminal user to logon to VM/SP though the Group Control System (GCS) facility.

Y

Y-disk. An extension of the CMS system disk.

Numerics

2305. refers to IBM 2305 Fixed Head Storage, Models 1 and 2.

2741. refers to the IBM 2741 and the IBM 3767, unless otherwise specified.

3066. refers to the IBM 3066 System Console.

3081. refers to the IBM 3081 Processor Unit model D16.

3088. refers to the IBM 3088 Multisystem Communications Unit (MCU) Models 1 and 2.

3262. refers to the IBM 3262 Printer, Models 1, 5, and 11. 3262 Models 3 and 13 are supported remotely as 3287 printers.

3270. refers to a series of display devices, namely, the IBM 3275, 3276 (referred to as a Controller Display Station), 3277, 3278, and 3279 Display Stations, and the 3290 Information Panel. A specific device type is used only when a distinction is required between device types.

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

3285. or 3286 printer references also pertain to the IBM 3287, 3288, and 3289 printers, unless otherwise noted.

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 compatibility mode.

3340. refers to the IBM 3340 Direct Access Storage Facility and the 3344 Direct Access Storage.

3350. refers to the IBM 3350 Direct Access Storage Device when used in native mode.

3370. refers to the IBM 3370 Direct Access Storage Model.

3375. refers to the IBM 3375 Direct Access Device.

3380. refers to the IBM 3380 Direct Access Storage. The Speed Matching Buffer Feature (No. 6550) for the 3380 supports the use of extended count-key-data channel programs.

3422. refers to the IBM 3422 Magnetic Tape Subsystem.

3430. refers to the IBM 3430 Magnetic Tape Subsystem.

3480. refers to the IBM 3480 Magnetic Tape Subsystem.

370X. refers to IBM 3704 and 3705 Communications Controllers.

3705. refers to the 3705 I and the 3705 II unless otherwise noted.

3800. refers to the IBM 3800 Printing Subsystems, Models 1, 3, and 8. A specific device type is used only when a distinction is required between device types. References to the 3800 Model 3 apply to both Models 3 and 8 unless otherwise explicitly stated. The IBM 3800 Model 8 is available only in selected world trade countries.

4245. refers to the IBM 4245 Line Printer.

4248. refers to the IBM 4248 Printer.

4250. refers to the IBM 4250 Printer.

4361. refers to the IBM 4361

4381. refers to the IBM 4381

Bibliography

Here is a list of IBM books that can help you use your system. If you don't see the book you want in this list, check the *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001.

Fundamentals

IBM System/360 Principles of Operation, GA22-6821

IBM System/370 Principles of Operation, GA22-7000.

Books About VM/SP

The *VM/SP Library Guide, Glossary, and Master Index*, GC19-6207 describes all the VM/SP books and contains an expanded glossary and master index to all the books in the VM/SP library.

Virtual Machine/System Product:

General Information, GC20-1838

Introduction, GC19-6200

CMS Command Reference, SC19-6209

CMS User's Guide, SC19-6210

Installation Guide, SC24-5237

System Messages and Codes, SC19-6204

OLTSEP and Error Recording Guide, SC19-6205

Terminal Reference, SC19-6206

Library Guide, Glossary, and Master Index, SC19-6207

Operator's Guide, SC19-6202

EXEC 2 Reference, SC24-5219

System Product Editor User's Guide, SC24-5220

System Product Editor Command and Macro Reference, SC24-5221

System Product Interpreter User's Guide, SC24-5238

System Product Interpreter Reference, SC24-5239

CMS for System Programming, SC24-5286

Transparent Services Access Facility Reference, SC24-5287

Data Areas and Control Block Logic,

Volume 1 Control Program (CP), LY24-5220

Volume 2 Conversational Monitor System (CMS), LY24-5221

System Logic and Problem Determination Guide,

Volume 1 Control Program (CP), LY20-0892

Volume 2 Conversational Monitor System (CMS), LY20-0893

Vitual Machine:

System Facilities for Programming, SC24-5288

Diagnosis Guide, LY24-5241

Running Guest Operating Systems, SC19-6212

Monitor Analysis Program (VMMAP) General Information, GC-34-2164

Other Publications

Device Support Facilities User's Guide and Reference (Current Release), GC35-0033

IBM 2821 Control Unit Component Description, GA24-3312

3203 Component Description and Operator's Guide, GS33-1515

IBM 3203 Model 5 Component Description and Operator's Guide, GA33-1529

IBM 3203 Component Description and Operator's Guide, GA33-1515

IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide, GA24-3543

IBM 3262 Printers 1 and 11 Component Description, GA24-3733

IBM 3289 Line Printer Model 4 Component Description, GA27-3177

Introducing the IBM 3800 Printing Subsystem and Its Programming, GC26-3829

Concepts of the IBM 3800 Printing Subsystem, GC20-1775

IBM 3800 Printing Subsystem Programmer's Guide, GC26-3846

IBM 3800 Printing Subsystem Model 3 Programmer's Guide: Compatibility, SH35-0051

Reference Manual for the IBM 3800 Printing Subsystem, GA26-1635

Reference Manual for the IBM 3800 Printing Subsystem Model 3, GA32-0050

IBM 3270 Information Display System Library User's Guide, GA23-0058

3704 and 3705 Communications Controllers

Introduction to the IBM 3704 and 3705 Communications Controllers, GA27-3051

IBM 3704 and 3705 Communications Controllers Operator's Guide, GA27-3055

IBM 3704 Control Panel Guide, GA27-3086

IBM 3705 Control Panel Guide, GA27-3087

IBM 3704 and 3705 Control Program Generation and Utilities Guide and Reference Manual (OS/VS TCAM Levels 5 and 6 SVS - 5742-017) SCP 5742, 5744-AN1/BA2, 5747-AG1/AJ2, GC30-3008 in VS1; VS2 Rel 1.6, 1.7, 2, SCP 5744-BA1, GC30-3007

IBM 3704 and 3705 Control Program Generation and Utilities Guide and Reference Manual (TCAM 10 SVS - 5742-017) SCP 5742, 5744-AN1/BA2, 5747-AG1/AJ2, GC30-3008

IBM 3725 Communication Controller Operator's Guide, GA33-0014

IBM 3725 Operator Console Reference and Problem Analysis Guide, GA33-0015

IBM Virtual Machine Facility/370: Performance/Monitor Analysis Program, SB21-2101

ACF/VTAM

VTAM General Information (for VM), GC30-3246

Network Program Products Planning, SC23-0110

VTAM Installation and Resource Definition, SC23-0111

VTAM Customization, SC23-0112

VTAM Operation, SC23-0113.

EREP

Environmental Recording Editing and Printing (EREP) Program, GC28-1178

Environmental Recording Editing and Printing (EREP) Program User's Guide and Reference, GC28-1378

VM/SP Remote Spooling Communications Subsystem Networking (RSCS Networking) Version 2

Remote Spooling Communications Subsystem Networking Version 2

Planning and Installation, SH24-5057

Operation and Use, SH24-5058

Diagnosis Reference, LY24-5228

Resource Access Control Facility (RACF)

Resource Access Control Facility (RACF)

General Information Manual, GC28-0722

Command Language Reference, SC28-0733

Security Administrator's Guide, SC28-1340

Auditor's Guide, SC28-1342

Messages and Codes, SC38-1014

System Programming Library: Resource Access Control Facility (RACF), SC-28-1343

IBM 3850 Mass Storage System (MSS)

3850 Mass Storage System (MSS)

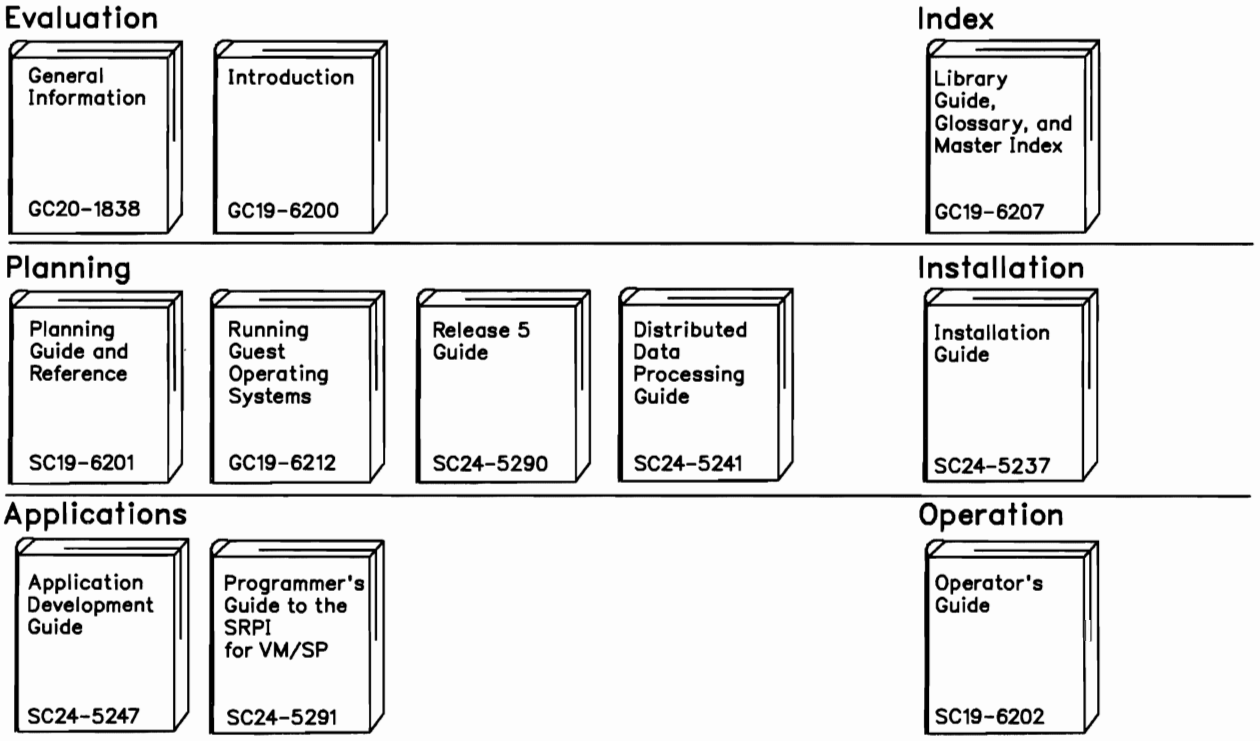
Introduction and Preinstallation Planning, GA32-0038

Principles of Operation: Theory, GA32-0035

Principles of Operation: Reference, GA32-0036

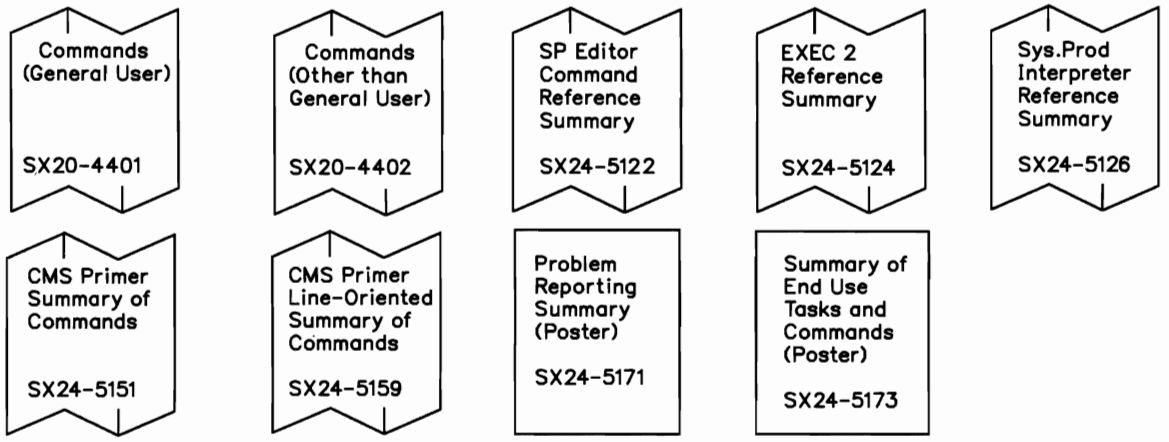
If you use the IBM 3767 Communication Terminal as a virtual machine console, the *IBM 3767 Operator's Guide*, GA18-2000, may also be helpful.

The VM/SP Library (Part 1 of 3)



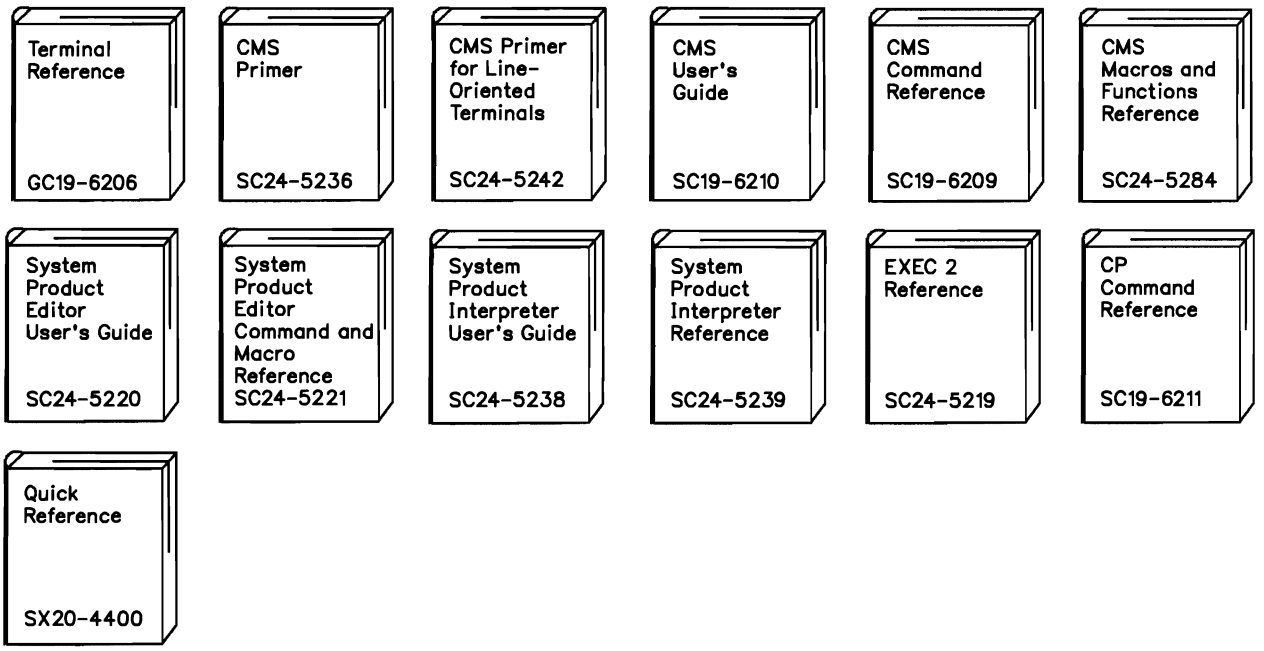
Reference Summaries

To order all of the Reference Summaries, use order number SBOF-3242

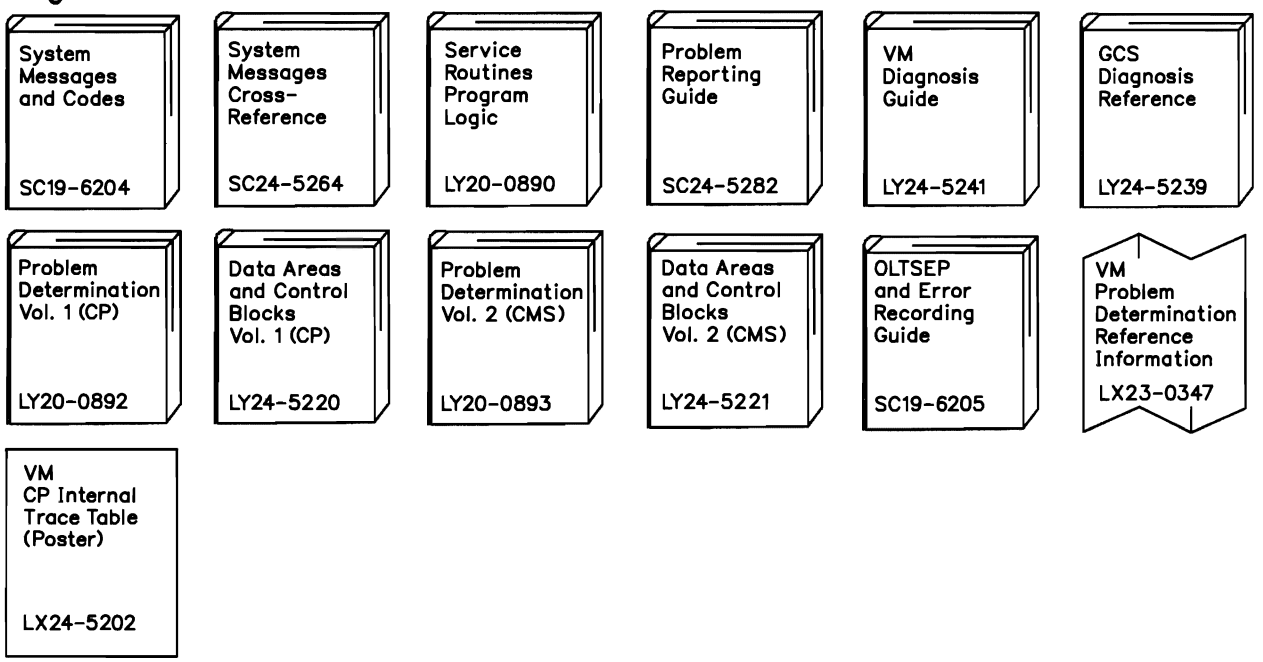


The VM/SP Library (Part 2 of 3)

End Use



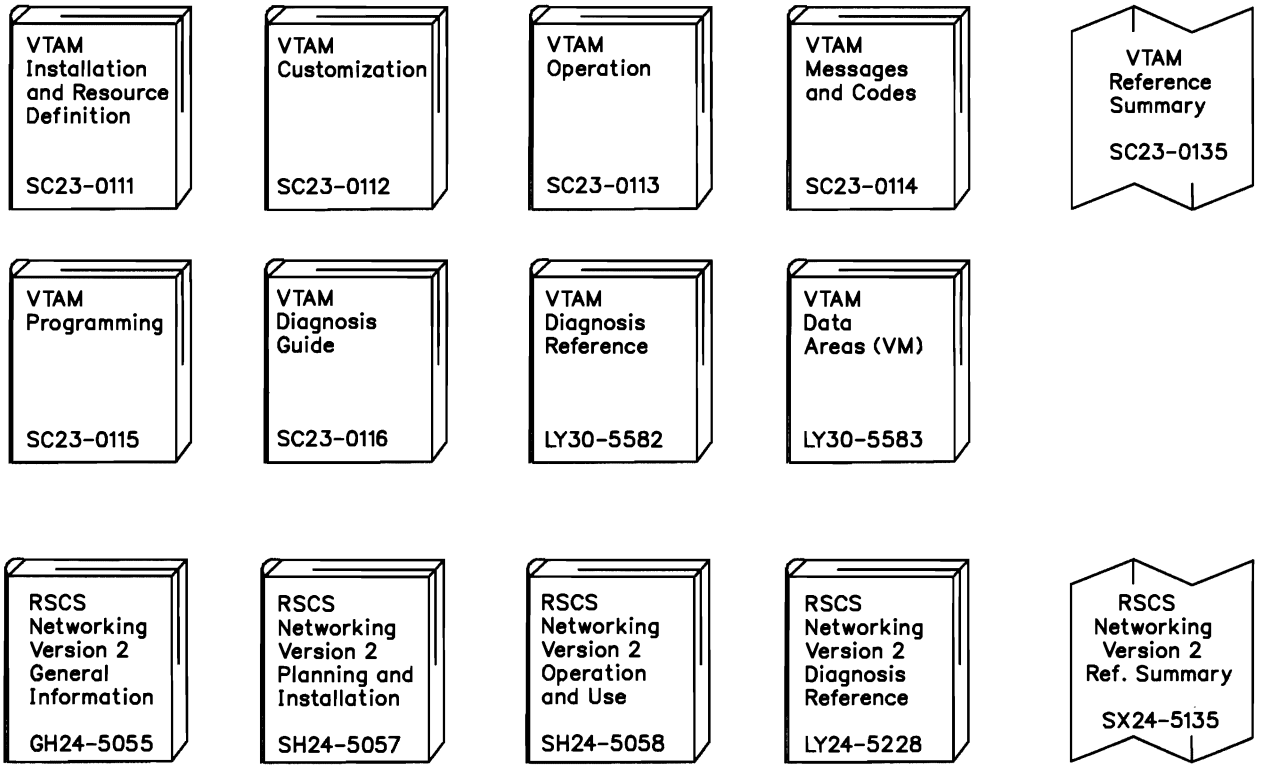
Diagnosis



The VM/SP Library (Part 3 of 3)

Administration

Auxiliary Communication Support



Special Characters

(VTOC) Volume Table of Contents
See CMSPROG
bclose
See CMSPROG
bdump
See CMSPROG
bopen
See CMSPROG
bopenr
See CMSPROG
bopnlb
See CMSPROG
bopnr2
See CMSPROG
bopnr3
See CMSPROG
bosvlt
See CMSPROG
*BLOCKIO (DASD Block I/O System Service)
See SFPROG
*CCS (SNA Console Communication Services)
See SFPROG
*LOGREC (Error Logging System Service)
See SFPROG
*MSG (Message System Service)
See SFPROG
*MSGALL (Message All System Service)
See SFPROG
*NCCF
See SFPROG
*SIGNAL (Signal System Service)
See SFPROG
*SPL (Spool System Service
See SFPROG
/JOB control cards
See CMSPROG

A

abend
See abnormal termination (abend)
ABEND macro
See DIAG
ABEND macro (SVC 13)
See CMSPROG
abend messages
See DIAG
abend, reason for
See DIAG
ABENDs
See CMSPROG
See DIAG
ABNEXIT macro
See CMSPROG
abnormal termination (abend)
See also CMSPROG
program interrupt 43
recovering spool files 14
saving virtual machine with VMSAVE option 7
abnormal termination procedures
See DIAG
ACCEPT, IUCV and logical device support facility
See SFPROG
ACCESS command
See CMSPROG
access device dependent information DIAGNOSE code X'8C'
See SFPROG
access diagnostic information saved for protected application facility users DIAGNOSE code X'B0'
See SFPROG
access method services
See CMSPROG
access method supported by OS
See CMSPROG
access to commands, changing 21
access verification routines 65
accounting
ACCTOFF routine 52
ACCTON routine 52
records 47-51
user options 52
ACCTOFF routine 52
ACCTON routine 52

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

ACF/VTAM, VM/SP SNA support
See SFPROG
action routines
See SFPROG
activating the TOD-clock accounting interface DIAGNOSE code X'70'
See SFPROG
active disk table (ADT)
See CMSPROG
ADDENTRY macro
See CMSPROG
ADSTOP command 54, 56
See also DIAG
ADT (active disk table)
See CMSPROG
affinity 181
in attached processor or multiprocessor mode 90
ALL subcommand of TRAPRED command
See DIAG
ALLOCATE control statement 126, 131
example of operation 131
allocating DASD space for CP use 128
alter contents of storage
See DIAG
altering a protected shared segment 56
altering storage contents
See DIAG
alternate path for I/O 101
with virtual=real option 85
alternate userid DIAGNOSE code X'D4'
See SFPROG
AP (attached processor mode)
advantages 170
affinity 90, 181
defer locks 177
fetching and storing 175
identify processor address 172
improving performance of 103
locking code 176
prefixing 170
real I/O interrupts 45
shared segments 181
signaling 172
special code in CP 169
spin locks 177
storage 171
synchronous interrupts 45
time-of-day clock 175
TOD clock 167
virtual machine I/O management 12
AP mode
See attached processor mode (AP)
APPLMSG macro
See CMSPROG
assembler language macros
See CMSPROG
assembler language programs
See CMSPROG
assembler virtual storage
See CMSPROG
ASSGN command
See CMSPROG
assigning dedicated channels to virtual machine 12
assist, CP 99
ATTACH macro (SVC 42)
See CMSPROG
attached processor mode (AP)
advantages 170
affinity 90, 181
defer locks 177
fetching and storing 175
identify processor address 172
improving performance of OS/VS2 MVS guest 103
locking code 176
prefixing 170
real I/O interrupts 45
shared segments 181
signaling 172
special code in CP 169
spin locks 177
storage 171
synchronous interrupts 45
time-of-day clock 175
TOD clock 167
virtual machine I/O management 12
attaching virtual devices 11
AUTHORIZE VMCF function
See SFPROG
Auto-Deactivation of Restricted Passwords (ADRP) 64
AUTOLOG command, journaling 63
auxiliary directories
See CMSPROG
auxiliary files
See CMSPROG
auxiliary storage 243
AUXPROC option of FILEDEF command
See CMSPROG

B

batch facility
See CMSPROG
BATEXIT1 routine
See CMSPROG
BATEXIT2 routine
See CMSPROG
BATLIMIT macro
See CMSPROG
BDAM
See CMSPROG
BEGIN command
See DIAG
BLDL macro (SVC 18)
See CMSPROG

BLIP character
See CMSPROG
BLIP facility 70
block, FBA device 125
*BLOCKIO
See SFPROG
BOTTOM subcommand of TRAPRED command
See DIAG
BPAM
See CMSPROG
breakpoint setting
See DIAG
BSAM/QSAM
See CMSPROG
BSP macro (SVC 69)
See CMSPROG
buffers
forms control 183
print 183
buffers used by FSCB
See CMSPROG

C

calculating, dispatching priority 74
CALL command
See CMSPROG
calling IBM for assistance, data needed
See DIAG
CANCEL command
See CMSPROG
CANCEL VMCF function
See SFPROG
capacity of device when formatted 123
card input to Format/Allocate program 126
CAT (character arrangement table) 204
See also character arrangement table (CAT) for 3800 printer
*CCS
See SFPROG
CHANGE command 205
3800 printer support 205
change summary 235
changing storage keys 56
changing, displaying, and setting SRM variables 86
channel program modification DIAGNOSE code X'28'
See SFPROG
channel use, improving 100
channel-to-channel adapter (CTCA) 243
See also Multisystem Communication Unit for 3088
CHAP macro (SVC 44)
See CMSPROG
character arrangement table (CAT) 204
character arrangement table (CAT) for 3800 printer 203, 205, 207, 208, 209
character modification for 3800 printer 203
CHECK macro
See CMSPROG
checkpoint start 14
CHKPT macro (SVC 63)
See CMSPROG
class
class override file
alternate 18
creating 28
example 30
making class assignments effective 31
verifying syntax 31
privilege 19
CLASS control statement 33, 35
clean-up after virtual IPL by device DIAGNOSE code X'40'
See SFPROG
clear error recording DIAGNOSE code X'1C'
See SFPROG
clock comparator 167
CLOSE function for SPOOL system service
See SFPROG
CLOSE/TCLOSE macro (SVC 20/23)
See CMSPROG
CMD command used by programmable operator
See SFPROG
CMD option of the PER command
See DIAG
CMS (Conversational Monitor System)
See virtual machines
CMS (Conversational Monitor System) commands
GENIMAGE 207
IMAGELIB 207
IMAGEMOD 207
PRINT 206
TRC option 206
SETKEY 54
SETPRT 205, 206
loading virtual 3800 printer modules 206
CMS abend dump reading
See DIAG
CMS abend recovery function
See DIAG
CMS BLIP facility 70
CMS control block relationship
See DIAG
CMS debugging
See DIAG
CMS dump file printing
See DIAG
CMS IUCV
See SFPROG
CMS loader, controlling

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

See CMSPROG
CMS/DOS
See CMSPROG
CMSDEV macro
See CMSPROG
CMSIUCV macro
See SFPROG
CMSLEVEL macro
See CMSPROG
CMSPROG
See the VM/SP CMS for System Programming manual
coding conventions for CP
addressing 60
constants 59
CP (Control Program) 59
error messages 61
format 59
loadlist requirements 62
module names 61
register use 59
title card 61
cold start 14
collecting CP data
See DIAG
collecting virtual machine data
See DIAG
command access
changing for virtual machines 21
migration considerations 28
security and system integrity 27
command authorization, planning 23
command language, CMS
See CMSPROG
command syntax files, updating
See SFPROG
commands
See also CMS commands and CP commands
assigning privilege classes 28
changing privilege classes 21
COMMANDS 33
GENIMAGE 208
how a user can display which can be used 33
IMAGELIB 210
IMAGEMOD 210
OVERRIDE 31
COMMANDS command 33
common segment facility 102
compiler
See CMSPROG
completion code X'00B' 44
COMPSWT macro
See CMSPROG
concurrent operation 243
CONNECT IUCV function
See SFPROG
connection complete external interrupt in IUCV
See SFPROG
connection pending external interrupt in IUCV
See SFPROG
connection quiesced external interrupt in IUCV
See SFPROG
connection resumed external interrupt in IUCV
See SFPROG
connection severed external interrupt in IUCV
See SFPROG
CONSOLE function
See CMSPROG
console input to Format/Allocate program 126
CONSOLE macro
See CMSPROG
control blocks
See CMSPROG
Control Program (CP)
assist 99
attached processor mode 169
changing privilege classes of internal CP functions 36
coding conventions 59
concurrent execution of virtual machines 3
eliminating paging 83
I/O management on virtual machine 11
in attached processor and multiprocessor modes 169
internal trace table 110
introduction 3
loadlist requirements 62
multiprocessor mode 169
Recovery Management Support (RMS) 44
RMS (Recovery Management Support) 44
scheduler 69
small CP option 88
spooling 13
control register allocation
See DIAG
control registers
See DIAG
control statement
See Format/Allocate service program
control statements
See also DASD Dump/Restore (DDR) program
CLASS 35
OVERRIDE 28
controlling PA2 program function key DIAGNOSE code X'54'
See SFPROG
conventions of coding 59
COPY function control statement, DDR program 145
copy modification for 3800 printer 203
copy modification modules (COPYMOD) 204, 207
COPYMOD (copy modification modules) 204, 207
COPYV command for MSS volumes 163
COUNT subcommand of the PER command
See DIAG
count-key-data DASD
copy restriction 153
formatting cylinder volumes 123
moving data with tape dump 141-153

record count area initializing 122
CP (Control Program)
assist 99
attached processor mode 169
changing privilege classes of internal CP functions 36
coding conventions 59
commands
See CP (Control Program) commands
concurrent execution of virtual machines 3
eliminating paging 83
I/O management on virtual machine 11
in attached processor and multiprocessor modes 169
internal trace table 110
introduction 3
loadlist requirements 62
multiprocessor mode 169
Recovery Management Support (RMS) 44
RMS (Recovery Management Support) 44
scheduler 69
small CP option 88
spooling 13
CP (Control Program) commands 19
ADSTOP 54, 56
assigning privilege classes 28
CHANGE for 3800 printer support 205
changing privilege classes 21
COMMANDS 33
DEFINE for 3800 printer support 205
DUMP 54
how a user can display which can be used 33
INDICATE 107
LOGON
journaling 63
password suppression 64
MIGRATE 86
MONITOR 107, 109
OVERRIDE 31
QUERY
for 3800 printer support 207
PAGING 87
SRM 86
SRM operand 107
SET
MIH (missing interrupt handler) 40
PAGING 87
SRM MHFULL 86
SPOOL for 3800 printer support 205
SRM 109
SRM operand 109
START for 3800 printer support 206, 207
STCP 56
STORE 54, 56
TRACE 54, 56
CP abend dumps, reading
See DIAG
CP coding conventions
addressing 60
constants 59
CP (Control Program) 59
error messages 61
format 59
loadlist requirements 62
module names 61
register use 59
title card 61
CP data, recording
See DIAG
CP debugging
See DIAG
CP FRET Trap
See DIAG
CP internal trace table
See DIAG
CP macros
FCB 197
FOB 192
FOBCCW 192
LOCK 179
PIB 196
PIBCCW 196
SIGNAL 172
SWTCHVM 181
UCB 188
UCBCCW 189
UCC 193
UCCCCW 193
UCS 186
UCSCCW 186
CP message repository
See SFPROG
CP SET DUMP command
See DIAG
CP system services
See SFPROG
CP trace table 43
CP trace table entries, recording
See DIAG
CPEREP program
See DIAG
CPRB macro
See CMSPROG
CPTRAP command
See DIAG
CPTRAP facility
See DIAG
CPU timer 166
CQYSECT macro
See CMSPROG
CSMRETCD macro
See CMSPROG
CTCA
See Multisystem Communication Unit for 3088

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

CVTSECT (CMS Communications Vector Table)
See DIAG
cylinder faults, MSS, VM/SP processing 162
cylinder zero format 124

D

DASD (direct access storage device)
See also disks
space allocating on CP owned volumes 128
volumes
formatting 121
initializing 121
labeling 131
DASD Block I/O System Service
See SFPROG
DASD Dump/Restore (DDR) program
See also DIAG
control statements
INPUT 141
OUTPUT 141
SYSPRINT 145
DDR command 138
function control statements 145
COPY 145
DUMP 145
PRINT 153
RESTORE 145
TYPE 153
functions of 138
invoking
as a stand-alone program 139
under CMS 138
restrictions 139, 150
sample output 159
data needed before calling IBM for assistance
See DIAG
data records for VM Monitor 218
data security in batch facility
See CMSPROG
Data Set Control Block (DSCB)
See CMSPROG
data sets, DOS
See CMSPROG
data sheet, problem inquiry
See DIAG
DCB (data control block)
See CMSPROG
DCB macro
See CMSPROG
DCP command
See DIAG
DCSS (discontiguous shared segment)
See CMSPROG
ddnames
See CMSPROG
DDR
See DASD Dump/Restore (DDR) program
DDR command
See CMSPROG
DDR program
See DIAG
deadline priority 72
deadline priority, calculating 74
debugging an AP/MP system
See DIAG
debugging CMS
See DIAG
debugging CP
See DIAG
debugging the virtual machine
See DIAG
debugging tools summary
See DIAG
debugging, introduction
See DIAG
declarative macros, DOS
See CMSPROG
DECLARE BUFFER IUCV function
See SFPROG
dedicated channel, assigning to virtual machine 12
DEFINE command for 3800 printer support 205
defining
privilege classes 33
changing for a virtual machine 33
virtual 3800 printer 205
DELENTRY macro
See CMSPROG
DELETE macro (SVC 9)
See CMSPROG
demand paging 75
DEQ macro (SVC 48)
See CMSPROG
DESCRIBE IUCV function
See SFPROG
DETACH macro (SVC 62)
See CMSPROG
detaching virtual devices 11
determine virtual machine storage size DIAGNOSE code X'60'
See SFPROG
Device Support Facility 121
device table (DEVTAB)
See CMSPROG
device type and features DIAGNOSE code X'24'
See SFPROG
device type class and values
See DIAG
devices
I/O 40
changing the time interval 41
default time intervals 41
determining time interval settings 42
sense information 39
devices, FBA 125
DEVTAB (device table)

See CMSPROG
DEVTYPE macro (SVC 24)
See CMSPROG
DIAG
See the VM Diagnosis Guide
DIAGNOSE code
assigning privilege classes 28
changing privilege classes 21
interface with a discontiguous saved segment 54
X'64', finding, loading, purging named segments
FINDSYS function 54
LOADSYS function 54
DIAGNOSE codes
See SFPROG
DIAGNOSE instruction
See also SFPROG
assigning privilege classes 28
determine virtual machine storage size 54
finding address of discontiguous saved segment 54
FINDSYS function 54
load discontiguous saved segment 54
LOADSYS function 54
MSS mount and demount processing 161
purge discontiguous saved segment 55
PURGESYS function 55
Direct Access Storage Device (DASD) 121
directory 244
directory update in-place DIAGNOSE code X'84'
See SFPROG
discontiguous saved segment (DCSS) 53-55
discontiguous shared segment (DCSS)
See CMSPROG
See SFPROG
Disk Operating System (DOS)
See CMSPROG
disks
allocating space for CP use 122
volumes
formatting 121
labeling 131
dispatcher stack lock 179
dispatching
priority, calculating 74
queue drop 70
run list protocols 73
time slice 73
DISPLAY command
See DIAG
display data on 3270 console screen DIAGNOSE code X'58'
See SFPROG
display real CP data
See DIAG
display terminals, CMS interface
See CMSPROG
display virtual data
See DIAG
displaying
and setting paging variables 87
changing and setting SRM variables 86
commands available to a user 33
DISPW macro
See CMSPROG
distributed system use of the programmable operator
See SFPROG
DMKDDR
See DASD Dump/Restore (DDR) program
DMKDIR (directory program) 131
DMKFMT
See Format/Allocate service program
DMKSNT (system name table) 57
DMSABN macro
See CMSPROG
See DIAG
DMSEXS macro
See CMSPROG
DMSFRE service routines
See CMSPROG
DMSFREE macro
See CMSPROG
DMSFRES macro
See CMSPROG
DMSFRET macro
See CMSPROG
DMSFST macro
See CMSPROG
DMSINA module
See CMSPROG
DMSINT module
See CMSPROG
DMSIOW module
See CMSPROG
DMSITE module
See CMSPROG
DMSITI module
See CMSPROG
DMSITP module
See CMSPROG
DMSITP routine
See DIAG
DMSITS module
See CMSPROG
DMSKEY macro
See CMSPROG
DMSNUC
See CMSPROG
DMSPAG module
See CMSPROG
DMSTVS module
See CMSPROG
DMSXFLPT XEDIT routine

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

See CMSPROG
DMSXFLRD XEDIT routine
See CMSPROG
DMSXFLST XEDIT routine
See CMSPROG
DMSXFLWR XEDIT routine
See CMSPROG
DOS (Disk Operating System)
See CMSPROG
DOSLIB command
See CMSPROG
DOWN subcommand of TRAPRED command
See DIAG
DSCB (Data Set Control Block)
See CMSPROG
DTFCD macro
See CMSPROG
DTFCN macro
See CMSPROG
DTFDI macro
See CMSPROG
DTFMT macro
See CMSPROG
DTFPR macro
See CMSPROG
DTFSD macro
See CMSPROG
DUMP command
See DIAG
DUMP function control statement, DDR program 145
Dump Restore service program, DASD 138
dump, used in problem determination
See DIAG
dumping to DASD
See DIAG
dumping to printer
See DIAG
dumping to tape
See DIAG
dumps
See DASD Dump/Restore (DDR) program
dynamic linkage
See CMSPROG
dynamic load overlay
See CMSPROG
dynamic loading
See CMSPROG

E

ECMODE option 166
ECPS (Extended Control-Program Support) 98, 100
virtual interval timer assist 99, 165
editing error messages DIAGNOSE code X'5C'
See SFPROG
efficiency of VM/SP performance options 77
eligible list 69
eliminating CP paging 83
eliminating queue drop overhead 96
end, abnormal
See abnormal termination (abend)
ENQ macro (SVC 56)
See CMSPROG
entry points
See CMSPROG
environment of VM/SP for system load 117
EPLIST (extended PLIST)
See CMSPROG
EPLIST macro
See CMSPROG
error codes
See CMSPROG
Error Logging System Service
See SFPROG
error messages editing DIAGNOSE code X'5C'
See SFPROG
examine real storage DIAGNOSE code X'04'
See SFPROG
EXCP supported by CMS/DOS
See CMSPROG
EXEC action routines
See SFPROG
EXEC procedures
See CMSPROG
EXIT/RETURN macro (SVC 3)
See CMSPROG
expanded virtual machine assist 99
extended control PSW description
See DIAG
Extended Control-Program Support (ECPS)
See ECPS (Extended Control-Program Support)
extents
See CMSPROG
External Attribute Buffer (XAB)
See SFPROG
external interrupt
See also CMSPROG
external console interrupt 45
interval timer 45
external interrupts in IUCV
See SFPROG
external references, resolving
See CMSPROG
EXTRACT macro (SVC 40)
See CMSPROG

F

faults, MSS cylinder, VM/SP processing 162
favored execution option 88
FB-512 format 125
FBA DASD
block format 125

copy restriction 148
device capacity 125
format 125
Format/Allocate program
using allocate function 128
using allocate overlap function 135
using label function 131
general description 122
page formatting/allocating 121
tape dump, movement of data 146
FCB
See Forms Control Buffer (FCB)
FCB (file control block)
See CMSPROG
FCB (Forms Control Buffer)
FEEDBACK command used by programmable operator
See SFPROG
feedback file
See SFPROG
FEOV macro (SVC 31)
See CMSPROG
fetch storage protection 5
file control block (FCB)
See CMSPROG
FILEDEF command
See CMSPROG
FILEDEF options
See CMSPROG
FIND macro (SVC 18)
See CMSPROG
finding address of discontiguous saved segment 54
finding discontiguous shared segment DIAGNOSE code X'64'
See SFPROG
FINDSYS function 54
See also SFPROG
fixed-head preferred paging area, migration 86
flashing, forms overlay, 3800 printer 203
FOB (font offset buffer)
FOBCCW macro instruction 192
3289 Model 4 184, 192
FOBCCW macro instruction 192
font offset buffer
See FOB (font offset buffer)
font offset buffer (FOB)
FOBCCW macro instruction 192
3289 Model 4 184, 192
force start 15
foreign languages
See SFPROG
FORMAT control statement of Format/Allocate program 127
format of cylinder zero 124
format of cylinders for CP use 125
FORMAT subcommand of TRAPRED command
See DIAG
Format/Allocate service program 122, 138
ALLOCATE control statement 128
card input 126
console input 133
control statements 126
cylinder format 123
cylinder zero format 124
description of 122
example of program execution 133
FORMAT control statement 127
LABEL control statement 131
overlap function for FBA devices 134
formatted device capacity 123
formatting
count-key-data DASD 123
FBA DASD 125
volumes, general information 121
Forms Control Buffer (FCB) 183, 202, 207
default image for virtual 3211-type printers 198
examples 198
index feature 198
macro 197
3203, 3211, 3262, 3289 Model 4, 4245, 4248 184
3800 printer 203, 204
forms overlay (flashing), 3800 printer 203
free storage
See CMSPROG
free storage lock 179
FREEDBUF macro (SVC 56)
See CMSPROG
FREELOWE
See CMSPROG
FREEMAIN macro (SVC 5)
See CMSPROG
FREEWORK (DMKFRE and DMKFRT save area)
See DIAG
FSCB (file system control block)
See CMSPROG
FSCB macro
See CMSPROG
FSCBD macro
See CMSPROG
FSCLOSE macro
See CMSPROG
FSERASE macro
See CMSPROG
FSOPEN macro
See CMSPROG
FSPOINT macro
See CMSPROG
FSREAD macro
See CMSPROG
FSSTATE macro
See CMSPROG
FST (file status table)
See CMSPROG
FSWRITE macro

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

See CMSPROG
full-screen console service
See CMSPROG
function control statement, DDR program 145
function control statements
See DASD Dump/Restore (DDR) program

G

GENDIRT command
See CMSPROG
general I/O DIAGNOSE code X'20'
See SFPROG
generate accounting records for the virtual user DIAGNOSE code X'4C'
See SFPROG
GENIMAGE command 207, 210
IEBIMAGE utility program 207
GENMOD command
See DIAG
GET command used by programmable operator
See SFPROG
GET macro
See CMSPROG
GETMAIN macro
See CMSPROG
GETMAIN macro (SVC 4)
See CMSPROG
GETPOOL/FREEPOOL macro
See CMSPROG
getting national languages on your system
See SFPROG
graphic character modification modules (GRAPHMOD) 207
graphic modification modules (GRAPHMOD) 204
GRAPHMOD (graphic modification modules) 204, 207
GUESTR option of PER command
See DIAG
GUESTV option of PER command
See DIAG

H

hardware assist 98
header record of VM Monitor 217
HEX subcommand of TRAPRED command
See DIAG
HNDEXT macro
See CMSPROG
HNDINT macro
See CMSPROG
HNDIUCV macro
See CMSPROG
See SFPROG
HNDSVC macro
See CMSPROG
HOSTCHK statement
See SFPROG

I

I/O (input/output)
See also CMSPROG
interrupt 39
lock 178
management 11
overhead in CP, reducing 79
recovery from errors 212
virtual machines 78
I/O definition statements for DDR program 141
IBCDASDI 121
IBM 3800 Printing Subsystem 203, 212
See also 3800 printer
CHANGE command 205
SETPRT command 206
SPOOL command 205
IBM-defined user classes, reverting to 32
IDENTIFY macro (SVC 41)
See CMSPROG
identify processor address in AP/MP environment 172
IDENTIFY VMCF function
See SFPROG
IEBIMAGE utility program 207
IIP (ISAM Interface Program)
See CMSPROG
IMAGE library 207
IMAGELIB command 207
IMAGELIB service program
command format 210
responses 209
IMAGEMOD command 207
format and description 210
modifying 3800 named system 210
IMMBLOK macro
See CMSPROG
IMMCMD macro
See CMSPROG
imperative macros
See CMSPROG
improving channel use 100
INDICATE command 107
See also DIAG
FAVORED operand, E privilege class 108
indicators of system load 107
INITDISK 121
initial program load (IPL)
INITIATE logical device support facility function
See SFPROG
INPUT control statement for DDR program 141
input spool file manipulation DIAGNOSE code X'14'

See SFPROG
input/output (I/O)
See I/O (input/output)
Installation DCSS
See CMSPROG
installing the programmable operator facility
See SFPROG
instruction
See DIAGNOSE instruction
integrity of system
when changing command access 27
Inter-User Communications Vehicle (IUCV)
See SFPROG
Interactive File Sharing, performance improvement 97
internal CP functions, changing privilege classes 36
internal trace table, CP
See DIAG
interrupt handler, DMSITI module
See CMSPROG
interrupt handling
See also CMSPROG
attached processor
real I/O interrupts 45
synchronous interrupts 45
external interrupts 45
I/O interrupts 11
machine check interrupts 44
missing interrupt handler 39
multiprocessor
real I/O interrupts 45
synchronous interrupts 45
program interrupts 43
SVC interrupts 44
interval timer 99, 165, 166
introduction to VM/SP Control Program (CP) 3
introduction to VM/SP CP (Control Program) 3
INTSVC for SVC handling routine
See CMSPROG
invoking DDR
as a stand-alone program 139
under CMS 138
invoking the programmable operator facility
See SFPROG
IPL command
See CMSPROG
IPL performance using saved system
See CMSPROG
ISAM
See CMSPROG
ISAM Interface Program (IIP))
See CMSPROG
issue SVC 76 from a second level virtual machine DIAGNOSE code X'48'
See SFPROG
IUCV (Inter-User Communications Vehicle)
See SFPROG
IUCV functions
See SFPROG
IUCV macro instruction
See SFPROG

J

job control cards (/JOB)
See CMSPROG
journaling
accounting records 49
LOGON, AUTOLOG, LINK commands 63

K

keys
See also CMSPROG
changing storage 56
storage key 6

L

LABEL control statement, Format/Allocate program 131
labeling DASD volumes 131
LANGBLK macro
See CMSPROG
LANGGEN command
See SFPROG
LANGMERG command
See SFPROG
language, CMS command
See CMSPROG
languages, national
See SFPROG
LCS (library character sets) 204, 207
LGLOPR statement
See SFPROG
libraries
See CMSPROG
library character sets (LCS) 204, 207
library, IMAGE 207
LINEDIT macro
See CMSPROG
LINERD macro
See CMSPROG
LINEWRT macro
See CMSPROG

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

LINK command
See also CP (Control Program) commands
journaling 63
password suppression 64
LINK macro (SVC 6)
See CMSPROG
LIOCS routines supported by CMS/DOS
See CMSPROG
load 107
environments of VM/SP 117
indicators 107
LOAD command
See DIAG
LOAD macro (SVC 8)
See CMSPROG
load map generation
See DIAG
load maps
See CMSPROG
See DIAG
loader
See CMSPROG
loader tables in CMS
See CMSPROG
loading and saving discontiguous saved segments 54
discontiguous shared segment DIAGNOSE code X'64'
See SFPROG
loading virtual 3800 printer modules 206
loadlist requirements 62
LOADSYS function 54
See also SFPROG
LOADTBL command used by programmable operator
See SFPROG
LOCK macro 179
locked pages option 81
locks for serializing AP/MP functions 178, 180
LOG command used by programmable operator
See SFPROG
log file
See SFPROG
LOGGING statement
See SFPROG
logical device support facility
See SFPROG
logical device support facility DIAGNOSE code X'7C'
See SFPROG
logical operator
See SFPROG
logical units
See CMSPROG
LOGON command
See also CP (Control Program) commands
journaling 63
password suppression 64
*LOGREC
See SFPROG
long wait state 71
loop procedures
See DIAG
looping programs
See DIAG
low address protection 102

M

machine check
during start-up 44
interrupt 44
machine check interrupts
See CMSPROG
macro notes, missing interrupt handler 43
MAINHIGH
See CMSPROG
MAP option of GENMOD command
See DIAG
MAP option of LOAD command
See DIAG
Mass Storage System
See MSS (Mass Storage System)
Mass Storage System (MSS)
cylinder faults, VM/SP processing 162
mount and demount processing 161
mount processing, asynchronous 162
VM/SP access 161
volumes 12
backup copies 163
I/O management 12
mdisk 125
Message All System Service
See SFPROG
message complete external interrupt in IUCV
See SFPROG
MESSAGE function for SPOOL system service
See SFPROG
message pending external interrupt in IUCV
See SFPROG
message repository
See CMSPROG
See SFPROG
Message System Service
See SFPROG
microcode assist, none for SSK instruction with shared system 57
MIGRATE command 86
migration
page, managing 86
MIH (missing interrupt handler) 39
description 39
devices monitored 40
diagnostic aids 42
error recording area 43
macro notes 43
messages 43

use of 40
minidisks 11
See also CMSPROG
minidisks initializing 121
missing interrupt handler (MIH) 39
description 39
devices monitored 40
diagnostic aids 42
error recording area 43
macro notes 43
messages 43
use of 40
mixed environment use of the programmable operator
See SFPROG
modifying 3800 named system 210
MODMAP command
See DIAG
module load map
See DIAG
MONITOR CALL instruction 109
MONITOR command 107, 109
See also DIAG
format 111
implemented classes 111
MONITOR START command
See DIAG
monitoring recommendations 116
movable-head preferred paging area, managing migration 86
MOVEFILE command
See CMSPROG
MP (multiprocessor mode)
advantages 170
affinity 90, 181
configuring I/O 182
defer locks 177
fetching and storing 175
identify processor address 172
locking code 176
prefixing 170
real I/O interrupts 45
shared segments 181
signaling 172
special code in CP 169
spin locks 177
storage 171
synchronous interrupts 45
time-of-day clock 175
virtual machine I/O management 12
MP mode
See multiprocessor mode (MP)
*MSG
See SFPROG
*MSGALL
See SFPROG
MSS (Mass Storage System)
cylinder faults, VM/SP processing 162
mount and demount processing 161
mount processing, asynchronous 162
VM/SP access 161
volumes 12, 77
backup copies 163
I/O management 12
MSS communication DIAGNOSE code X'78'
See SFPROG
MSSF SCPINFO command
See SFPROG
MSSFCALL DIAGNOSE code X'80'
See SFPROG
multiple address stops
See DIAG
multiple copy printing, 3800 printer 203
multiple extents
See CMSPROG
multiple shadow table support 94
multiprocessing systems, improving performance of 103
multiprocessor
virtual machine I/O management 11
multiprocessor mode (MP)
advantages 170
affinity 90, 181
configuring I/O 182
defer locks 177
fetching and storing 175
identify processor address 172
locking code 176
prefixing 170
real I/O interrupts 45
shared segments 181
signaling 172
special code in CP 169
spin locks 177
storage 171
synchronous interrupts 45
time-of-day clock 175
virtual machine I/O management 12
Multisystem Communication Unit for 3088 213
CTCA with V=R option 85
CTCA, extending addressing and scheduling of 213
multivolume extents
See CMSPROG
MVS/system extensions support 101
common segment facility 102
enabling 103
low address protection 102
special operations and instructions 103

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

N

named segments, finding, loading, purging
  See SFPROG
named system
  See saved system
NAMENCP macro
  for 37XX control program 57
NAMESYS macro 7, 8, 55
  See also CMSPROG
national languages
  See SFPROG
native languages
  See SFPROG
native mode, switching to or from 104
*NCCF
  See SFPROG
NCCF (Network Communications Control Facility)
  See SFPROG
NCCF logical operator
  See SFPROG
Network Communications Control Facility (NCCF)
  See SFPROG
non-recoverable machine check
  See DIAG
nonrelocatable modules
  See CMSPROG
NOTE macro
  See CMSPROG
NOTIFY function for SPOOL system service
  See SFPROG
nucleus free storage
  See CMSPROG
nucleus load map
  See DIAG
nucleus, CMS
  See CMSPROG
NUCON macro
  See CMSPROG

O

OPEN macro
  See CMSPROG
OPEN/OPENJ macro (SVC 19/22)
  See CMSPROG
Operating System (OS)
  See CMSPROG
OS (Operating System)
  See CMSPROG
OS/VS in native mode 105
OS/VSAM
  See CMSPROG
OS/VS2 MVS guest on a multiprocessor 103
OUTPUT control statement for DDR program 141
overhead, CP, reducing for I/O 79
overlap function for FORMAT/ALLOCATE program 134
overlapping areas 9
overlay structures
  See CMSPROG
OVERRIDE command 31
OVERRIDE control statement 28
override file
  See class, class override file
override, ALLOCATE control statement 128, 130

P

page frames, reserving 82
page management
  See CMSPROG
page on FBA DASD 123
page release function DIAGNOSE code X'10'
  See SFPROG
page, storage
  exceptions, effects of 80
  frames 75
  locking 81
  migration, managing 86
  reserved page frames 82
  Set Page Boundary (SPB) card 62
  SPB (Set Page Boundary) card 62
  table 74
pageable module, identify and locate
  See DIAG
paging 75, 83
  by demand 75
  considerations 80
paging variables, displaying and setting 87
parameter lists of IUCV functions
  See SFPROG
parsing facility
  See CMSPROG
partitioned data set
  See CMSPROG
password
  See also CMSPROG
  auto-deactivation 64
  suppressing on command line 64
paths, IUCV
  See SFPROG
PA1 program function key 3
PA2 program function key controlling DIAGNOSE code X'54'
  See SFPROG
PER command
  See DIAG
performance of virtual machines
  factors affecting performance 77
  for mixed mode foreground/background systems 118

for time-shared multibatch virtual machines 116
High Performance Option 6
locked pages 81
measurement 107
performance options, VM/SP 87-105
reserved page frames 82
switching SCP to or from native mode 104
virtual machine assist feature 97
PGRLSE macro (SVC 112)
See CMSPROG
PIB buffer images
examples 197
macro format 196
PIBCCW macro format 196
PIBCCW macro 196
PLIST (parameter list)
See CMSPROG
PMX (Programmable Operator/NCCF Message Exchange)
See SFPROG
POINT macro
See CMSPROG
POST macro (SVC 2)
See CMSPROG
poster, CP internal trace table
See DIAG
preferred auxiliary files
See CMSPROG
prefixing in an AP/MP environment 170
PRESENT logical device support facility function
See SFPROG
preserving virtual storage 7, 10
prestructured overlays
See CMSPROG
print buffers
adding new images 185
LOADBUF command 186
PIB buffer images 196
PIBCCW macro 196
print chain image 186
UCB macro 188
UCBCCW macro 189
UCC examples 194
UCC macro 193
UCCCCW macro 193
UCS
examples 187
macro 186
1403 and 3203 183
UCSB
associative field 188
examples 189
3211 183, 188
3262 184
UCSCCW macro 186
PRINT command 206
TRC option 206
PRINT function control statement
DDR program 153
DDR sample output 159
printer interrupts
See CMSPROG
PRINTER subcommand of TRAPRED command
See DIAG
printers
See Numeric section of this index
printing tape dump
See DIAG
printing virtual 3800 spool files 206
priority of virtual machines
deadline 72, 74
of execution 74
performance option 90
setting for storage preservation 8
privilege classes 19
assigning to commands and DIAGNOSE codes 28
changing for internal CP functions 36
changing the definition of 33
defining eight or fewer 35
defining more than eight 35
reverting to IBM-defined 32
privileged instructions 78
problem analysis
See DIAG
problem exist ?
See DIAG
problem identification
See DIAG
problem inquiry data sheet
See DIAG
problem types
See DIAG
processor
resources 69
timer 166
utilization 69
PROFILE EXEC
See CMSPROG
program check debugging
See DIAG
program exceptions
See DIAG
program interrupts
See CMSPROG
program modules, creating
See CMSPROG
program status word (PSW)
virtual 4
programmable operator facility
See SFPROG
Programmable Operator/NCCF Message Exchange (PMX)

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

See SFPROG
programs
interruption 43
states 4
projected working set size 72
promotion to run list 73
PROPCHK statement
See SFPROG
PROPRTCV for converting routing tables
See SFPROG
protected application facility DIAGNOSE code X'B0'
See SFPROG
protected shared segment 55
protecting storage 5
keys 6
PRTDUMP command
See DIAG
pseudo timer 167
pseudo timer DIAGNOSE code X'0C'
See SFPROG
PSW (program status word) 4
See also CMSPROG
virtual 4
punch interrupts
See CMSPROG
PUNCHC macro
See CMSPROG
PURGE IUCV and VMCF functions
See SFPROG
PURGESYS function 55
See also SFPROG
purging discontiguous saved segments 55
purging discontiguous shared segment DIAGNOSE code X'64'
See SFPROG
PUT macro
See CMSPROG
PUTX macro
See CMSPROG

Q

QUERY command
See also CMSPROG
PAGING operand 87
SRM operand 86, 109
3800 printer support 205, 207
QUERY IUCV function
See SFPROG
QUERY SRM command
See DIAG
querying SRM variables 109
queue drop 70, 71-72
queue drop elimination 96
QUIESCE IUCV and VMCF functions
See SFPROG
QUIT subcommand of TRAPRED command
See DIAG

R

RACF (Resource Access Control Facility) 65
RACF/VM 1.7.1 65
RCTLUNIT macro 213
coding 213
RDCARD macro
See CMSPROG
RDEVBLOK lock 179
RDEVICE macro 213
coding 213
RDJFCB macro (SVC 64)
See CMSPROG
RDTAPE macro
See CMSPROG
RDTERM macro
See CMSPROG
READ functions for SPOOL system service
See SFPROG
read LOGREC data DIAGNOSE code X'30'
See SFPROG
READ macro
See CMSPROG
read system dump spool file DIAGNOSE code X'34'
See SFPROG
read system symbol table DIAGNOSE code X'38'
See SFPROG
read/write operation
See CMSPROG
reader interrupts
See CMSPROG
real channel program support DIAGNOSE code X'98'
See SFPROG
real storage
making best use of 75
real storage management lock (RM Lock) 178
REALTIMER option 166
RECEIVE IUCV and VMCF function
See SFPROG
record format
See CMSPROG
records, accounting 47, 51
for AUTOLOG, LOGON, and LINK journaling 49
recovering spool files 14, 15
RECOVERV command for MSS volumes 163
Recovery Management Support (RMS) 44
recovery, CMS abend
See CMSPROG
reducing purges 94
reduction
of CP overhead for virtual machine I/O 79
of paging activity 80

of SIO operation 79
reenterable code, usage 80
references, resolving unresolved
See CMSPROG
REGEQU macro
See CMSPROG
REJECT IUCV and VMCF functions
See SFPROG
releasing storage
See CMSPROG
Remote Spooling Communication Subsystem (RSCS) networking 13
repetitive output
See DIAG
REPLY IUCV and VMCF function
See SFPROG
repository files
See CMSPROG
reserved page frames 82
Resource Access Control Facility (RACF) 65
See also RACF (Resource Access Control Facility)
resources, processor 69
responses of VM Monitor to unusual tape conditions 113
RESTORE function control statement, DDR program 145
RESTORE macro (SVC 17)
See CMSPROG
restricted passwords auto-deactivation 64
restrictions
count-key-data devices 150
DDR 150
FBA DASD 150
for using DDR 139
RESUME IUCV and VMCF function
See SFPROG
retrieve a group name DIAGNOSE code X'A0'
See SFPROG
RETRIEVE BUFFER IUCV function
See SFPROG
RM lock (real storage management lock) 178
RMS (Recovery Management Support) 44
ROUTE statement
See SFPROG
routing table (RTABLE)
See SFPROG
RSCS (Remote Spooling Communication Subsystem) networking 13
RTABLE (routing table)
See SFPROG
run list 69
run list lock 178
run list protocols 73

S

SAM (sequential access method)
See CMSPROG
save the 370X control program image DIAGNOSE code X'50'
See SFPROG
saved system
See also CMSPROG
definition 53
other saved systems 10
segment zero in protected named system 55
SAVENCP command 57
for 37XX control program 57
SAVESEQ operand for setting priority 8
saving or loading a 3800 named system DIAGNOSE code X'74'
See SFPROG
saving the CP message repository DIAGNOSE code X'CC'
See SFPROG
SCBLOCK control block
See CMSPROG
SCBLOCK macro
See CMSPROG
scheduler lists and "queues" 71
scheduler, CP 69-72
SCIF (Single Console Image Facility)
See SFPROG
SCP, switching to or from native mode 104
SCPINFO command
See SFPROG
screen management VM/SP SNA support
See SFPROG
search order
See CMSPROG
security
monitoring system access 63
RACF (Resource Access Control Facility) 65
when changing command access 27
segment table 74
segments, shared
See shared segments
SELECT function for SPOOL system service
See SFPROG
SEND IUCV and VMCF functions
See SFPROG
SEND/RECV VMCF function
See SFPROG
SENDX VMCF function
See SFPROG
service program
See Format/Allocate service program
See GENIMAGE command
See IMAGELIB service program

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

service routines
See CMSPROG
SET command
See also CMSPROG
MIH operand 40
PAGING operand 87
SRM MHFULL operands 86
SRM operand 86, 109
TIMER operand 166
SET command used by programmable operator
See SFPROG
SET CONTROL MASK IUCV function
See SFPROG
set languages DIAGNOSE code X'C8'
See SFPROG
SET MASK IUCV function
See SFPROG
SETPRT command 205, 206
loading virtual 3800 printer modules 206
setting and displaying paging variables 87
setting priority for saving virtual machines 8
setting SRM variables 109
setting, changing, and displaying SRM variables 86
SEVER IUCV function
See SFPROG
SFPROG
See VM System Facilities for Programming
shadow table bypass 95
shadow table maintenance DIAGNOSE code X'6C'
See SFPROG
shared segments 53, 56
no microcode assist for SSK instruction when a shared system is running 57
SHVBLOCK macro
See CMSPROG
*SIGNAL
See SFPROG
SIGNAL macro 172
Signal System Service
See SFPROG
signaling in an AP/MP environment 172
simulated OS supervisor calls
See CMSPROG
simulation 78
See also CMSPROG
CP (Control Program) 78
Single Console Image Facility (SCIF)
See SFPROG
single processor mode 103
single system use of the programmable operator
See SFPROG
single-instruction mode 3
SIO
See Start I/O (SIO) instruction
small CP option
effect on performance 88
purpose of 88
SNA (System Network Architecture)
See SFPROG
SNAP macro (SVC 51)
See CMSPROG
spanned records, usage
See CMSPROG
SPB (Set Page Boundary) card 62
Special Message Facility
See SFPROG
specifying overlapping areas 9
specifying target areas 8
SPIE macro (SVC 14)
See CMSPROG
*SPL
See SFPROG
SPOOL command 205
See also CP (Control Program) commands
3800 printer support 205
spool file external attribute buffer manipulation DIAGNOSE code X'B8'
See SFPROG
spool file recovery 14
SPOOL System Service
See SFPROG
spooling
described 13
terminal I/O 14
via RSCS 13
SRM variables, setting, changing, and displaying 86
SSERV command
See CMSPROG
SSK instruction, no microcode assist with shared systems 57
STAE macro (SVC 60)
See CMSPROG
standard DASD I/O DIAGNOSE code X'18'
See SFPROG
START command for 3800 printer support 206, 207
Start I/O (SIO) instruction
handling 78
reducing 79
start of LOGREC area DIAGNOSE code X'2C'
See SFPROG
STAX macro (SVC 96)
See CMSPROG
STCP command 56
See also DIAG
STIMER macro (SVC 47)
See CMSPROG
STOP command used by programmable operator
See SFPROG
stop execution
See DIAG
storage
See also CMSPROG
AP/MP environment 171
auxiliary 243
dynamic paging 80
keys 6
changing 56

protection 5
storage contents, altering
See DIAG
storage page
exceptions, effects of 80
frames 75
locking 31
migration, managing 86
reserved page frames 82
Set Page Boundary (SPB) card 62
SPB (Set Page Boundary) card 62
table 74
storage, CMS
See CMSPROG
STORE command 54, 56
See also DIAG
store extended-identification code DIAGNOSE code X'00'
See SFPROG
store real CP data
See DIAG
store virtual data
See DIAG
STOW macro (SVC 21)
See CMSPROG
STRINIT macro
See CMSPROG
structure of CMS storage
See CMSPROG
SUBCOM function
See CMSPROG
summary of changes 235
supervisor calls
See CMSPROG
support facilities for devices 121
surface analysis utility 121
SVC
See CMSPROG
SVC interrupts 44
See also CMSPROG
SVC save area (SVCSAVE)
See DIAG
SVC 202
See CMSPROG
SVC 203
See CMSPROG
SVCTRACE command
See DIAG
switching SCP to or from native mode 104, 105
switching, CMS tape volume
See CMSPROG
SWITCHVM macro 181
SYNADAF macro (SVC 68)
See CMSPROG
SYNADRLS macro (SVC 68)
See CMSPROG
SYSCAT system logical unit
See CMSPROG
SYSCLB system logical unit
See CMSPROG
SYSCOR macro
See DIAG
SYSIN system logical unit
See CMSPROG
SYSIPT system logical unit
See CMSPROG
SYSJRL macro instruction 64
SYSLOG system logical unit
See CMSPROG
SYSLST system logical unit
See CMSPROG
SYSPCH system logical unit
See CMSPROG
SYSPRINT control statement of DDR program 145
SYSPROF EXEC
See CMSPROG
SYSRDR system logical unit
See CMSPROG
SYSRLB system logical units assigned in CMS/DOS
See CMSPROG
SYSSLB system logical units assigned in CMS/DOS
See CMSPROG
system
integrity when changing command access 27
performance 107
security measures 63
system abend
See DIAG
system information, collect and analyze
See DIAG
system logical units
See CMSPROG
system name table (DMKSNT) 57
System Network Architecture (SNA)
See SFPROG
system profile
See CMSPROG
system resource management variables 86
system save area
See CMSPROG
system service, CP
See SFPROG
SYSxxx (programmer logical units)
See CMSPROG

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

T

TAG command
  See CP (Control Program) commands
tailoring your system
  See CMSPROG
tape conditions with VM Monitor 113
tape volume switching in CMS
  See CMSPROG
TAPECTL macro
  See CMSPROG
tapes
  See CMSPROG
TAPESL macro
  See CMSPROG
target areas 8
TCLEARQ macro (SVC 94)
  See CMSPROG
temporary disks
  See CMSPROG
TEOVEXIT macro
  See CMSPROG
terminal interrupts
  See CMSPROG
TERMINATE ALL logical device support facility function
  See SFPROG
TERMINATE logical device support facility function
  See SFPROG
termination, abnormal
  See abnormal termination (abend)
TEST COMPLETION IUCV function
  See SFPROG
TEST MESSAGE IUCV function
  See SFPROG
text deck names from GENIMAGE command 209
TEXTSYM statement
  See SFPROG
TGET/TPUT macro (SVC 93)
  See CMSPROG
TIME macro (SVC 11)
  See CMSPROG
time management 69
time sharing 247
time slice, run list 73
time-of-day (TOD) clock 167
  in attached processor environment 167
TIMER operand of SET 166
timer request queue lock 179
timers 165, 168
  clock comparator 167
  interval timer 99, 165
  processor (CPU) timer 166
  pseudo timer 167
  time-of-day (TOD) clock 167
TOD-clock accounting interface
  See SFPROG
tokenized PLIST
  See CMSPROG
TOP subcommand of TRAPRED command
  See DIAG
TRACCURR (current CP internal trace table entry)
  See DIAG
TRACE command
  See also DIAG
  protected shared page 56
  testing nonshared segments 55
trace execution
  See DIAG
trace real machine events
  See DIAG
trace table, CP 43
TRACEND (end of CP internal trace table)
  See DIAG
tracing
  See DIAG
tracing capabilities in EXECs
  See DIAG
tracing storage alteration using PER
  See DIAG
TRACSTRT (start of CP internal trace table)
  See DIAG
TRANSFER command
  See CP (Control Program) commands
transient program area
  See CMSPROG
transient routines
  See CMSPROG
TRAPRED command format
  See DIAG
TRAPRED facility
  See DIAG
TRAPRED subcommands
  See DIAG
TRKBAL macro (SVC 25)
  See CMSPROG
TTIMER macro (SVC 46)
  See CMSPROG
TVSPARMS macro
  See CMSPROG
TYPE function control statement
  DDR program 153
  DDR sample output 159
TYPE subcommand of TRAPRED command
  See DIAG
TYPEBACK subcommand of TRAPRED command
  See DIAG
typenum subcommand of TRAPRED command
  See DIAG
types of locks 177

U

UCS (Universal Character Set)
  adding buffer images 185
  supplied images 183
UCSB (Universal Character Set Buffer)
  supplied images 183
UNAUTHORIZE VMCF function
  See SFPROG
unexpected results procedures
  See DIAG
unit record, devices, sharing 13
Universal Character Set
  See UCS (Universal Character Set)
Universal Character Set (UCS)
Universal Character Set Buffer (USCB)
unprotected shared segment 55
UP subcommand of TRAPRED command
  See DIAG
updating VM/SP directory DIAGNOSE code X'3C'
  See SFPROG
user area (USERSECT)
  See CMSPROG
USER control statement 33
user free storage
  See CMSPROG
user program area
  See CMSPROG
user save area
  See CMSPROG
user-controlled device interrupts
  See CMSPROG
user-defined lock 179
userclasses, reverting to IBM-defined 32
USERSECT (user area)
  See CMSPROG
USERSECT macro
  See CMSPROG
using processor resources 69
using the VMSAVE option 7

V

V=R machine used with single processor mode 103
variable length record
  See CMSPROG
verification routines, access 65
virtual
  block multiplexer channel option 100
  I/O to virtual device 3
  operator's console 3
  processor 3
virtual address 247
virtual console function DIAGNOSE code X'08'
  See SFPROG
virtual console, operator 3
virtual device, attaching and detaching 11
virtual disks 247
virtual I/O device 3
virtual interval timer assist 99, 165
virtual machine assist feature
  described 97
  restrictions for use of 98
  use 97
Virtual Machine Communication Facility (VMCF)
  See SFPROG
virtual machine data, recording
  See DIAG
virtual machine debugging
  See DIAG
Virtual Machine Facility/370 (VM/370)
  using ECPS 99
Virtual Machine Monitor Analysis Program (VMMAP) 114
virtual machine storage size DIAGNOSE code X'60'
  See SFPROG
Virtual Machine/System Product (VM/SP)
  See also VM/SP (Virtual Machine/System Product)
  Control Program (CP) 3
  CP (Control Program) 3
  directory 3
  load environment 117
  program states 4
virtual machines
  changing access to commands 21
  changing the definition of privilege classes 33
  creation 3
  defining eight or fewer privilege classes 35
  defining more than eight privilege classes 35
  described 3
  directory 3
  I/O management 11, 12
  I/O operation 78
  operating system 3
  performance options 87-105
    for time-shared multibatch virtual machines 116
    VMMAP 114, 116
  program status word (PSW) 4
  shared segment operation 56
  storage management 74
    directory 74
    virtual storage 74
  time management 69
    interactive user 70
    noninteractive user 70
    priority of execution 74
    queue drop 70
  timers 165

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

virtual printer external attribute buffer manipulation DIAGNOSE code X'B4'
See SFPROG
virtual program status word 4
Virtual Reserve/Release support, virtual machine I/O management 11
virtual storage
management by CP 74
preservation 7, 10
Virtual Storage Access Method (VSAM)
See CMSPROG
virtual storage, altering
See DIAG
VM MONITOR 107
collection mechanism 110
considerations 111
data records 218
data volume and overhead 115
header record 217
monitor classes 110
output 114
responses to unusual tape conditions 113
tape format and contents 217
VMMAP 114
VM/SP (Virtual Machine/System Product)
Control Program (CP) 3
CP (Control Program) 3
directory 3
load environment 117
program states 4
VM/SP directory updating DIAGNOSE code X'3C'
See SFPROG
VM/VCNA, VM/SP SNA support
See SFPROG
VM/VTAM, VM/SP SNA support
See SFPROG
VM/370
See Virtual Machine Facility/370 (VM/370)
VMBLOK lock 178
VMCF (Virtual Machine Communication Facility)
See SFPROG
VMDUMP command
See DIAG
VMDUMP function
See SFPROG
VMDUMP Function DIAGNOSE code X'94'
See SFPROG
VMMAP (Virtual Machine Monitor Analysis Program) 114
VMSAVE areas 8
VMSAVE option 7
Volume Table of Contents (VTOC)
See CMSPROG
VSAM (Virtual Storage Access Method)
See CMSPROG
VSAM data sets
See CMSPROG
VSCS, VM/SP SNA support
See SFPROG
VSE
See CMSPROG
VSE/VSAM
See CMSPROG
VTAM service machine VM/SP SNA support
See SFPROG

W

wait bit, modifying
See DIAG
WAIT macro (SVC 1)
See CMSPROG
wait state procedures
See DIAG
WAITD macro
See CMSPROG
WAITECB macro
See CMSPROG
WAITT macro
See CMSPROG
warm start 14
working set size, projected 72
Writable Character Generation Modules (WCGM) 205
WRITE macro
See CMSPROG
WRTAPE macro
See CMSPROG
WRTERM macro
See CMSPROG
WTO/WTOR macro (SVC 35)
See CMSPROG

X

XAB (External Attribute Buffer)
See SFPROG
XCTL macro (SVC 7)
See CMSPROG
XDAP macro (SVC 0)
See CMSPROG
XEDIT (System Product Editor)
See CMSPROG
XEDIT command
See CMSPROG

Numerics

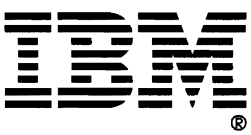
1403 UCS buffer images 183, 186
3081 processor MSSFCALL - DIAGNOSE code X'80'
  See SFPROG
3088 multisystem channel communication unit 213
3203
  forms control and print buffer 183
3211-type printer
  default virtual FCB image 198
  UCSB buffer images 183, 188
3262
  FCB 197
  PIB buffer images 195
  UCSB buffer images 184
3270 virtual console interface DIAGNOSE code X'58'
  See SFPROG
3289 font offset buffer
  adding FOBs 192
  FOB macro instruction 192
  purpose of 192
3310 DASD
  See also FBA DASD
  device capacity 125
3370 DASD
  See also FBA DASD
  device capacity 125
  general description 122, 123
3480 magnetic tape subsystem 140
  restrictions under DDR 140
3480 tape volume serial support DIAGNOSE code X'D0'
  See SFPROG
37XX Control Program
  See also SFPROG
  system name table 57
  using the NAMENCP macro 57
  using the SAVENCP command 57
3800 printer
  as a dedicated device 204
  as a real spooling device 206
  as a virtual spooling device 212
  character arrangement tables for Model 1 and 3 209
  character arrangement tables for Model 3 only 209
  commands 205, 212
  features 203
  modules 203
    constructing 207
    creating 207
  printing a spool file 206
    PRINT command with TRC option 206
  virtual
    defining 205
    displaying control information 205
    loading modules via SETPRT command 206
    recovery from I/O errors 212
3800, modifying named systems 210

These symbols are used in the index to refer to other VM and VM/SP references for system programming:
CMSPROG — VM/SP CMS for System Programming
SFPROG — VM System Facilities for Programming
DIAG — VM Diagnosis Guide

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

File No. S370/4300-36
Printed in U.S.A.

SC24-5285-0



VM/SP
CP for System Programming
Order No. SC24-5285-0

READER'S COMMENT FORM

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

_____ _____ _____ **Help Information line** ____ **of** ____

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? ___YES ___NO

Please print your name, company name, and address:

_____

_____

_____

_____

IBM Branch Office serving you: _____

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

SC24-5285-0

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

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

File No. S370/4300-36
Printed in U.S.A.

SC24-5285-0

SC24-5285-00

IBM®