File No. S370-30
Order No. GC20-1821-0

Systems

IBM Virtual Machine Facility/370: Operating Systems in a Virtual Machine

Release 3 PLC 1

This publication is intended for VM/370 users who plan to use any of the System/360 or System/370 operating systems that are supported under VM/370. As such, it is directed to the system programmer, the system operator, as well as to the general user. The manual includes sections on how to configure your virtual machine, how to use the VM/370 commands, and how to perform your system generation under VM/370. Other sections cover the running of DOS/VS and OS/VS under VM/370. Still other sections contain system planning and general operating information and considerations.

Users of the Conversational Monitor System (CMS), the Remote Spooling Communications Subsystem (RSCS), or the Interactive Problem Control System (IPCS) are directed to the respective User's Guides.

Prerequisite Publications

IBM Virtual Machine Facility/370:
Introduction, Order No. GC20-1800
Terminal User's Guide, Order No. GC20-1810

IBM

First Edition (February 1976)

This edition corresponds to Release 3 PLC 1 (Program Level Change) of IBM Virtual Machine Facility/370, and to all subsequent releases unless otherwise indicated in new editions or Technical Newsletters.

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

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, VM/370 Publications, 24 New England Executive Park, Burlington, Massachusetts 01803. Comments become the property of IBM.

Preface

This publication provides the general user, the system operator, and the system programmer with planning and operational information on running any of the supported System/360 and System/370 operating systems under control of VM/370.

Users of the Conversational Monitor System (CMS) should refer to the IBM Virtual Machine Facility/370: CMS User's Guide, Order No. GC20-1819.

Users of the Remote Spooling Communications Subsystem (RSCS) should refer to the IBM Virtual Machine Facility/370: Remote Spooling Communications Subsystem (RSCS) User's Guide, Order No. GC20-1816.

Users of the Interactive Problem Control System (IPCS) should refer to the IBM Virtual Machine Facility/370: Interactive Problem Control System (IPCS) User's Guide, Order No. GC20-1823.

This publication comprises nine sections grouped into two parts.

Part 1 contains five sections and includes general day-to-day usage information for the system operator and general user.

"Section 1. The Virtual Machine Facility/370" is an introductory section where the features of VM/370 are discussed as they apply both to the virtual machine and to the operating system that is running in it.

"Section 2. General Considerations" contains general information on how to use VM/370 facilities more efficiently when running your operating systems under VM/370.

"Section 3. General Operating Procedures" discusses those aspects of running operating systems under VM/370 that are common to all systems. The use of CP commands to control terminal sessions, control I/O devices, test and debug programs, control virtual machine functions, measure performance and set performance options is explained in this section.

"Section 4. DOS/VS in a Virtual Machine" contains operating information specific to DOS/VS and DOS.

"Section 5. OS/VS in a Virtual Machine" contains planning and operating information specific to OS/VS1 and OS/VS2.

Part 2 contains four sections and includes planning and system generation information and special considerations, primarily for the use of the system analyst and system programmer.

"Section 6. System Planning Considerations" contains information useful to the system programmer when planning his system for use under VM/370 rather than standalone.

"Section 7. Configuring Your Virtual Machine" discusses the directory entry control statements that define the virtual machines that will be running your operating system.

"Section 8. Generating Your Operating System under VM/370" covers the system generation of your operating systems using the VM/370 facilities of the CMS Editor, CMS EXEC, and the spool system.

"Section 9. Special Considerations" discusses unusual conditions and special situations affecting a relatively small subset of users.

This publication also contains two appendixes.

"Appendix A: Language Processors and Emulators" itemizes certain other facilities that are not part of VM/370 but can be ordered separately from IBM. These include Program Products, Installed User Programs, and Integrated Emulators.

"Appendix B: CP Restrictions" lists virtual machine restrictions and certain execution characteristics that must be considered when running operating systems in a virtual machine.

TERMINOLOGY

In this publication, the following terminology is used:

- 2305 refers to the IBM 2305 Fixed Head Storage, Models 1 and 2.

- 3270 refers to both the IBM 3275 Display Station, Model 2 and the IBM 3277 Display Station, Model 2.
- 3330 refers to the IBM 3330 Disk Storage Models 1, 2, 11; the IBM 3333 Disk Storage and Control Models 1 and 11; and the 3350 Direct Access Storage operating in 3330/3333 Model 1 or 3330/3333 Model 2 compatibility mode.
- 3340 refers to the IBM 3340 Disk Storage, Models A2, B1 and B2; and, the 3344 Direct Access Storage, Model B2.
- 3350 refers to the IBM 3350 Direct Access Storage, Models A2 and B2, in native mode.

Information on the IBM 3344 Direct Access Storage Device and the IBM 3350 Direct Access Storage contained in this publication is for planning purposes only until the availability of the product.

Any information pertaining to the IBM 2741 terminal also applies to the IBM 3767 terminal, Model 1, operating as a 2741, unless otherwise specified.

For a glossary of VM/370 terms, see the IBM Virtual Machine Facility/370: Glossary and Master Index, Order No. GC20-1813.

PREREQUISITE PUBLICATIONS

IBM Virtual Machine Facility/370:

Introduction, Order No. GC20-1800

Terminal User's Guide, Order No. GC20-1810

COREQUISITE PUBLICATIONS

IBM Virtual Machine Facility/370:

CP Command Reference for General Users, Order No. GC20-1820

CMS User's Guide, Order No. GC20-1819

CMS Command and Macro Reference, Order No. GC20-1818

Operator's Guide, Order No. GC20-1806

COREQUISITE PUBLICATIONS (if required)

IBM Virtual Machine Facility/370:

Remote Spooling Communications Subsystem (RSCS) User's Guide, Order No. GC20-1816

References in the text to titles of related VM/370 publications will be given in abbreviated form.

The reader must also have a basic knowledge of the operating systems he will be using under VM/370. For the titles and abstracts of the appropriate publications, refer to the latest IBM System/370 Bibliography, Order No. GC20-0001.

ASSOCIATED PUBLICATIONS

IBM Virtual Machine Facility/370:

System Programmer's Guide, Order No. GC20-1807

Planning and System Generation Guide, Order No. GC20-1801

OLTSEP and Error Recording Guide, Order No. GC20-1809

Interactive Problem Control System (IPCS) User's Guide, Order No. GC20-1823

SUBMITTING INFORMATION ON RUNNING OPERATING SYSTEMS IN A VIRTUAL MACHINE

You are welcome to submit recommendations and hints about generating and running operating systems in a virtual machine for possible inclusion in this publication. You can either use the Suggestion Input Form or the Reader Comment Form at the back of this publication, or send your recommendations to:

IBM Corporation
VM/370 Publications Department
24 New England Executive Park
Burlington, Mass. 01803

It is understood that IBM and its affiliated companies shall have the nonexclusive right, at their discretion, to use, copy, and distribute any or all submitted information or material, in any form, for any and all purposes, without any obligation or commitment to the submitter, and that the submitter has the right to submit such information or material upon such a basis.

When submitting recommendations, indicate the type of operating system (that is, DOS/VS, VS1) and the release level that is used under VM/370.

GUIDELINES AND PROCEDURES

Suggestions for generating and running operating system can cover any topic currently in the book or can address areas that you think should be added to this publication. Input does not have to be submitted on a Reader Comment Form or necessarily be a formal paper on the subject. Submit input in whatever manner and format is most convenient, but ensure that it is legible, understandable, and follows these guidelines:

- Do not include ZAPs or SUPERZAPs to object modules since such modules are likely to change over time.
- Do not describe a problem unless you have an appropriate solution, circumvention, or alternative included. Tips on things you have to watch out for, or unusual circumstances that can occur in virtual machine operation are, however, suitable topics.
- If you describe useful EXEC procedures, or CMS or other programs, test them out to ensure that they work.

Note: The recommendations in the DOS/VS and OS/VS chapters of this publication are meant to help an installation in generating operating systems to run more efficiently under VM/370 and also contain operational considerations or hints when using virtual machines. Many of these recommendations were suggested by VM/370 users and have not been submitted to any formal IBM test. As a result, potential users should evaluate the applicability of the recommendations to their installation before implementation.

Contents

Figures

Part 1. General Usage Information

Part 1 contains general usage information applicable to the system operator and general user as they control their operating systems in virtual machines running under VM/370.

Part 1 contains the following topics:

Section 1. The Virtual Machine Facility/370

- Introduction
- The Virtual Machine
- Error Recovery Management
- Other System Features

Section 2. General Considerations

- Uses of Virtual Machines
- Programming Considerations
- Virtual Machine Options
- Multiprogramming Systems under VM/370
- Other Considerations

Section 3. General Operating Procedures

- The CP Command Language
- Controlling a Terminal Session
- Controlling Input and Output Functions
- Controlling the Virtual Machine
- Testing and Debugging of Programs
- Using Alternating Operating Systems
- Performance Measurements and Options

Section 4. DOS/VS in a Virtual Machine

- Accessing and Loading the DOS/VS System
- System Operation
- Running a Batch DOS/VS System
- Alternating Operating Systems
- Concurrent Use of Multiple Systems

Section 5. OS/VS in a Virtual Machine

- Accessing and Loading the OS/VS System
- System Operation
- Running a Batch OS/VS System
- Alternating Operating Systems
- Concurrent Use of Multiple Systems

Section 1. The Virtual Machine Facility/370

Section 1 consists of brief summaries of the general concepts of VM/370. They are included here as background and reference material for the VM/370 user whose virtual machine is being controlled by one of the operating systems supported by VM/370. For an in-depth discussion of VM/370 concepts, see the VM/370: Introduction.

Section 1 contains the following information:

Introduction
- Defining a Virtual Machine
- VM/370 Components and Supported Operating Systems

The Virtual Machine
- Virtual Console
- Virtual CPU
- Virtual Storage
- Virtual Input/Output Devices

Error Recovery Management
- Hardware Recovery
- VM/370 Recovery
- Virtual Machine Recovery
- Error Recording

Other System Features
- Virtual Machine Accounting
- Saved Systems
- Shared Systems
- Discontiguous Saved Segments
- Shared Segment Protection

Virtual Machine Facility/370 (VM/370) is a system control program (SCP) that manages the real resources of an IBM System/370 in such a way that they are made available to multiple concurrent users. Each user has at his disposal the functional equivalent of a real CPU, main and auxiliary storage, and input/output devices. Since this functional equivalent is not real but only simulated by VM/370, it is referred to as a "virtual" machine.

Virtual machine resources are either shared among users or allocated to users alternately for a specified time period. The resources to be allotted to any particular virtual machine are specified in that virtual machine's directory entry. The directory entries for all virtual machines make up the VM/370 directory file usually located on the VM/370 system residence volume. When a user obtains access to the VM/370 system, a virtual machine is created based upon that user's directory entry. The user may then load any of the supported operating systems and begin processing.

DEFINING A VIRTUAL MACHINE

The configuration of a user's virtual machine is stored as a set of control statements, comprising the directory entry for that virtual machine in the VM/370 directory file. These control statements specify the user identification and password, the virtual CPU storage size, the virtual machine I/O configuration, accounting information, and other optional information required by the operating system to control the virtual machine.

The I/O configuration includes the virtual machine console, virtual unit record devices, and virtual disk space. Tape drives cannot be shared and are not defined in the VM/370 directory unless they are permanently required for a virtual machine. Usually, whenever a user requires a tape drive, the system operator temporarily attaches an available real device to the virtual machine. Many virtual disks can occupy space on one real DASD. Many virtual printers or punches can share a real printer or punch. On the other hand, tape devices must be defined on a one-to-one basis and be dedicated to a virtual machine.

Sometimes, the virtual machine's operating system provides support for a real device that, although available to the System/370, is not supported by VM/370. These devices can be entered in the directory as either dedicated or special devices and used by the virtual machine. They must be specified to the VM/370 nucleus via the RDEVICE macro with the appropriate CLASS operand (that is, DASD, TAPE, TERM, GRAF, URI, or URO). Control of both the real and virtual device then becomes the responsibility of the virtual machine's operating system.

A discussion of VM/370 directory entries for supported operating systems along with specific examples is included in "Section 7. Configuring Your Virtual Machine."

VM/370 COMPONENTS AND SUPPORTED OPERATING SYSTEMS

The VM/370 system consists of four elements: the Control Program (CP), the Conversational Monitor System (CMS), the Remote Spooling Communications Subsystem (RSCS) and the Interactive Problem Control System (IPCS).

The Control Program (CP) is that portion of VM/370 that controls the resources of the real machine and makes them available to each virtual machine as required. While CP manages the multiple virtual machine environment in the real machine, it is also necessary to have an operating system managing the work flow within each virtual machine. Under VM/370, virtual machines operate completely independently of one another. This allows the concurrent use of many different operating systems, as well as different releases of the same operating system.

THE CONVERSATIONAL MONITOR SYSTEM (CMS)

The Conversational Monitor System (CMS) is a major component of VM/370 and runs under control of CP. It cannot be operated in standalone mode; that is, without CP on a real machine. CMS provides the user with a comprehensive interactive time-sharing facility. With CMS, the user can create, update, and manipulate files as well as compile, test, and execute problem programs. These interactive capabilities are extended to DOS/VS users via the CMS/DOS environment of CMS. For OS/VS users, a combination of CMS commands and CMS simulation of OS macros provides similar interactive capabilities. For information on using the CMS virtual machine, refer to the VM/370: CMS User's Guide and the VM/370: CMS Command and Macro Reference.

THE REMOTE SPOOLING COMMUNICATIONS SUBSYSTEM (RSCS)

The Remote Spooling Communications Subsystem (RSCS) is a component of VM/370 and runs under control of CP. It cannot be operated in standalone mode; that is, without CP on a real machine. RSCS, together with the spool file system of CP, provides a telecommunication environment for the transfer of files between:

- VM/370 users and RSCS remote stations
- RSCS remote stations and other RSCS remote stations
- VM/370 users and remote job entry batch systems
- RSCS remote stations and job entry batch systems
- RSCS remote stations and a CMS Batch virtual machine

Examples of remote job entry batch systems are HASP, ASP, POWER, JES, RES, and CRJE, running under the appropriate system control program, DOS, DOS/VS, OS, or OS/VS.

For information about using the RSCS virtual machine, refer to the VM/370: Remote Spooling Communications Subsystem (RSCS) User's Guide.

THE INTERACTIVE PRCBLEM CONTROL SYSTEM (IPCS)

The Interactive Problem Control System (IPCS) is a component of VM/370 that standardizes the VM/370 program problem reporting process through

an online data base. Additional environmental conditions can now be included in problem reports; user detected problems can be entered, updated, and printed out; and the problem data base can be interrogated for individual as well as related problems. For information regarding this component of VM/370, see the VM/370: Interactive Problem Control System (IPCS) User's Guide.

VM/370 IN A VIRTUAL MACHINE

A VM/370 system can operate in a virtual machine under control of another VM/370 system running on the real machine. This allows the current level of VM/370 to be updated and the new system tested in a virtual machine while the current version on the real machine is processing its regular daily work. When testing has been completed, the new system can be transferred from the virtual machine to the real system disk while processing continues. The next time VM/370 is loaded, the new system will be in effect.

For a detailed discussion on VM/370 under VM/370 along with annotated examples of use, see the VM/370: System Programmer's Guide.

SUPPORTED OPERATING SYSTEMS

Operating systems that operate standalone as well as being supported under VM/370 fall into two general classifications.

The batch or single-user systems are represented by DOS, DOS/VS, OS/PCP, OS/MFT, OS/MVT, OS/VS1, and OS/VS2. With the exception of OS/PCP, these are all multiprogramming systems. However, when operating in a virtual machine under VM/370, the user has the choice of running multiple partitions in one virtual machine similar to standalone operation, or single partitions in multiple virtual machines. When running multiple partitions in one virtual machine, multiprogramming and unit record spooling is done by both the operating system and VM/370. This may decrease the overall efficiency of the virtual machine. When running single partitions in multiple virtual machines, the need for multiple virtual storages places a burden on auxiliary storage. This can be alleviated, however, by using shared systems; the concept of shared systems is discussed under "Other System Features."

Multiple-access systems are represented by VM/370 itself and the Time Sharing Option of OS/MVT and OS/VS2. A multiple-access system operates in one virtual machine and supports multiple interactive users. The multiple-access virtual machine must first gain access to VM/370 via the LOGON command. Subsequently, the interactive users can connect to the multiple-access system via the DIAL command or by using a terminal on a dedicated line. Communication between the two is carried out using the command language of the multiple-access system.

OTHER PROGRAMS AND SYSTEMS

For information about other programs and systems that have been used under VM/370, request information on Installed User Programs (IUPs) and Field Developed Programs (FDPs) from your local IBM branch office.

Among those available are MUSIC (McGill University System for Interactive Computing), SCRIPT/370 (a text-processing program), and VM/SGP (a statistics-generating package for VM/370).

The virtual machine is created by VM/370 from the control statements that make up that machine's directory entry. The statements generally define the following:

- Virtual Console
- Virtual CPU
- Virtual Storage
- Virtual Input/Output Devices

VIRTUAL CONSOLE

A real terminal (of a type supported by VM/370) becomes the user's virtual console and his primary interface with VM/370 and his operating system. Many of the real control panel functions, such as IPL, START, INTERRUPT, DISPLAY, and ENTER controls, and instruction address stop, are simulated by CP commands entered via this virtual console. The communication that normally occurs between an operator and the operating system in a real machine also exists in a virtual machine. Commands to the operating system are entered at the terminal and messages and responses from the operating system are returned to the user at the terminal.

In addition to the real control panel functions, VM/370 provides all virtual machines with tracing and debugging facilities in an interactive mode. This can be especially useful when running an operating system that does not include conversational capabilities. DOS, for example, does not include commands for online debugging.

VIRTUAL CPU

The virtual CPU that VM/370 provides to each virtual machine is, in reality, the shared use of the real CPU. VM/370 manages the distribution of its CPU resources via the dispatching and scheduling functions.

The dispatching function maintains two lists of dispatchable virtual machines or users: an interactive set and a noninteractive set. The total number of these virtual machines that are allowed to compete for the real CPU is governed by the real storage size. The interactive users are those usually running a conversational type operating system such as CMS or VSAPL. These users are characterized by regular and frequent terminal input and output operations. The noninteractive users are those usually running a batch type operating system such as DOS/VS or OS/VS. These users are characterized by heavy CPU usage and relatively high amounts of data transfer. The interactive users are considered for dispatching before the noninteractive users. When a user has used up his allotted slice of real CPU time, he is dropped from the dispatchable list and placed on the eligible list.

The scheduling function maintains two lists of virtual machines that are eligible to be added to the dispatchable lists but are not allowed to compete for the real CPU until some user is dropped from the dispatchable list. These two lists are again classified as interactive

and noninteractive, with the interactive user receiving the higher priority when the opportunity to move to the dispatchable list arises.

VIRTUAL STORAGE

Each virtual machine can have associated with it, up to 16 megabytes of virtual storage. The normal and maximum storage sizes are defined in the virtual machine configuration in the VM/370 directory. Virtual storage, from the standpoint of VM/370, is logically divided into 4096 byte areas called pages. Whether or not the virtual machine sees its storage as paged or unpaged depends on the operating system being used. DOS/VS, OS/VS, and VM/370, operating in extended control mode, can create one or more virtual storages of their own and control them in a paging environment. DOS, OS/PCP, OS/MFT, and OS/MVT treat their virtual storage as unpaged, that is, real storage.

Since real storage is usually much smaller than the combined virtual storage of all the active users, only certain referenced virtual storage pages for each user, called a working set, are kept in real storage. The philosophy of bringing a page into real storage when referenced by a program is called demand paging. Inactive virtual storage pages are kept on a direct access storage device known as the paging device.

The multiple virtual storages that exist under VM/370 are managed by the VM/370 control program (CP) via the use of a unique set of page and segment tables for each virtual machine. These tables relate virtual storage addresses to real storage locations. Since these tables are accessible only to VM/370, one virtual machine operating system cannot access or alter the virtual storage of another virtual machine. This form of storage protection is extended to any number of virtual machines, in contrast to the limited number of storage spaces that can be protected via storage keys.

VIRTUAL INPUT/OUTPUT DEVICES

Virtual devices make up the major portion of a virtual machines configuration as defined in the VM/370 directory. Each virtual machine definition must include the proper number and type of I/O devices that are required by the operating system(s) being run in that virtual machine. Virtual input and output devices are defined by the user with virtual address as required by the operating system. VM/370 allows the use of the same virtual address by any number of users. Whenever a virtual machine is executing input or output, VM/370 translates the virtual address to its real counterpart. The following types of virtual-to-real device mappings are supported by VM/370:

- Dedicated Devices
- Virtual Unit Record Devices
- Virtual Disks
- Virtual Transmission Control Units (Virtual Lines)
- Channel-to-Channel Adapters

DEDICATED DEVICES

Dedicated devices are those in which a one-to-one correspondence exists between the real and virtual devices. Tape devices are always

dedicated. Disks, terminals, and unit record devices may be dedicated to a virtual machine in order to improve the performance of that virtual machine. Care should be taken that restricting the use of these devices to one virtual machine does not impair the VM/370 system operation with regard to the other virtual machines.

A real device that is dedicated to a virtual machine is completely controlled by that virtual machine. Dedicated devices can be assigned dynamically via CP commands, assigned at logon time via directory control statements, or established via dialed connections to virtual 270x telecommunications control units or virtual lines. This relationship can be used to advantage in the operation of devices that, while available to the System/370, are not supported by VM/370. Take, for example, a user running DOS/VS who wants to use a paper tape reader, a device not supported by VM/370. By dedicating the device to his virtual machine, VM/370 control is bypassed and DOS/VS performs all the I/O operations.

VIRTUAL UNIT RECORD DEVICES

For operating systems that have modest input and output device requirements, the sharing of unit record devices is advantageous. VM/370 uses its spooling facility to temporarily store real input and virtual output data on direct access storage space. Real input data is then transferred to the appropriate virtual machine reader and virtual output is processed by the real printer and/or punch.

VIRTUAL DISKS

A real disk can have its storage space shared by many virtual machines in such a way that each user thinks he has access to a full disk. These virtual disks, also called minidisks, can range in size from 1 cylinder to the size of the real device. Utility programs are provided by VM/370 to format these minidisks for the appropriate operating system being used; CMS-FORMAT for CMS and IBCDASDI for OS/VS and DOS/VS.

VM/370 also allows the controlled concurrent use of a virtual disk cr minidisk by more than one virtual machine. Certain disks, such as those used for VSAM data sets, can be shared by virtual machines running different operating systems; DOS/VS, OS/VS, and CMS. The owner of the minidisk can specify whether other virtual machines are to have read or read/write access to his disk and whether they must supply a password to validate their requests. VM/370 then monitors each access request. Usually, only one virtual machine is given write access to a shared disk to avoid updating conflicts.

VIRTUAL TRANSMISSION CONTROL UNITS (VIRTUAL LINES)

A real 270x transmission control unit (TCU), or a 370x when used in 270x emulation mode, can be used by multiple virtual machines, each being given a virtual 270x TCU composed of a subset of the real TCU ports. Disks and 270x transmission control units are the only real devices that can have subsets for virtual-to-real device mapping.

CHANNEL-TO-CHANNEL ADAPTER

A virtual channel-to-channel adapter may be defined with or without a real equivalent device. A real channel-to-channel adapter, dedicated to a virtual machine, makes it possible for that virtual machine to communicate with another real or virtual machine. A virtual machine running as an OS ASP main processor can use this method to communicate with an ASP support processor running on a real machine other than the VM/370 host. Alternatively, a complete ASP system can be tested in a virtual machine environment by running both the main and support processors in two virtual machines connected to each other via virtual or real channel-to-channel adapters. If they are connected via a virtual channel-to-channel adapter, a real channel-to-channel adapter is not necessary.

Whenever a machine malfunction or software error occurs, the responsibility for attempting recovery falls into one of three areas:

- Hardware recovery
- VM/370 recovery
- Virtual machine recovery

HARDWARE RECOVERY

The System/370 systems supported by VM/370 have built-in error detection logic in the processor and main storage. This detection logic, working with additional hardware logic, allows the system to attempt the correction of certain CPU error conditions and single bit main storage errors within a doubleword. When errors are correctable, they are referred to as soft errors and have no adverse effect on VM/370. They are also generally transparent to the virtual machine's operating system.

When errors are not correctable, hardware-initiated machine check interrupts invoke the Recovery Management Subsystem (RMS) of VM/370.

VM/370 RECOVERY

Error recovery by VM/370 falls into two categories: (1) recovery from machine malfunctions by the Machine Check Handler and (2) recovery from channel errors associated with I/O events (except those initiated by virtual machines) by the Channel Check Handler.

MACHINE MALFUNCTIONS

When VM/370 is notified that a machine malfunction could not be corrected by the hardware recovery features, it records the error and attempts a software recovery at one of four levels.

Functional Recovery

Functional recovery, like a successful hardware recovery, results in no adverse effects on either VM/370 or the interrupted user.

System Recovery

System recovery is attempted when functional recovery is not possible. The user experiencing the error is terminated in order to allow VM/370 to continue processing all other users. If the user in error is vital to VM/370 operation, recovery is handled by the next level.

System-Supported Restart

When the error occurs in a critical routine, the system operator is notified that VM/370 cannot continue. An automatic restart of the system is performed.

System Repair

This form of recovery is the least acceptable. System-supported restart may have been tried unsuccessfully several times and the operator must now obtain the services of maintenance personnel.

CHANNEL ERRORS

When a channel error associated with some I/O event occurs, the error is recorded and a device-dependent recovery procedure is initiated. If the Channel Check Handler determines that the system integrity has been damaged, the operator is notified and the system is placed in a disabled wait state. The system operator may then initiate a system-supported restart.

VM/370 does not perform any error recovery for virtual machine initiated events; in fact, channel control or interface checks will cause the reset of the virtual machine after the user has been notified of the failure. The user must then reload the operating system in order to continue. Channel errors that do not jeopardize the system are recorded and then reflected to the virtual machine's operating system.

VIRTUAL MACHINE RECOVERY

Since each virtual machine is a functional equivalent of a System/370 and its associated I/O devices, VM/370 normally reflects virtual machine I/O errors to the virtual machine that initiated the I/O event in the same manner as the error would be reflected if the user were running standalone on a real machine. Device error recovery, error recording, and error messages are then dependent upon the virtual machine's operating system release level. An exception to the above procedure occurs when the virtual operating system uses VM/370's Diagnose Interface to initiate an I/O event. VM/370 would then handle the recovery for I/O errors.

ERROR RECORDING

An operating system in a dedicated environment (for example, DOS/VS operating standalone in a System/370 Model 145) exercises complete control over the entire hardware configuration. There is a direct relationship between the address of the real device and the address used to access data on that device. Error recording by an operating system running standalone produces error records that contain factual addresses of the devices that are malfunctioning.

The same operating system, when running in a virtual machine under VM/370, presents a different situation. For some devices, the device address and data address may be virtual and not real. For example, data that DOS considers to be located on track 4 of cylinder 15 of a 2314 with a device address of 230 could, in reality, be located on track 4 of cylinder 35 of a 2314 with a device address of 330. A virtual 3330, Model 1 mapped onto a real 3330, Model 11 would be another example. Other devices, whether or not supported by VM/370, can be dedicated to an operating system, in which case, VM/370 does not perform any data address or device type translation. Device address mapping, however, may still be done.

When a virtual machine initiated I/O event fails, the I/O error is reflected to the virtual machine's operating system. However, the accompanying sense data is associated with the virtual device and recording this data may not be of much help to system support personnel who really need data regarding the real device causing the error. The SVC 76 interface capability of VM/370 takes care of this problem.

SVC 76 HANDLING

SVC 76 is the Supervisor Call used by IBM operating systems to effect the recording on disk of either statistical data or permanent I/O error events. The following systems support SVC 76:

VM/370 (running in a virtual machine) (Release 2 and 3 only)
OS/360 (Release 21.7 and above)
OS/VS1 (Release 3.0 and above)
OS/VS2 (Release 1.6 and above)
DOS (Release 27 with required PTF)
DOS/VS (Release 28 and above)

VM/370 will trap a valid SVC 76 when it is issued by an operating system running in a virtual machine. Valid SVC 76 events are those involving permanent I/O error records and OS/VS2 Release 2 multiple virtual storage programming abend records.

VM/370 checks the error recording data parameters passed with the SVC 76. If invalid, the SVC 76 is reflected to the virtual machine's operating system for disposition. If valid, VM/370 will:

- Resolve device address differences
- Record the error data on VM/370's Error Recording cylinders
- Inform the VM/370 system operator of the error
- Return control to the virtual machine's operating system at the instruction address following the SVC 76 instruction.

By returning control to the instruction following the SVC 76, VM/370 prevents the double recording of the error data.

When the above-mentioned operating systems are run standalone, SVC 76s are reflected to the operating system's supervisor for recording on the LOGREC data set on the system residence device.

In most cases, the SVC 76 and the error recording is transparent to the jobs being processed under the operating systems. Exceptions would be in certain situations such as the RETRY or CANCEL options in DOS, or when rewinding 3420 tape drives under OS/VS1.

In addition to the error recovery and CP command features, VM/370 contains several other features that expand the capabilities of operating systems running in virtual machines. They are:

- Virtual machine accounting
- Saved systems
- Shared systems
- Discontiguous saved segments
- Shared segment protection

VIRTUAL MACHINE ACCOUNTING

VM/370 keeps track of a virtual machine's usage of system facilities and generates accounting cards whenever the use of some chargeable resource is terminated. For operating systems that do not already have job accounting built in, this would be one way of having VM/370 gather job accounting data. A special DIAGNOSE instruction code, X'4C', is also provided so that an operating system that is servicing multiple customers can generate accounting data below the virtual machine level, for example, at a customer or job level.

Accounting data records, when generated, are stored in real storage and remain there until punched out on the real punch. The system operator with control over the spooling devices, should ensure that the real punch is periodically started in order to punch out accumulated accounting records. Failure to do so will gradually tie up more and more real storage and impact system performance.

VIRTUAL MACHINE ACCOUNTING RECORD

Whenever a virtual machine logs off the system or issues a DIAGNOSE code X'4C', an accounting card is created in real storage containing userid, account number, connect time, virtual CPU time, plus paging and spooling statistics.

DEDICATED DEVICE ACCOUNTING RECORD

Whenever a previously dedicated device or temporary disk space is released by a user, or whenever a DIAGNOSE code X'4C' is issued, an accounting card is created in real storage containing userid, account number, device identification, amount of storage of temporary disk space being accounted for, and system connect time for the device.

A dedicated device is one that has been attached or dedicated to a virtual machine for its exclusive use. Temporary disk space is a portion of system auxiliary storage that is assigned to a virtual machine for its exclusive use during the current terminal session only.

USER FORMATTED ACCOUNTING RECORD

A user may format up to 70 bytes of information and issue a DIAGNOSE code X'4C' with a function code of X'10'. VM/370 will create an accounting card with the user's identification, the supplied data, and a special code distinguishing this card from VM/370 originated accounting cards.

SAVED SYSTEMS

When an operating system is initially loaded into a virtual machine by device address, VM/370 goes through a process of reading the resident nucleus into real storage and writing it back out to the system paging device, all the while updating the virtual machine's paging tables.

This process is repeated each time the system is loaded and, depending upon the operating system, can consume sizable amounts of both time and system resources.

VM/370 allows an installation's system programmer to save the results of an initial program load at a point determined by the type of operating system and assign this saved system a name. This procedure consists of two steps. A system name table entry is generated via the NAMESYS macro as part of the assembly of the DMKSNT module. This entry creates an area on some CP-owned device that is to contain a page-formatted copy of the system that is to be saved. A user-supplied name is also associated with this area and will be used both for saving the system and subsequent loading. The second step consists of loading the system normally (that is, by device address), stopping the loading process at a point predetermined by the operating system and unique installation use, and then issuing the SAVESYS command to save the system. Subsequently, the system can be loaded by name, for example:

```
ipl disdos
```

rather than by device address:

```
ipl 250
```

When an operating system is loaded by name, it is already page-formatted and each page is brought into storage on demand rather than loading the entire nucleus. This process saves both time and system resources. Under certain conditions, a saved system can be shared among a number of virtual machines with further savings on system resources.

SHARED SYSTEMS

A saved system has the additional capability of having reentrant portions of its virtual storage sharable among many concurrently operating virtual machines.

The virtual storage area to be shared must begin on a segment boundary and must contain only reentrant code. In addition, shared segments must be part of a saved system.

The advantages realized by the use of shared systems are many. Regardless of the number of virtual machines using the shared segments,

only one copy of the pages in the shared segment need ever occupy real storage and external page storage. This reduces the total amount of real storage and auxiliary storage required; or, looking at it another way, more efficient use can be made of available storage. The greater the number of virtual machines that are using a shared system, the greater the storage savings and the greater the probability that the shared pages will be frequently referenced. Pages that are frequently referenced tend to remain in real storage, thereby reducing paging activity. Less paging activity increases the efficiency of the CPU in performing productive work by reducing CPU overhead.

Note: Shared systems cannot be loaded into the V=R area of VM/370. An attempt to do so will result in an error message.

DISCONTIGUOUS SAVED SEGMENTS

Discontiguous saved segments (DCSS) are similar in many ways to the saved systems previously discussed. They must have been named, loaded, and saved as when saving a system with the added condition that the discontiguous saved segments must be loaded at an address higher than the highest address of the virtual machine that is to use it. It must be segment boundary aligned; it must be a multiple of a segment size with a maximum of 78 segments; and, if it is to be shared, it must contain only reentrant code.

A discontiguous saved segment may be logically attached by a virtual machine whenever it is needed and detached when it is not needed. The CP instructions necessary to perform the linkage must be part of the virtual machine's operating system.

An example of a discontiguous saved segment, that is also shared, is the CMS segment that supports DOS program development and testing under CMS. This segment, called CMSDOS, is used, as needed, by all VSAM and CMS/DOS users. If CMSDOS were not a discontiguous shared segment, each CMS user would require segment table entries for this facility for the entire terminal session. Use of the DCSS function means that real storage for these segments need only be allocated while the CMSDOS facility is actually being used by the virtual machine.

Note: Systems using discontiguous saved segments cannot be loaded into the V=R area of VM/370. An attempt to do so will result in an error message.

SHARED SEGMENT PROTECTION

VM/370 protects the multiple users of shared segments from any intentional or inadvertent alteration of the shared segment's code or data. Before CP selects a new user to be dispatched, it checks that the current virtual machine has not altered any pages that are in shared segments. If the current virtual machine has altered any shared page, the virtual machine is placed in nonshared mode and the altered system is assigned to the virtual machine that altered it. A fresh copy of the shared system is then loaded and new paging tables are set up for the remaining shared system users.

Also, whenever a shared page is stolen to satisfy another more recent storage requirement, the paging routine checks whether that page has been modified. If it has, the user responsible is assigned the altered system in nonshared mode, and a fresh copy of the system is provided to the remaining users.

Section 2. General Considerations

Section 2 contains general information on the VM/370 facilities available to you and your operating system when running in a virtual machine. A basic understanding of these facilities is necessary in order to obtain the best possible efficiency out of the combination of your virtual machine and your operating system.

Section 2 contains the following topics:

Uses of Virtual Machines

- Systems Programming
- Application Programming
- Operations

Programming Considerations

- Locality of Reference
- Abends in a Virtual Machine
- Reducing the Virtual Machine's I/O Operations

Virtual Machine Options

- The REALTIMER Option
- The ISAM Option
- The ECMODE Option
- The BMX Option
- Specifying the Options
- OS ISAM Channel Programs

Multiprogramming Systems under VM/370

- Page Waits
- I/O Waits
- Spooling Considerations
- Storage and CPU Utilization
- Virtual Machine I/O Management
- Minidisks

Other Considerations

- The Two-Channel Switch
- Unsupported Devices

CMS and any combination of operating systems such as OS/PCP, MFT, MVT, VS1, VS2, DOS, DOS/VS, and RSCS may be executed as operating systems in a virtual machine created and controlled by VM/370. Programs or systems to be run in a virtual machine may not perform adequately or consistently if they include any hardware or software timing dependencies. For example, if I/O devices (or the programs that control them) require program responses within a relatively short period of time for correct data transmission, they may not work correctly. Execution of programs under VM/370 that require dynamic modification of channel programs is allowed only in certain cases, such as OS ISAM; otherwise, they must be executed in a virtual machine with the VM/370 performance option called virtual=real. For example, an ISAM program to be used under OS/MFT, OS/MVT, or in a virtual=real region of OS/VS can execute in a virtual machine only if the ISAM option is specified in the VM/370 directory entry for that virtual machine or invoked via the CP SET ISAM command. Also, an OS/VS TCAM program can execute in a virtual machine that has OS/VS TCAM Level 5 generated or invoked with the VM/370 option. Each operating system under VM/370 control requires that the virtual machine have an I/O configuration compatible with the one for which the operating system was originally generated.

The environment of multiple virtual machines permits several of these systems to be executing concurrently. Production work may be run in one or more virtual machines while other virtual machines are executing:

- A terminal-oriented conversational system.
- A remote spooling subsystem, which transmits bulk data to and from remote locations.
- A back-release system running application programs not upgraded to the current level of the production system.
- A current-release system with new Program Temporary Fixes (PTFs) applied.
- A new release or option being generated and tested.
- Conversion testing from one operating system to another, for example, from DOS to OS/VS1.

The execution of conventional operating systems in an interactive virtual machine environment provides many convenient operating facilities in the areas of system programming, application programming, and operations.

SYSTEM PROGRAMMING

In the area of systems programming, an interactive virtual machine environment provides for:

- A reduction in the amount of dedicated systems time for hands-on testing on the real machine; therefore, less requirement for off shift testing.

- The testing of new or modified Supervisor Call (SVC) routines in a virtual machine.
- The application and testing of PTFs on a virtual machine.
- The generation and testing of new independent component releases (ICRs) or new releases of an operating sytem in a virtual machine.
- Hands-on console debugging as though on a dedicated real machine, via a terminal device.

APPLICATION PROGRAMMING

In the area of application programming, an interactive virtual machine environment provides for:

- The use of the CMS Editor to create source programs and data files.
- The use of the CMS UPDATE command and the CMS Editor to maintain source programs and data files.
- Debugging from a terminal while under operating system control.
- Faster turnaround time, more test shots per day, and a reduction in the length of the development cycle.
- The designing of application programs without real storage constraints.
- Defining minidisks and other virtual devices to design and test a slightly different or larger machine configuration before the actual hardware is installed.
- The use of SCRIPT/370, an Installed User Program for text preparation, to create and update the program specifications.

OPERATIONS

In the area of operations, an interactive virtual machine environment provides for:

- The training of operators in their own virtual machine isolated from production virtual machines.
- A shorter training period.
- The definition of a virtual machine and its devices as backup to some other real machine.
- The concurrent running of divergent installation requirements on a single real machine.
- A reduction in the manual handling and scheduling of user test shots by the operations staff. Users run their own virtual machines.

New application programs should be designed to operate efficiently in a paging environment. Whenever possible, you should use CP's paging mechanism instead of that of DOS/VS or OS/VS. That is, make the DOS/VS partitions and OS/VS regions V=R and large enough to contain the largest jobs. Eliminate all overlays and, if possible, combine multistep jobs that use temporary DASD storage into one larger job.

You should be aware that the following factors affect the performance of a virtual machine:

- The frequency of real interruptions that occur.
- The frequency and type of privileged instructions executed.
- Whether the virtual machine assist hardware is on the machine and enabled by both the system operator and by the user.
- The frequency of START I/O instructions.
- Locality of reference within virtual storage.

These factors are in addition to those described under "Performance Guidelines" in the System Planning Considerations Section.

LOCALITY OF REFERENCE

When a virtual machine refers to virtual storage addresses that are not in real storage, a paging exception, and paging activity, occurs. Routines that have widely scattered storage references, and thus a large working set size, tend to increase the paging load caused by this virtual machine.

When possible, modules of code that are dependent upon each other, as well as the related reference tables, constants, and literals, should be located in the same 4K page. Infrequently used routines, such as those that handle unusual error conditions, should not be placed near main routines. Reentrant coding techniques should be used whenever possible since unchanged page frames do not have to be written out.

ABENDS IN A VIRTUAL MACHINE

Whenever possible, particularly with virtual storage operating systems, use the virtual machine's dumping procedure instead of CP's. This is because CP's dump program will not print out second level storage pages (that is, V=V regions or partitions of OS/VS or DOS/VS machines) in the correct sequence. Pages that happen to be stored on the OS/VS or DOS/VS paging disk are not printed at all. Also, there are several special formatting dump programs available that make it easier for a user to trace through DOS/VS and OS/VS control blocks. Refer to the VM/370 System Logic and Problem Determination Guide or the VM/370: System Programmer's Guide for more information on debugging.

REDUCING A VIRTUAL MACHINE'S I/O OPERATIONS

The number of START I/O (SIO) instructions executed by a virtual machine may be substantially reduced:

- By making use of large (that is, 3-4K byte) blocking factors for I/O areas.
- By preallocating the DASD space if OS or OS/VS work data sets are being used.
- By using virtual storage instead of DASD work files for smaller temporary files. Build the information in virtual storage and let CP page the data out, if needed.
- By not assigning virtual printers, punches and readers to each partition or region in a virtual machine since records for these devices are unblocked. Use the virtual operating system's spooling mechanism, such as POWER/VS or JES; these facilities use large I/O areas and long chains of CCWs.

VM/370 provides several optional services to virtual machines.

THE REALTIMER OPTION

The REALTIMER option updates a virtual machine interval timer when that virtual machine is in a self-imposed wait state. This option is required for virtual machines running systems that wait for a timer interrupt to continue processing.

THE ISAM OPTION

The ISAM option allows the virtual machine to execute the self-modifying CCW command sequences generated by the OS ISAM modules in OS PCP, MFT, or MVT. This option is not required for the proper functioning of ISAM in DOS or OS/VS. However, if ISAM is run in the virtual=real area of an OS/VS virtual machine, the ISAM option is required. This option does not permit other types of self-modifying CCW sequences to function.

THE ECMODE OPTION

The ECMODE option allows the virtual machine to use the complete set of virtual System/370 control registers and the dynamic address translation feature of the System/370. Programming simulation and hardware features are combined to allow use of all the available features in the hardware. This option is required for DOS/VS, VS1, and VS2 virtual machines, and also when running VM/370 under VM/370.

THE BMX OPTION

The BMX (virtual block multiplexer) option allows an operating system that is running in a virtual machine to overlap multiple SIO (Start I/O) requests on a specified channel path. The selector channel mode is the normal (and default) channel mode for virtual machines. When the BMX option is invoked, it applies to all channels in the virtual machine except channel 0 or channels that have a channel-to-channel adapter (CTCA). This option can be specified regardless of whether block multiplexer channels are attached to the computer. The CP DEFINE command can redefine the channel mode for a virtual machine.

SPECIFYING THE OPTIONS

The REALTIMER, ISAM, and ECMODE options increase the amount of CP overhead incurred by the virtual machines using them, and should not be given to a virtual machine unless they are required. These options can

be specified either in the OPTION statement in the VM/370 directory or via the CP SET command. If a particular situation requires an option only occasionally, you should use the CP SET command and not the OPTION statement. In this way, the additional overhead is incurred only while the option is in effect.

The REALTIMER can be set off by a user via the CP SET TIMER command. Timers are described in more detail in "Section 6. System Planning Considerations."

For more information about these and the other options (ACCT, VIRT=REAL, and SVCOFF) that can be specified in the OPTION statement, see "Section 7. Configuring Your Virtual Machine."

OS ISAM CHANNEL PROGRAMS

Certain ISAM channel programs that run under OS/PCP, MFT, MVT, or in a V=R region of OS/VS use a self-modifying operation that is not allowed under normal VM/370 processing. With the ISAM option selected, CP can scan the specific ISAM channel program to handle the self-modifying sequence properly.

Only those users with the ISAM option in their VM/370 directory entry or who have issued the CP SET ISAM ON command have their CCW strings checked for self-modifying operation; thus, not all users incur the additional CP overhead. This option is not needed for DOS, nor for OS/VS ISAM when run in a V=V region.

When a multiprogramming system such as OS/VS or DOS/VS is run on a virtual machine, its resource algorithms interact with those of VM/370; especially when the virtual operating system has a page wait or I/O wait. There are two standard methods in which multiprogramming systems interact with VM/370: one method is without the benefit of a special communication path and the second method is with the special path. The communication path, called the VM/VS Handshaking feature, is only available for OS/VS1 virtual machines. A means of communicating with CP called the Diagnose Interface is available if your installation's system programmer would like to add such communication facilities to other operating systems. See the "Special Considerations" section for a summary of the Diagnose Interface, or refer to the VM/370: System Programmer's Guide for detailed information.

PAGE WAITS

If, during its execution, an OS- or DOS-created task or program must wait for a CP-provided service such as a virtual storage page, the virtual machine is marked nondispatchable even though other partitions or tasks in that virtual machine may be ready and available to run. Those other tasks in the virtual machine cannot be dispatched by the operating system until the CP page wait is satisfied. The net effect is that the highest priority program of the virtual machine gets almost all of the CPU time allocated to that virtual machine, if it can utilize the time. Therefore, programs running in the other partitions experience significant degradation.

When multiprogramming systems must be run in a virtual machine, make the partitions or regions as large as practicable and run all jobs V=R; also, consider using the VM/370 virtual=real option, reserved page frames, or locked pages. Note, however, that certain programs, such as the OS SORT/MERGE, will not run very efficiently if the region size is made too large. When the VM/370 virtual=real option is used, paging is eliminated for one virtual machine but this may adversely affect the paging performance of other virtual machines. The reserved page frames option tends to keep the most active pages in storage and the locked pages option locks the specified pages in storage.

I/O WAITS

On a real machine, when a task is waiting for an I/O operation to complete, the lower priority tasks are given use of the CPU. Under VM/370, the I/O operations of a particular virtual machine are overlapped with the CPU execution of that and other virtual machines. Consequently, lower priority tasks created by OS and DOS are given the CPU resource less frequently when executing in a virtual machine than when executing in a real machine. In some operating systems, such as OS/VS, you can specify time slicing if you want a group of regions to get roughly equal performance in each of the regions.

The system operator can issue the CP command SET FAVORED with no percentage value specified for a specific virtual machine to ensure that its lower priority DOS or OS tasks have a chance to execute. The

favored execution option reduces the effect of a variable system load on the favored virtual machine.

In addition, a percentage value (such as 50 percent) can be specified in the SET FAVORED command to guarantee a certain amount of CPU time is made available to the virtual machine, provided that it is able to use the time.

The system operator can also set the priority of a virtual machine by issuing the CP command SET PRIORITY. A low value gives the virtual machine a higher priority, and ensures that it is dispatched for execution more frequently than other virtual machines.

SPOOLING CONSIDERATIONS

Most multiprogramming operating systems provide their own spooling facilities. Since VM/370 also provides these facilities, double spooling can occur. If you have a significant amount of printing or punching to do, you may think one of the spooling facilities should be eliminated. This is not necessarily true. In fact, if the multiprogramming operating system's spooling facility blocks its output (such as OS/VS1 does), the most efficient spooling arrangement is usually to let both VM/370 and OS/VS1 spool. When a virtual machine operating system uses large I/O areas for its output, VM/370 has fewer SIOs to process. Note, however, that the increased thoughput with double spooling could be at the expense of an increase in elapsed time for some particular output file.

See the "General Operating Procedures" section for the CP spooling commands available. See the "System Planning Considerations" section for more detailed information about alternative approaches to reading, punching, and printing in a virtual machine.

If a virtual machine that does not have the handshaking feature or that does not use the Diagnose Interface shares real unit record devices with other virtual machines, the CP spool files are not closed until the virtual machine operator closes them. This means the CP spool files are not sent to the real printer or punch until the virtual machine operator intervenes. These spool files should be closed periodically so that spool files do not build up excessively on the spooling DASD devices.

STORAGE AND CPU UTILIZATION

The system operator may assign the reserved page frames option to a single virtual machine. This option, specified by the SET RESERVE command, assigns a specific amount of the storage of the real machine to the virtual machine. CP will dynamically build up a set of reserved real storage page frames for this virtual machine during its execution until the maximum number "reserved" is reached. Make sure that enough frames are reserved to equal those needed by the most active pages to minimize paging in this virtual machine. Since other virtual machines' pages are rarely allocated from this reserved set, the effect is that the most active pages of the selected virtual machine remain in real storage.

During CP system generation, the installation may specify an option called virtual=real. With this option, the virtual machine's storage is allocated directly from real storage at the time the virtual machine logs on (if it has the VIRT=REAL option in its directory). All pages except page zero are allocated to the corresponding real storage

locations. In order to control the real computing system, real page zero must be controlled by CP. Consequently, the real storage size of the computer must be large enough to accommodate the CP nucleus, the entire virtual=real virtual machine, and the remaining pageable storage requirements of CP and the other virtual machines. For more detailed V=R storage estimates, see the VM/370: Planning and System Generation Guide.

The virtual=real option improves performance in the selected virtual machine since it removes the need for CP paging operations for that machine. The virtual=real option is also necessary whenever programs containing dynamically modified channel programs that are not supported by VM/370 are to execute under control of CP.

VIRTUAL MACHINE I/O MANAGEMENT

A real disk device and a real or emulated 270x TCU can be shared among multiple virtual machines. Virtual disk device sharing is specified in the VM/370 directory entry or by a user command. If specified by the user, an appropriate password must be supplied before gaining access to the virtual device. A particular virtual machine may be assigned read-only or read/write access to a shared disk device. CP checks each virtual machine input/output operation against the parameters in the virtual machine configuration to ensure device integrity.

The virtual machine operating system is responsible for the operation of all virtual devices associated with it. These virtual devices may be defined in the VM/370 directory entry of the virtual machine, or they may be attached to (or detached from) the virtual machine's configuration, dynamically, for the duration of the terminal session. Virtual devices may be dedicated, as when mapped to a fully equivalent real device; shared, as when mapped to a minidisk or when specified as a shared virtual disk device; or spooled by CP to intermediate direct access storage.

In a real machine running under control of OS for example, input/output operations are normally initiated when a problem program requests OS, to issue a START I/O instruction to a specific device. Device error recovery is handled by the operating system. In a virtual machine, OS can perform these same functions, but the device address specified and the storage locations referenced will both be virtual. It is the responsibility of CP to translate the virtual specifications to real.

In addition, the interrupts caused by the input/output operation are reflected to the virtual machine for its interpretation and processing. If input/output 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).

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

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 interruption to the appropriate storage areas of each virtual machine.

By specifying a dedicated channel(s) for a virtual machine via the class B ATTACH CHANNEL command, the virtual machine has the channel(s) and all the devices for its own exclusive use. 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, minidisks cannot be used. An alternative that should be considered is the use of dedicated devices, since then the only translation done by CP is device address. The dedicated devices should be specified as being on separate channels, because VM/370 can handle this situation more efficiently for a virtual machine.

SPOOLING FUNCTIONS

A virtual unit record device, which is mapped directly to a real unit record device, is said to be dedicated. The real device is then controlled completely by the virtual machine's operating system.

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

Spooling operations cease if the direct access storage space assigned to spooling is exhausted, and the virtual unit record devices appear in a not ready status. The system operator may make additional spooling space available by purging existing spool files, by assigning additional direct access storage space to the spooling function, or by assigning additional real printers and/or punches to speed up the processing of spool files.

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 at the time they are created (closed). Files transferred between virtual unit record devices by the spooling routines 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.

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, 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, 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 log terminal input and output in a console spool file. All data sent to the terminal, whether it be from the virtual machine, the control program, or the virtual machine operator, can be

spooled. Spooling is particularly desirable when a virtual machine has a display device as its console or if the virtual machine is run with its console disconnected. Console spooling is usually started via the CP SPOOL command:

```
spool console start
```

An exception to this is when the VM/370 system operator logs on using a display device. In this instance, console spooling is automatically started and continues in effect even if the system operator should disconnect from the display device and reconnect using a printing device. In order to stop console spooling, the user must issue the command:

```
spool console stop
```

When your console spool file is closed, CP categorizes it as a printer spool file and normally schedules it for printing on the system printer.

Who Should Spool

Most operating systems, such as DOS/VS, VS1, and VS2, provide their own spooling subsystems. Should you use the spooling facilities offered by your own operating system (that is, by POWER, POWER/VS, HASP, or JES), or should you use CP's spooling, or use both? The three main choices are:

- Use only the operating system's spooling mechanism.
- Use only CP spooling.
- Use double spooling.

If you eliminate CP spooling for this virtual machine, you must have a sufficient number of unit record devices to dedicate some to this virtual machine (via VM/370 directory entries) while allocating the remainder of the unit record devices to CP or to other virtual machines. If this virtual machine has a sufficient volume of spooled data, this alternative has some merit. Be aware, though, that the unit record devices will probably not be able to run consistently at rated speed because this virtual machine will not always be executing. Whenever some other virtual machine is executing, spooling stops for these dedicated devices.

If you eliminate operating system spooling and use only CP spooling, you must assign virtual unit record devices to each partition that needs them. The primary advantages of letting CP do the spooling is that CP spooling is one of the highest priority functions in CP. Therefore, the unit record devices are more likely to run closer to their rated speeds. Also, all virtual machines will be able to access those devices so the scheduling of output is simpler.

However, each unit record read, punch, and print I/O operation causes CP to handle SIO simulation for the device at an additional cost of overhead. In certain situations (such as special forms handling or loading different sets of characters into the 3211 printer's print buffer), CP spooling may not offer the exact facilities desired by the user. In other cases, such as when a user wants to flush the remainder of his job stream when an earlier job step aborts, there is no convenient or easy way to do this under CP spooling. With most of the spooling subsystems that are part of DOS/VS and OS/VS, provisions are available for flushing the remaining job steps of a job stream.

At first glance, double spooling appears to be the least desirable alternative, since you need substantially more DASD space in order to let both systems spool, and you would suspect that double spooling would be ineffective from a performance standpoint. This, however, is usually not true, since the virtual machine's spooling subsystem normally creates its unit record I/O in large blocks, thereby substantially reducing the number and frequency of SIO instructions (and I/O interrupts) that VM/370 has to handle.

Spooling Recommendations

If DASD space is not a limiting factor, use double spooling. If possible, generate a stripped-down version of the virtual machine's spooling subsystem, eliminating those functions not used by that virtual machine. Make the I/O buffer sizes as large as possible to cut down on SIO instructions.

If you have only enough DASD spooling space for one spooling subsystem and there is only one virtual machine that generates significant amounts of spooled output, then let that virtual machine do the spooling. If, however, you have many virtual machines that spool data and must use a common pool of unit record devices, then you should probably let CP do the spooling.

Closing Spool Files

Output spool files are not scheduled for the real printer or punch devices until one of the following occurs:

- The user logs off or is forced off.
- The user loads an operating system via the CP IPL command.
- The user (either manually or through an EXEC procedure) issues the CP command CLOSE to close the spool file.
- If using VS1 with the VM/VS Handshaking feature, VS1 uses the Diagnose Interface to issue the CP CLOSE command after each job completes.
- The installation's system programmer has modified the operating system(s) by adding DIAGNOSE instructions to communicate to CP to close the spool files.

MINIDISKS

The external storage requirements of multiple virtual machines executing concurrently would probably be excessive if each virtual machine were assigned one real direct access storage device for each virtual DASD device specified in its configuration.

Therefore, if you do not require the full capacity of a real DASD device, you can be assigned one or more minidisks instead. A minidisk is a logical subdivision of a physical disk pack with its own virtual device address, virtual cylinders (starting with 0, 1, 2, and so on), and a VTOC (volume table of contents or disk label identifier). Each of

your minidisks is preallocated the number of contiguous full cylinders that were specified in the VM/370 MDISK directory record, and that space is considered to be a complete virtual disk device.

Minidisks are controlled and managed by CP. If a virtual machine attempts to use DASD space beyond the boundaries defined for its minidisks, CP presents a command reject (seek check) to the virtual machine. If a system is to be run on both a virtual and a real machine, VM/370 recommends that minidisks not be used for that system unless the minidisk starts at real cylinder zero. For a detailed list of minidisk restrictions, see "Appendix B. CP Restrictions."

The following functional characteristics of minidisks are described below:

- Definition
- Space allocation
- Track characteristics
- Alternate tracks
- Labels

DEFINING MINIDISKS

Permanent minidisks are defined in the VM/370 directory entry for a virtual machine. A minidisk defined in the directory via an MDISK statement is a permanent part of the virtual machine configuration and the data on the minidisk is available to the user from session to session. Examples of MDISK statements can be found in "Section 7. Configuring Your Virtual Machine."

If any virtual machine has a temporary requirement for direct access space, this can be filled from a pool of T-disk space.

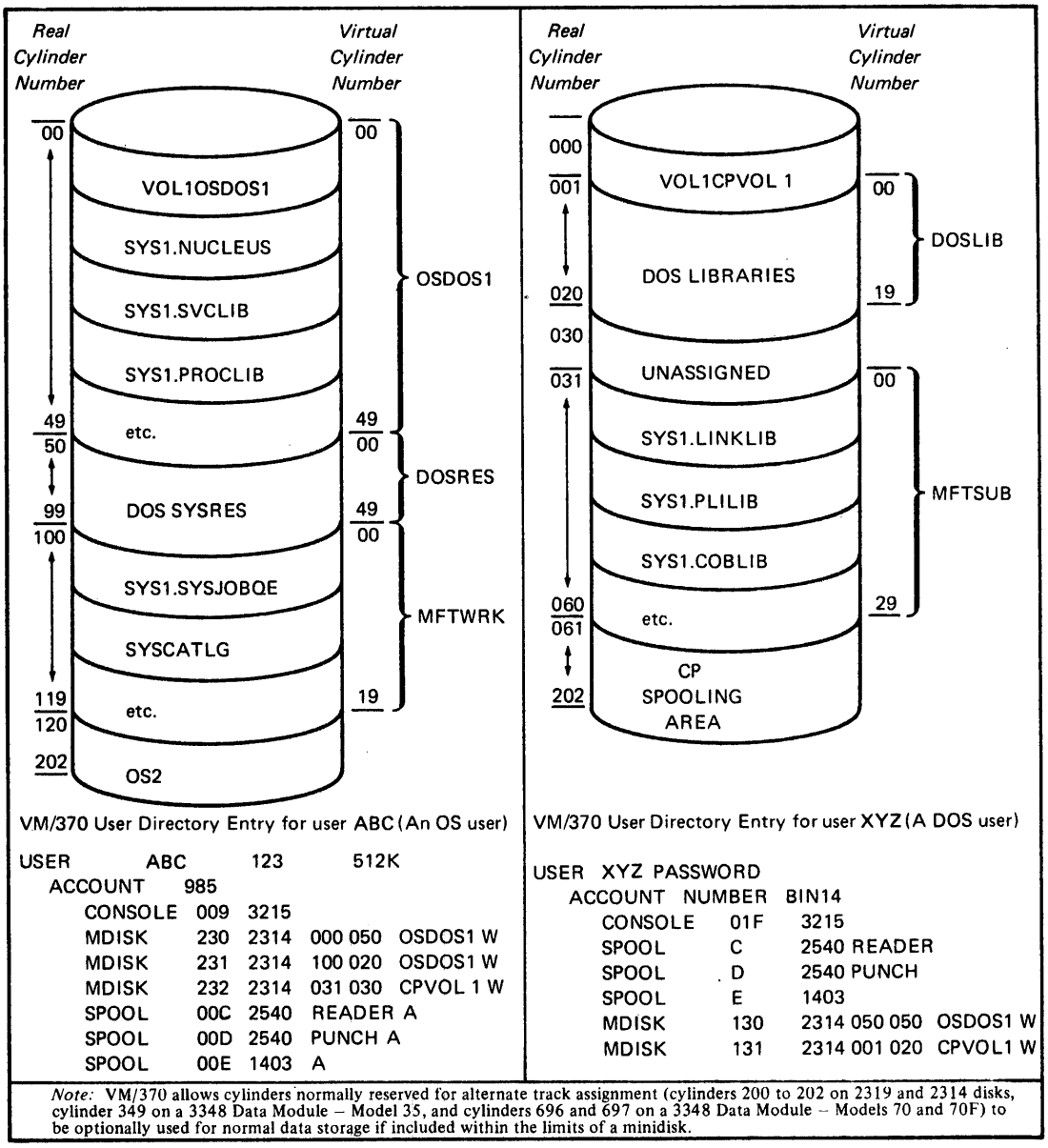
Note: The installation's system programmer specifies the size of the T-disk pool when he allocates disk space with the standalone CP Format/Allocate program.

Minidisks created from the T-disk area must be initialized by the user. If DOS, OS, or VSAM files are to be stored, the IBCDASDI program must be used to initialize the minidisk and set up the Volume Table of Contents (VTOC). If the disk is to be used for CMS files, the CMS FORMAT command must be issued. Temporary minidisks are available to the virtual machine for the duration of one terminal session. When the virtual machine logs off, is forced off, or issues a CP command to release the temporary minidisk, the area is returned to CP.

It is up to you to allocate minidisks on VM/370 disks in a manner that minimizes arm contention and physical overlap.

Note: The VM/370 Directory function does not check for or flag overlapped or duplicate minidisk extents or provide a record of unused cylinders or totals of all used and unused DASD space.

Figure 1 illustrates the use and definition of minidisks. The disk labeled OSDOS1 contains several minidisks, some formatted to OS requirements and others to DOS requirements. OSDOS1 is a 2314 volume. The directory entry for userid ABC (an OS user) describes the virtual device 230 as a 50-cylinder area, and the virtual device 231 as a 20-cylinder area on real volume OSDOS1. The directory entry for userid XYZ (a DOS user) describes the virtual device 130 as a 50-cylinder disk area on a real volume OSDOS1.



Note: VM/370 allows cylinders normally reserved for alternate track assignment (cylinders 200 to 202 on 2319 and 2314 disks, cylinder 349 on a 3348 Data Module – Model 35, and cylinders 696 and 697 on a 3348 Data Module – Models 70 and 70F) to be optionally used for normal data storage if included within the limits of a minidisk.

Figure 1. Use and Definition of Minidisks

The real volume CPVOL1 also contains disk areas assigned to userid ABC (virtual device address 232) and userid XYZ (virtual device address 131).

Note: On a 3330, 3340, or 3350, an OS or OS/VS minidisk must start at real cylinder 0 unless the VTOC is limited to one track. See the list of CP restrictions in "Appendix B. CP Restrictions" for more information and explanation of 3330/3340/3350 restrictions.

MINIDISK SPACE ALLOCATION

OS bases all of its space allocation parameters on the Volume Table Of Contents (VTOC) label written on each disk; it determines the amount of space available on that volume from the format 5 (space accounting) data set control block. Thus, for OS to support minidisks, a VTOC must be written whose space accounting data set control block (DSCB) is equal to the desired size of the minidisk. The remainder of the disk space on the real disk appears to OS to be permanently dedicated, and not assignable by the OS space accounting routines.

A DASD volume containing multiple minidisks will contain some tracks in which the cylinder address in the count fields of R0 and R1 will not agree with each other. If an attempt is made to read this volume by IEHDASDR, you may get messages IEH813I or IEH869I. To prevent this, initialize the disk with the FORMAT function of IEHDASDR before using it. This will rewrite R0 and R1 on each track so that the count fields agree with each other.

DOS space allocation is specified in the EXTENT job control card. It is your responsibility to see that the EXTENT cards refer to valid minidisk cylinders. On a 2314, 2319, or 3340 volume, the last cylinder of any minidisk initialized by IBCDASDI is always reserved for use as an alternate track cylinder. Therefore, a DOS minidisk on a 2314, 2319, or 3340 must have a minimum of two cylinders. For example, if you are specifying a ten-cylinder minidisk, the EXTENT card must refer to cylinders 0 through 8 only. This leaves the last cylinder for alternate track assignment. However, on a 3330, 3333, or 3350 minidisk, IBCDASDI does not reserve alternate tracks. Therefore, a ten-cylinder minidisk must be defined in the EXTENT card as cylinders 0 through 9.

A minidisk always begins at virtual cylinder zero. Its minimum size is one cylinder unless it is located on a 2314, 2319, or 3340 disk and is formatted by the IBCDASDI service program; in which case, the minimum number of cylinders is two. The second cylinder is used as the alternate track cylinder. Except for the 3350, which can be used in 3330-1 or 3330-11 compatibility mode or in native mode, a minidisk must exist on its real counterpart, that is, a virtual 3340 minidisk must reside on a real 3340.

VM/370 controls the boundaries of minidisks. If an attempt is made to refer to a DASD address outside the boundaries specified in the MDISK directory statements, CP presents a command reject (seek check) I/O error to the virtual machine.

Note: If the cylinder addresses in the MDISK statements inadvertently overlap each other, the integrity of data in the overlapped cylinders may be compromised with no error indicated.

TRACK CHARACTERISTICS

Like real disks, minidisks must also be formatted for use by the appropriate service program. A minidisk is initialized for use by executing one of the following service programs in a virtual machine:

- For all CMS disks the CMS FORMAT command is used to format specified tracks into 800-byte blocks or physical records. FORMAT must not be used for VSAM data sets used in CMS.
- For CP disks, the standalone CP Format/Allocate program must be used to format specified tracks into 4096-byte blocks.
- For OS, DOS, and VSAM minidisks, the IBCDASDI service program is used to write read-only track descriptor records for each track, and clear the remaining portion of each track to binary zeros. It also writes a Format 5 DSCB whose contents are the minidisk size. If full disks are being initialized, any initialize-disk program that supports the operating system's use of the DASD device type may be used.

Minidisks defined in the VM/370 directory need be initialized only once; temporary minidisks must be initialized each time they are used.

ALTERNATE TRACKS

3330/3350 Disks

Alternate tracks flagged at the factory are automatically handled on the 3330 or 3350 by the control unit. Alternate tracks on a 3330 or 3350 cannot be assigned in the field. Minidisks on the 3330 Model 1 or 2 should be specified on cylinder 0 through cylinder 403 only. The remaining cylinders (404 to 411) are automatically used by the 3830 Control Unit for alternate tracks. Minidisks on the 3330 Model 11 can be specified on cylinder 0 through cylinder 807. Minidisks on the 3350 should be specified on cylinder 0 through cylinder 554 only. The remaining cylinders (555 to 559) are automatically used by the 3830 control unit for alternate tracks.

2314, 2319, and 3340 Disks

On 2314, 2319, and 3340 devices, CP and CMS (except CMS/VSAM) do not recognize or support alternate track techniques for their own use. DOS, OS, and VSAM minidisks, however, do recognize and support alternate tracks on these types of DASD devices. The IBCDASDI service program automatically assigns the last cylinder in any minidisk on these disks as an alternate track cylinder. When you initialize 2314/2319 devices, you can assign all 203 cylinders for virtual machine and system use. And for a 3340, you can initialize up to 696 cylinders, depending on the model. For a 3348 Data Module, Model 35, you can assign the entire disk, 349 cylinders. For a 3348 Data Module, Model 70, you can assign up to 682 cylinders. These numbers of cylinders apply only to CMS, not to DOS, OS, or CMS/VSAM.

If a track assigned to a virtual machine minidisk area subsequently becomes defective, you can:

- Run the standalone CP Format/Allocate program if the minidisk is used by CP, and flag the whole cylinder containing the defective track as permanently assigned (PERM). This prevents CP from ever allocating that cylinder for CP paging, spooling, or temporary files. You must remember not to include this cylinder when you allocate disk space for any virtual machine's minidisk in the VM/370 directory.

- If the minidisk is used by either DOS, OS, or CMS/VSAM, reformat the minidisk (including the defective track) with the IBCDASDI service program. An alternate track is assigned at the end of the minidisk.

- Set up the entire volume containing the defective track as an OS, DOS, or VSAM volume and format it with either IBCDASDI, INTDK, or IEHDASDR. Alternate tracks are assigned in the standard manner by these programs.

INVOKING THE IBCDASDI PROGRAM

The format and description of the IBCDASDI control statements can be found in the VM/370: Operator's Guide. The IBCDASDI program is invoked for minidisks by specifying the operand

```
...,CYLNO=nnn
```

on the DADEF control statement. This operand specifies the number of cylinders of a minidisk that are to be formatted. If this value is larger than the size of the minidisk, as defined in the VM/370 directory, the IBCDASDI program will abort when it attempts to process the undefined area. If the value is less than the minidisk size defined in the VM/370 directory, only the number of cylinders specified in the DADEF statement are formatted. The format 5 DSCB for that minidisk will reflect the smaller extent and the unformatted portion of the minidisk will not be available to the operating system.

The IBCDASDI program, which is distributed as a CMS file with a filename of IPL and a filetype of IBCDASDI, should be spooled to your own virtual card reader. Control statements for the program can follow the last card or card image for the program, or can be entered via a separate input device.

To execute the IBCDASDI program:

1. Spool a copy of the IBCDASDI object module to your virtual card reader, or mount and attach the tape containing the object program.

2. Load the object program from the virtual reader or tape by issuing the CP IPL command for the appropriate virtual device address. When the program is loaded, an enabled wait state is entered with the address field of the PSW containing the hexadecimal value FFFF.

3. When the program is loaded and waiting for input, signal attention from the virtual console device. The message:

   ```
   DEFINE INPUT DEVICE
   ```

   is sent to your virtual console. Enter the following reply from the virtual console:

   ```
   input=type,cuu
   ```

where:

type — is the device type of the virtual device containing the control statements. Valid device types are 2400, 2540, 3410, 3420, and 3505.

cuu — is the device address of the virtual device containing the control statements.

Control statements are printed on the message output device. At the end of job, the END OF JOB message is printed on the message output device and the program enters the wait state.

DISK LABELS

All disks to be handled by CP (as an entity or as a combination of logical disks) must have a label on real cylinder 0, track 0, record 3. This label identifies the physical volume to VM/370 and must be in the form

```
VOL1xxxxxx

 -- or --

CMS=xxxxxx
```

where xxxxxx is a six-character volume label.

In addition, all virtual machine minidisks should have a label at virtual cylinder 0, track 0, record 3. Labels created by IBCDASDI, IEHDASDR, or INTDK are in the form

```
VOL1xxxxxx
```

where xxxxxx is a six-character volume label.

A physical volume that holds only virtual machine minidisks can have the first of those minidisks starting at real cylinder 0. CP recognizes the physical volume if the first minidisk has a valid label.

In Figure 1, the volume indicated as OSDOS1 has its real cylinder 0 allocated to a minidisk which is formatted for use by OS. The volume serial number of that minidisk must be OSDOS1, the label that is associated with the real volume. Since the minidisk label identifies the physical volume, changing it affects the directory entries of all users who have minidisks on that volume.

You should not assign real cylinder 0 to a user as a data area, because that user (if he has read/write access to the disk) can rewrite the label on the minidisk.

Additionally, you must not assign user minidisks to begin on real cylinder 0 of any physical volumes which are to contain CP-controlled areas (for paging, spooling, and so on). On these volumes, cylinder 0 track 0 record 4 contains control information required by CP. The VTOC labels written are compatible with OS, but indicate to OS that there is no space on that DASD storage device. The initialization programs used to format OS, DOS, and CMS/VSAM minidisks write over and destroy this necessary control information if the space is assigned to a user minidisk, and this causes CP system failures.

SHARING MINIDISKS

A minidisk can be shared by multiple virtual machines. One virtual machine is designated the owner of the minidisk (it has an MDISK control statement in its VM/370 directory entry describing the minidisk) and other virtual machines can link to the minidisk.

For example, assume that a virtual machine called USERA owns a minidisk at address 194. The VM/370 directory entry for USERA contains the following statement:

```
MDISK 194 2314 050 010 DOSSYS R READPASS
```

USERA's virtual disk is on the volume labeled DOSSYS and occupies real cylinders 050-059.

Any other virtual machine that issues the CP LINK command with the proper password, or has the following LINK control statement in its VM/370 directory entry, can read the 194 minidisk belonging to USERA.

```
LINK USERA 194 cuu R
```

The cuu is the virtual device address at which the 194 minidisk belonging to USERA is linked to another virtual machine. For example, if you define another virtual machine, USERB, with the following statement in its VM/370 directory entry:

```
LINK USERA 194 195 R
```

USERB can read data from USERA's 194 virtual disk whenever it issues a read for data on its own 195 virtual disk.

You can link to any minidisk that is defined in the VM/370 directory if that minidisk has a read and/or write password specified in the MDISK control statement and if the type of link you desire is allowed. Three types of sharing may exist and, correspondingly, three passwords may be specified.

The three types of sharing are:

- Read-only (R) indicates that all virtual machines sharing the disk are using it in read status.
- Read/write (W) indicates that one virtual machine may have read/write access and multiple virtual machines may have concurrent read-only access.
- Multi-write (MW) indicates that multiple virtual machines may issue writes concurrently to the disk. Generally, this mode of access requires that the virtual machines include code to control this, such as the shared DASD support of OS.

Note: See the description of the CP LINK command in the VM/370: CMS User's Guide for more information about linking to minidisks.

THE TWO-CHANNEL SWITCH

VM/370 has no multiple path support and does not take advantage of the two-channel switch. However, a two-channel switch can be used between the System/370 running a virtual machine under VM/370 and another CPU.

If any I/O devices controlled by VM/370 for its exclusive use are attached to control units with two-channel switches, the CPU or virtual machine controlling the other channel interface must vary the CP-owned devices offline.

See "Channel Switching" in "Section 9. Special Considerations" for additional information about using the two-channel switch.

UNSUPPORTED DEVICES

Any input/output device not supported by VM/370 that can be attached to the IBM System/370 can be used by a virtual machine under VM/370, provided that:

- No timing dependencies exist in the device or the program
- No dynamically modified channel programs except OS Indexed Sequential Access Method (ISAM) or OS/VS Telecommunications Access Method (TCAM) Level 5 are used
- No special functions need to be provided by VM/370
- None of the other restrictions outlined in "Appendix B. CP Restrictions" are violated.
- The device is generated into the VM/370 nucleus via the RDEVICE macro with the appropriate CLASS operand.

Input/output devices that are part of a virtual machine's configuration require real device equivalents, except for:

- Unit record devices, which CP can simulate using spooling techniques.
- Virtual 2311 Disk Storage Drives, which CP can map onto 2314 or 2319 disks. Up to two full 2311 units can be mapped onto a 2314 or 2319 disk in this manner.
- When 3350 disks are used in 3330-1 or 3330-11 compatibility mode.

DEVICES THAT CANNOT BE USED UNDER VM/370

Certain types of devices are not supported by VM/370, nor can they be dedicated to virtual machines even if these virtual machine operating systems have support for those devices. Examples of such devices are:

- The 3850 Mass Storage System
- The 3800 Printer

Section 3. General Operating Procedures

Section 3 covers the operational functions that are common to operating systems being run under VM/370. Any operating system, running in a virtual machine, must have access to a real CPU for program execution and to real input and output hardware for data movement and program access. Over and above these two basic needs, VM/370 provides the user with CP commands that will simulate the functions of the real console buttons, keys, switches, and lights; initiate program tracing; facilitate the alternate use of multiple operating systems; measure system and virtual machine performance; and, set performance options.

Section 3 contains the following topics:

The CP Command Language

Controlling a Terminal Session

- Logging on to VM/370
- Loading an Operating System
- Connecting to a Multiple Access Virtual Machine
- Controlling Terminal Functions
- Temporarily Disconnecting the Terminal
- Placing the Terminal in a Dormant State
- Terminating a Terminal Session

Controlling Input and Output Functions

- Virtual Disks
- Virtual Unit Record Devices
- Dedicated Devices

Controlling the Virtual Machine

- Simulating Interrupts
- Entering CP Commands
- Reconfiguring the Virtual Machine

Testing and Debugging of Programs

- Stopping Execution of Your Virtual Machine
- Displaying Virtual Storage
- Altering Virtual Storage
- Tracing Virtual Machine Activity

Using Alternating Operating Systems

- Transferring Output
- Configurations

Performance Measurements and Options

- VM/370 System Performance
- Virtual Machine Performance
- Performance Options

Note: Throughout the rest of this publication, the text will be interspersed with examples of VM/370 commands and system responses. All user input will be shown in lowercase; all system responses will be shown in uppercase.

The CP commands represent a set of interactive console functions that are used (1) by the operator to control the VM/370 system, and (2) by a user to control a virtual machine.

Each VM/370 user is assigned one or more privilege classes as part of the directory entry for his virtual machine. Each privilege class is indicative of a specific function of a virtual machine and entitles the user to a specific subset of the CP command language.

Figure 2. identifies each CP privilege class along with the associated type of user and function performed. Also listed are the specific publications in which each class of CP commands is described in detail.

Figure 3. contains an alphabetical list of the CP commands, the privilege classes which may execute the command, and a brief statement about the use of each command.

Class	User and Function
A[1]	Primary System Operator: The class A user controls the VM/370 system. Class A is assigned to the user at the VM/370 system console during IPL. The primary system operator is responsible for the availability of the VM/370 system and its communication lines and resources. In addition, the class A user controls system accounting, broadcast messages, virtual machine performance options and other command operands that affect the overall performance of VM/370. Note: The class A system operator who is automatically logged on during CP initialization is designated as the primary system operator.
B[1]	System Resource Operator: The class B user controls all the real resources of the VM/370 system, except those controlled by the primary system operator and spooling operator.
C[1,2]	System Programmer: The class C user updates certain functions of the VM/370 system.
D[1]	Spooling Operator: The class D user controls spool data files and specific functions of the system's unit record equipment.
E[1,2]	System Analyst: The class E user examines and saves certain data in the VM/370 storage area.
F[1,3]	Service Representative: The class F user obtains and examines in detail, certain data about input and output devices connected to the VM/370 system.
G[4]	General User: The class G user controls functions associated with the execution of his virtual machine.
Any[1,4]	The Any classification is given to certain CP commands that are available to any user. These are primarily for the purpose of gaining and relinquishing access to the VM/370 system.
H	Reserved for IBM use.

[1]Described in the VM/370: Operator's Guide.
[2]Described in the VM/370: System Programmer's Guide.
[3]Described in the VM/370: OLTSEP and Error Recording Guide.
[4]Described in the VM/370: CP Command Reference for General Users.

Figure 2. CP Privilege Class Descriptions

Command	Privilege Class	Usage
*	any	Annotate the console sheet.
#CP	any	Execute a CP command while remaining in the virtual machine environment.
ACNT	A	Create accounting records for logged on users and reset accounting data.
ADSTOP	G	Halt execution at a specific virtual machine instruction address.
ATTACH	B B B	Attach a real device to a virtual machine. Attach a DASD device for CP control. Dedicate all devices on a particular channel to a virtual machine.
ATTN	G	Make an attention interruption pending for the virtual machine console.
AUTOLOG	A,B	Automatically log on a virtual machine and have it operate in disconnect mode.
BACKSPAC	D	Restart or reposition the output of a unit record spooling device.
BEGIN	G	Continue or resume execution of the virtual machine at either a specific storage location or at the address in the current PSW.
CHANGE	D,G	Alter one or more attributes of a closed spool file.
CLOSE	G	Terminate spooling operations on a virtual card reader, punch, printer, or console.
COUPLE	G	Connect virtual channel-to-channel adapters.
CP	any	Execute a CP command while remaining in the CMS virtual machine environment.
DCP	C, E	Display real storage at terminal.
DEFINE	G	Reconfigure your virtual machine.

Figure 3. CP Command Summary (Part 1 of 4)

Command	Privilege Class	Usage
DETACH	B	Disconnect a real device from a virtual machine.
	B	Detach a DASD device from CP.
	B	Detach a channel from a specific user.
	G	Detach a virtual device from a virtual machine.
	G	Detach a channel from your virtual machine.
DIAL	any	Connect a terminal or display device to a virtual multiple-access system.
DISABLE	A,B	Disable 2701/2702/2703, 3704/3705 in EP mode, and 3270 communication lines.
DISCONN	any	Disconnect your terminal from your virtual machine.
DISPLAY	G	Display virtual storage on your terminal.
DMCP	C,E	Dump the specified real storage location on your virtual printer.
DRAIN	D	Halt operations of specified spool devices upon completion of current operation.
DUMP	G	Print the following on the virtual printer: virtual PSW, general registers, floating-point registers, storage keys, and contents of specified virtual storage locations.
ECHO	G	Test terminal hardware by redisplaying data entered at the terminal.
ENABLE	A,B	Enable communication lines.
EXTERNAL	G	Simulate an external interruption for a virtual machine and return control to that machine.
FLUSH	D	Cancel the current file being printed or punched on a specific real unit record device.
FORCE	A	Cause logoff of a specific user.
FREE	D	Remove spool HOLD status.
HALT	A	Terminate the active channel program on specified real device.
HOLD	D	Defer real spooled output of a particular user.
INDICATE	E,G	Indicate resource utilization and contention.
IPL	G	Simulate IPL for a virtual machine.
LINK	G	Provide access to a specific DASD device by a virtual machine.
LOADBUF	D	Load real UCS/UCSB or FCB printer buffers.

Figure 3. CP Command Summary (Part 2 of 4)

Command	Privilege Class	Usage
LOADVFCB	G	Load virtual forms control buffer for a virtual 3211 printer.
LOCATE	C,E	Find CP control blocks.
LOCK	A	Bring virtual pages into real storage and lock them; thus excluding them from future paging.
LOGOFF	any	Disable access to CP.
LOGON	any	Provide access to CP.
MESSAGE	A,B,any	Transmit messages to other users.
MONITOR	A,E	Trace events of the real machine and record system performance data.
NETWORK	A,B,F	Load, dump, trace and control the operation of the 3704/3705 control program. Control the operation of remote display units.
NOTREADY	G	Simulate "not ready" for a device to a virtual machine.
ORDER	D,G	Rearrange closed spool files in a specific order.
PURGE	D,G	Remove closed spool file from system.
QUERY	A,B,C,D,E,F,G	Request information about machine configuration and system status.
READY	G	Simulate device end interruption for a virtual device.
REPEAT	D	Repeat (a specified number of times) printing or punching of a specific real spool output file.
REQUEST	G	Make an attention interruption pending for the virtual machine console.
RESET	G	Clear and reset all pending interruptions for a specified virtual device and reset all error conditions.
REWIND	G	Rewind (to load point) a tape and ready a tape unit.
SAVESYS	E	Save virtual machine storage contents, registers, and PSW.
SET	A,B,F,G	Operator--establish system parameters. User--control various functions within the virtual machine.

Figure 3. CP Command Summary (Part 3 of 4)

Command	Privilege Class	Usage
SHUTDOWN	A	Terminate all VM/370 functions and checkpoint CP system for warm start.
SLEEP	any	Place virtual machine in dormant state.
SPACE	D	Force single spacing on printer.
SPOOL	G	Alter spooling control options; direct a file to another virtual machine or to a remote location via the RSCS virtual machine.
START	D	Start spooling device after draining or changing output classes.
STCP	C	Change the contents of real storage.
STORE	G	Alter specified virtual storage locations and registers.
SYSTEM	G	Simulate RESET, CLEAR STORAGE and RESTART buttons on a real system console.
TAG	G	Specify variable information to be associated with a spool file or output unit record device. Interrogate the current TAG text setting of a given spool file or output unit record device.
TERMINAL	G	Define or redefine the input and attention handling characteristics of your virtual console.
TRACE	G	Trace specified virtual machine activity at your terminal, spooled printer, or both.
TRANSFER	D,G	Transfer input files to or reclaim input files from a specified user's virtual card reader.
UNLOCK	A	Unlock previously locked page frames.
VARY	B	Mark a device unavailable or available.
WARNING	A,B	Transmit a high priority message to a specified user or to all users.

Figure 3. CP Command Summary (Part 4 of 4)

Before you can gain access to VM/370 you must have been assigned a virtual machine user identification (userid) and a password. The userid identifies a particular virtual machine configuration that has been entered into the VM/370 directory. The password is a 1- to 8-character symbol that functions as a protective device to prevent unauthorized use of your virtual machine. A representative directory entry for a DOS/VS user would be:

```
(1)   USER DOSVUSER 123456 512K 2M
(2)   ACCOUNT 87321 BIN14
(3)   OPTION ECMODE
(4)   CONSOLE 01F 3215
(5)   SPOOL 00C 2540 R
(5)   SPOOL 00D 2540 P
(5)   SPOOL 00E 1403
(6)   MDISK 130 2314 050 050 OSDOS1 R
(6)   MDISK 151 3330 001 020 CPVOL1 W
(6)   LINK CMSSYS 190 190 RR
```

where:

(1) The userid of this virtual machine is DOSVUSER and the password is 123456. The virtual machine's storage size is defined as 512K; however, this can be redefined up to 2 megabytes if, during a operating session, the need for additional storage arises (see the section on "Reconfiguring the Virtual Machine).

(2) VM/370 punches out accounting data for CPU time and I/O device utilization. This data is charged to a virtual machine by account number, which in this case is 87321. BIN14 is a distribution code that is printed or punched on spooled printer and punch output and usually denotes a location where spooled output of a terminal session can be picked up by the user.

(3) Option ECMODE must be specified for DOS/VS.

(4) The virtual address of the virtual machine console is 01F and your operating system will correspond with the terminal as if it were a 3215.

(5) The virtual unit record devices (reader, punch, and printer) are to be spooled and their addresses, as far as the virtual machine is concerned, are 00C, 00D, and 00E, respectively.

(6) In this configuration, a 50 cylinder read-only minidisk with a virtual address of 130 is located on cylinders 50 through 99 of a real 2314 volume labeled OSDOS1. Similarly, a 20 cylinder read/write minidisk with a virtual address of 151 is located on cylinders 1 through 20 of a real 3330 volume labeled CPVOL1. The last entry provides a link to a sharable CMS system volume in the event the user wishes to use CMS functions such as Editor or the EXEC Processor.

LOGGING ON TO VM/370

To initiate a terminal session you must identify yourself to VM/370. This requires that a communication path exist between your terminal and the real machine running VM/370. This communication path can be (1) integrated as part of the CPU for local terminals used as system consoles; (2) a leased or switched line through a transmission control unit for typewriter terminals; (3) through a 3272 control unit for local display terminals; or (4) through a leased line and a 3271 control unit for remote display terminals. When a line connection to VM/370 has been established, turning the power on at your terminal will result in some form of a ready message.

On a typewriter terminal, a "vm/370 online" message will type out.

On a display terminal, the "vm/370 online" message is displayed on the screen, the system available indicator light is activated, and the screen status indicator on the last line of the screen displays "RUNNING".

Pressing the Attention key (or equivalent) notifies VM/370 that you are about to communicate with it. You now identify yourself to VM/370 by entering the LOGON command with your userid:

```
logon smith
```

When VM/370 has accepted your userid as being in its directory of users, it will request your password:

```
ENTER PASSWORD:
```

You then enter your password which in most installations will not be displayed or printed for security reasons. When your password has been accepted and verified, VM/370 issues the LOGON message:

```
LOGON AT hh:mm:ss yyy weekday mm/dd/yy
```

where:

hh:mm:ss	is the time of day
yyy	is the time zone
weekday	is the day of the week
mm/dd/yy	is the date

You can also enter your password on the LOGON command line following your userid; however, this nullifies the system's password protection feature.

If you attempt to log on and receive a message saying that your userid is already logged on, someone else is using your virtual machine. You can send a message asking him to release the virtual machine, for example:

```
msg smith this is smith - pls log off
```

LOADING AN OPERATING SYSTEM

At the completion of the LOGON procedure you have a virtual machine of a predetermined configuration at your disposal. As with a real machine, its use is limited without an operating system. An operating system can

be loaded via the IPL command or automatically at logon time if an IPL control statement is included in your virtual machine's directory entry.

When a virtual machine runs the same operating system with very few exceptions, it may be expedient to use automatic loading. If the VM/370 directory entries for your userid include an IPL control statement with the name or the virtual address of a specific system to be loaded, that system will be automatically loaded when you log on to VM/370 unless you issue the LOGON command as:

```
logon userid noipl
```

in which case, the automatic loading does not take place and you can IPL any supported system that is available.

A more flexible approach is loading an operating system via the IPL command. Again you have a choice. You can IPL by device address or by system name. If you IPL by the virtual address of the device containing the operating system you can also clear virtual storage to binary zeroes before loading the system:

```
ipl 190 clear
```

This facility can be useful if your operating system does not automatically clear storage when it is loaded.

You can stop the IPL procedure at a point just before the initial PSW is loaded by issuing the IPL command with the STOP operand:

```
ipl 190 clear stop
```

At this point, you can issue CP commands to display or alter data in your nucleus. To restart the virtual machine, issue the command:

```
begin
```

You can load your operating system by name providing that the name refers to a system that has been previously saved by your installation's system programmer. As an example:

```
ipl dosvs
```

If you do load by name, the options to clear storage or to stop before loading the initial PSW are not available.

Whether you IPL by device address or by name you can pass up to 64 bytes of data (including embedded and trailing blanks) to your operating system via the IPL command line. The data is entered following the keyword PARM:

```
ipl dosvs parm this data will be passed
```

VM/370 will load the general registers, starting with register zero with the contents of the command line beginning with the first non-blank character after the keyword PARM and ending with the last character or blank entered. Embedded and trailing blanks will be passed. For the above example, the general registers will contain the following:

```
GPR  0   THIS    DAT   A WI   LL B
GPR  4   E PA   SSED   xxxx   xxxx
GPR  8   xxxx   xxxx   xxxx   xxxx
GPR 12   xxxx   xxxx   xxxx   xxxx
```

where xx... denotes no change to previous contents.

If you had entered three blank characters following the word "passed" in the command line, the registers would contain:

```
GPR  0   THIS    DAT   A WI   LL B
GPR  4   E PA   SSED      x   xxxx
GPR  8   xxxx   xxxx   xxxx   xxxx
GPR 12   xxxx   xxxx   xxxx   xxxx
```

CONNECTING TO A MULTIPLE ACCESS VIRTUAL MACHINE

If you do not have a userid in the VM/370 directory you may still be able to gain access to the system, indirectly. If a multiple-access system such as TSO has been previously logged on, you can use the DIAL command to logically connect a switched line, leased line or a local terminal to the multiple-access system. If you issue the command:

```
dial tsosys
```

and a connection can be made, you will receive a confirmation message such as:

```
DIALED TO TSOSYS 0B6
```

where 0B6 is the address of the line that you have been connected to. You can now proceed as if you had logged on directly to a TSO system. Your terminal will then be entirely under the control of the TSO system. If no line connections are available or if your terminal is not supported by both VM/370 and the TSO system, the DIAL command is not accepted.

CONTROLLING TERMINAL FUNCTIONS

Your terminal is the console of your virtual machine. With it you will enter commands that will simulate the real system console and control panel. VM/370 also provides a number of CP commands that you can use to modify the terminal functions to suit your particular environment.

TERMINAL INPUT AND OUTPUT

If your terminal is a display device (such as a 3270) and you wish to retain a hard copy record of your terminal session, you can spool the console input and output to the real printer:

```
spool console start
```

To stop console spooling and close the file for printing, issue:

```
spool console stop close
```

If you forget to stop console spooling or to close the console spool file, CP will automatically do this for you when you log off the system. Before the console spool file is closed you can change any of the spool file characteristics such as class, hold status, and number of copies by using the SPOOL CONSOLE command with appropriate operands; after the file is closed, you must use the CHANGE PRINTER command because the file has now become a printer spool file.

If you have a slow speed typewriter type terminal, you may want to eliminate the display of certain messages. Use the SET command as follows:

set msg [on|off] controls messages sent via the MSG command.

set imsg [on|off] controls informational messages from the system.

set emsg [on|off|text|code]
controls error messages. In addition to controlling the entire message, you can shorten the message to display only the error code or only the message text.

Note: If you are spooling your console, and issue the command:

set emsg off

the OFF option will be ignored and the full error message will appear on your console output. The other three options result in identical output whether or not the console is spooled.

COMMAND ENTRY

VM/370 provides you with four logical line edit characters to assist you in entering commands at your terminal. These are the character delete, line delete, line end, and escape characters and their functions are detailed in the VM/370: Terminal User's Guide.

If you wish to stop all the line editing functions you can issue:

```
set linedit off
```

If one or more of the default line edit characters is a character normally used by you in your console communication with your operating system, you can use the TERMINAL command to assign another character to the particular line edit function the new character, however, must be a special character. As an example, the command:

```
terminal linend :
```

replaces the normal default line end character (#) with a colon (:).

You can stop just one of the line edit functions, for example the line delete function, by specifying:

```
terminal linedel off
```

If you are in an APL operating environment and your terminal has the required APL hardware feature, you can use the TERMINAL command to control the selection of character translation tables.

```
terminal apl on
```

will cause CP to use the APL translation tables when communicating with the terminal.

```
terminal apl off
```

will cause CP to use the normal EBCDIC or correspondence code translation tables.

There are two basic command environments: the CP command environment and the virtual machine (VM) command environment. You are in the CP command environment when you log on and issue CP commands. You are in the VM command environment when you load your virtual machine with some operating system.

Using the Attention key (or its equivalent) on your terminal you can switch from one environment to the other at the same time that you request an interruption. How CP interprets attention interrupts depends on the mode setting of your terminal. In VM mode, a single attention interrupt will pass an interrupt pending condition to the virtual operating system. Pressing the Attention key twice quickly will place your virtual machine in the CP command environment. In CP mode, pressing the Attention key one or more times will place your virtual machine in the CP environment. To return to the virtual machine environment, issue the command:

```
begin
```

VM mode is the default status for all users except the primary VM/370 system operator; his is CP mode. To change your terminal mode, issue:

```
terminal mode cp
    --or--
terminal mode vm
```

TEMPORARILY DISCONNECTING THE TERMINAL

If your virtual machine is running in a batch environment, with no terminal interaction required, VM/370 allows you to disconnect the terminal while the virtual machine continues operation. This is accomplished via the DISCONN command. By issuing the command:

```
  disconn
  - or -
disconn hold
```

all subsequent output to the virtual console is ignored; and, unless the console had been previously spooled via the SPOOL CONSOLE START command, any further output is lost.

The HOLD operand of the DISCONN command prevents the disabling of the communication line when the connection is a dial-up or switched line. If HOLD is specified for a dialup line or if you are disconnecting on a leased or non-switched line, the "VM/370 online" message appears and you can then use the terminal to log on to another virtual machine. The virtual machine, running in disconnect mode, will remain so until it is reconnected via the normal logon procedure. When you do log back on to the virtual machine, it will be stopped and the terminal will be in CP mode. Issue the command:

```
begin
```

to resume execution of the virtual machine.

The disconnected virtual machine will be automatically logged off by the VM/370 system 15 minutes after either of the following conditions arise:

- An attempt by the virtual machine's operating system to read from the disconnected terminal.
- The virtual machine going into a disabled WAIT state.

If, during a terminal session, your line connection is broken because of terminal, line, or TP control unit errors, VM/370 will place your virtual machine in disconnect mode and stop execution. You have 15 minutes to log on again. If the logon attempt is successful, you can resume operation by issuing the BEGIN command. If you do not succeed in logging on within the allotted 15 minutes, VM/370 will automatically log you off.

PLACING THE TERMINAL IN A DORMANT STATE

With the SLEEP command, you can place your virtual machine in a dormant state where it is not running but will display any messages sent to it. However, your connection time for accounting purposes is still being counted. You can, additionally, specify a sleep interval after which the virtual machine will automatically awaken. This interval ranges from zero time to 99 hours. For example, the command:

```
sleep 30 min
```

will stop execution in the virtual machine for 30 minutes after which time execution resumes automatically.

TERMINATING A TERMINAL SESSION

When you have completed your terminal session, you log off from VM/370. If you are already in the CP environment, you need only issue the command:

```
logoff
--or--
logoff hold
```

The latter form of the command implies that you are operating on a dial-up line and want VM/370 to retain your terminal connection. This will allow you, or another user, to log on from the same terminal without redialing the VM/370 system.

If you are in a virtual machine environment at the time you want to log off (under control of DOS/VS, OS/VS, or some other operating system) you can enter the CP environment by (1) pressing the Attention key (or its equivalent) the appropriate number of times depending on the TERMINAL MODE setting of your virtual machine, or (2) issuing the command:

```
#cp
```

When in the CP environment, you can issue the LOGOFF command with or without the HOLD option.

While the virtual machine is running, you may enter:

```
#cp logoff
```

and log off directly from the virtual machine environment.

The VM/370 system responds to the LOGOFF command with the following:

```
CONNECT=hh:mm:ss VIRTCPU=mmm:ss.hs TOTCPU=mmm:ss.hs
LOGOFF AT hh:mm:ss zone weekday mm/dd/yy
```

and disconnects your terminal from the system unless the HOLD option was invoked. All of your virtual machine's active spool files are closed, any temporary disk space is returned to the system, and any dedicated devices are detached. Accounting records are created for your connect and CPU time, for each dedicated device, and for each temporary disk space used.

SECURITY CONSIDERATION

You should always terminate a session via the LOGOFF command. If you merely turn off the power to your terminal, VM/370 will not automatically log you off until fifteen minutes after it senses that the power has been turned off. While VM/370 will immediately sense a power off condition from a typewriter terminal or a remote display terminal, it may be hours before this same condition may be sensed from a local display terminal. The only way VM/370 can sense that a local display terminal has powered down is to attempt to send it a message and receive the error code in return indicating that the terminal is turned off. Should anyone turn power on to a local display terminal before the automatic logoff has taken place, they would have unauthorized use of your virtual machine for as long as they desired. Thus, the time period in which someone can gain unauthorized access to your virtual machine is much greater if you are using a local display terminal.

For security's sake, log off before turning your terminal off.

The VM/370 directory entries for your virtual machine include, among other information, the I/O configuration required to run whatever operating system you will be using in that virtual machine. This will usually include a virtual console, virtual printer, a virtual card reader, a virtual card punch, and virtual disks. The virtual addresses that appear in the directory entries correspond to the device addresses that you specified when you generated your operating system.

Some devices such as unit record devices are usually defined as being spooled; in this way, a few real unit record devices can support a large number of virtual unit record devices.

Other devices such as magnetic tapes require a one-to-one virtual to real correspondence. This means that for its period of use, the device must be dedicated to one virtual machine. For this reason, these devices are not permanently assigned but are temporarily attached to a user as needed.

Still other devices are those that even though supported by your virtual operating system, are not supported by VM/370. In most cases, if the real device is available to the System/370 on which VM/370 is running, it can be attached to your virtual machine in such a way that your virtual operating system does all the input and output handling and VM/370 is effectively bypassed.

VIRTUAL DISKS

Under VM/370 a single real direct access storage device (DASD) can be managed as if it were made up of a number of virtual disks (sometimes called minidisks).

Virtual disks, to VM/370, are extents on real DASD. To the virtual machine, they are functional equivalents of real disks. They can range in size from 1 cylinder to the size of the real DASD volume.

Virtual disks can be permanently or temporarily defined for your virtual machine.

PERMANENT VIRTUAL DISKS

Permanent virtual disks are defined in your VM/370 directory. They can be your own personal disks which you may or may not wish to have other users access; or, they may be common disks, owned by one user, but generally shared in read-only mode by any user on the system. Virtual disks, defined in your directory, are made available to you when you log on to the system. Typical entries would look like:

```
LINK CMSSYS 490 190 RR
MDISK 191 3330 50 5 UDISK1 WR RPASSWD
MDISK 192 3330 20 10 UDISK2 WR RPASSWD
```

The LINK entry makes the virtual disk that userid CMSSYS has at his virtual address 490, available to your virtual machine in read-only mode at virtual address 190.

The MDISK statements define two minidisks with virtual addresses 191 and 192. The first, at address 191, occupies cylinders 50 through 54 of a 3330 volume with a serial number of UDISK1. The second, at address 192, occupies cylinders 20 through 29 of a 3330 volume with a serial number of UDISK2.

The read password entry means that anyone with the proper password can share the use of the minidisk in read-only mode.

The write access mode (WR) means that you can write to the disk as long as no one else has a link to it; if someone else links to it before you log on, you still have read-only access.

For detailed information on permanent virtual disks, see the section "Directory Control Statements" in the VM/370: Planning and System Generation Guide.

TEMPORARY VIRTUAL DISKS

If during a terminal session you require additional disk space, you can define a temporary minidisk via the CP DEFINE command:

```
define t2319 as 133 cyl 15
```

In the above example, a virtual 2319 disk comprising 15 cylinders will be allocated to your virtual machine at virtual address 133. You can then notify your operating system of the additional storage space using the appropriate control statements or commands.

Since there is no way of knowing the previous structure or use of this temporary disk space, you must format it to conform to the operating system you are using. If you are using CMS, use the CMS FORMAT command. For OS, DOS or VSAM applications use the IBCDASDI program.

When you have no further need for the above temporary disk space, you can release the space to the system by issuing the command:

```
detach 133
```

If you do not release it during your terminal session, it will be automatically released to the system when you log off.

Note: When temporary disk space is released to the system, it is not automatically cleared. Another user, requesting temporary disk space and receiving all or part of your former disk area, can access any data that you had left there. To preserve security, you should clear all temporary disk space before detaching it or logging off.

You can also gain temporary access to someone else's permanent virtual disk during your terminal session. You must know the userid of the disk's owner, as well as its virtual address in his system. If the owner is controlling the access to his disk, you'll have to obtain the read or write password. You can then issue the command:

```
link to smith 330 as 134 rr rpasswd
```

The virtual disk at address 330 in user SMITH's configuration will be made available to your virtual machine at address 134. You will have read-only access even if SMITH has the disk in write status.

VIRTUAL UNIT RECORD DEVICES

The VM/370 directory entries for your virtual machine will most probably define at least one reader, punch and printer. These are the virtual unit record devices that your operating system will address when performing unit record input and output. Rather than dedicate a real device to each virtual unit record device, VM/370 uses the concept of spooling to provide a buffered interface between the few real unit record devices and the relatively large number of virtual unit record devices.

VIRTUAL UNIT RECORD SPOOLING

Whenever your operating system creates a punch or printer file, the output data is organized into a spool file by VM/370. It is then placed in a queue by device type and stored on auxiliary DASD for eventual processing by the real unit record device. Similarly when card input data is read on a real card reader, VM/370 organizes the data into a spool file and places it in the queue for the specified virtual card reader.

Spool files are assigned certain characteristics that enhance their manageability by VM/370. Some of these characteristics such as spooling class and distribution code are assigned in the VM/370 directory entries. Other characteristics such as number of copies, concatenation of files, destination, and hold status are set to default values whenever you log on

The CP SPOOL and CHANGE commands allow you to change some or all of the above characteristics. To change spool file characteristics by device, use the SPOOL command; to change characteristics by file, use the CHANGE command.

The identification of individual spool files by VM/370 is accomplished by assigning each file a spool identification number (spoolid). This number ranges from 0001 through 9900 after which it restarts at 0001. One series of spoolids covers the reader, printer, punch, and console spool files. The spoolid is normally assigned to a spool file when that file is closed. An exception to this is a console spool file where the number is assigned when the file is opened. Also, if you close a console file without stopping console spooling, VM/370 will automatically open another console spool file with a new spoolid at the same time.

SPOOL FILE CHARACTERISTICS

The spooling class of a virtual device logically groups its output with that from similar virtual devices belonging to other users. For example, you could spool all printer output that required two-part green striped forms as class G. The real printer could then be set up with two-part green striped forms and told to process only class G spool files. If you issue the command:

```
spool 00e class g
```

all subsequent output of the virtual unit record device at address 00E will have a spool file class of G. This command is used to set the spooling class of files before they are closed.

If you decide to change the class of a spool file that has been closed but not selected for processing by a real device, you can issue the command:

```
change printer 1234 class m
```

and the printer spool file with a spool identification (spoolid) number of 1234 will have its class changed to M. If you don't remember the spoolid of the spool file, you can issue the command:

```
query printer all
```

and all printer spool files that have not been processed or selected for processing will have a one line description displayed at the terminal. The information displayed includes the spoolid number and the filename and filetype as well as other spool file characteristics.

The HOLD/NOHOLD status is a characteristic of a spool file that determines what happens to the file after it is closed. For example:

```
spool printer hold
```

will prevent the release of all subsequent printer output spool files to the real printer. If many files are being generated, of which only a few are required to be printed, you can put a blanket hold on all files with the HOLD option. You can then release only the required files via the NOHOLD option of the CHANGE or CLOSE commands.

If, when you are about to close a file, you know that the file is to be printed; you can issue the CLOSE command with the NOHOLD option:

```
close printer nohold
```

The HOLD status of the SPOOL command is overridden for that one file and it will be placed on the real printer queue with a NOHOLD status.

If you do not know which files are to be printed until they are all closed, you can use the QUERY command:

```
query printer all
```

to determine the spoolid numbers of all the closed printer files. Then issue the CHANGE command to alter the HOLD status of each file to be released. For example, if the file with a spoolid of 0246 is to be released, enter the command:

```
change printer 0246 nohold
```

In order to change your printer so that all subsequent files will be automatically released to the real printer, enter:

```
spool printer nohold
```

If your virtual reader has the default status NOHOLD, input spool files are deleted from the system after they have been read. To prevent this, enter:

```
spool reader hold
```

Reader files will now be held in the system until you issue:

```
spool reader nohold
```

and re-read the files; or, if you want to delete the files immediately, enter:

```
purge reader all
```

Again you have the option of using the CHANGE command to control the HOLD status of specific reader files via the spoolid number.

The CONT/NOCONT status of a spooling device controls the concatenation of files. Since this is a characteristic of a spooling device it can be changed only by the SPOOL command. When CONT is in effect, the spooling device will ignore input end-of-file indicators and output CLOSE requests.

For output devices, the effect is to concatenate multiple output files into one logical spool file.

For input devices the effect is to keep reading files, ignoring end-of-file indicators, until all files spooled to that virtual reader have been read. At that time, the end-of-file indicator that is reflected to the virtual machine depends on the EOF|NOEOF option setting of the SPOOL READER command. EOF results in a unit exception; this corresponds to pressing the end-of-file button on a real card reader. NOEOF results in a unit check/intervention required status.

VIRTUAL CONSOLE SPOOLING

While spooling is usually associated with unit record equipment, VM/370 will also spool both input and output data that is displayed on your virtual console. To start console spooling, enter the command:

```
spool console start
```

If, for some reason, you wish to spool your console data and not have it displayed on the terminal you can enter:

```
spool console start noterm
```

Note, however, that the NOTERM option (or its default value, TERM) is not effective until console spooling has been started. Also, the NOTERM option will not prevent the printing of:

- CP commands entered from CP mode.
- Commands entered on a 3270 in CMS EDIT mode.

When you wish to stop console spooling, enter the command:

```
spool console stop
```

This will stop any further spooling of console data but will not close the console spool file; for that you must enter the command:

```
close console
```

The converse is also true; closing the console spool file does not automatically stop console spooling.

REORDERING AND PURGING SPOOL FILES

If you would like to change the order of your closed spool files you can do so by device type. You can specify the new order either by spoolid or class or both. For example:

```
order printer 2468 1357 class c
```

will re-order your closed printer spool files as follows:

```
file 2468
file 1357
all class C files (in their original sequence)
other files in original sequence
```

You can remove any of your closed spool files from the system with the PURGE command. Files can be specified individually by spoolid, by class, or both.

```
purge 1234 class b
```

will remove file 1234 and all class B files from the system.

```
purge punch all
```

will remove all punch files.

```
purge all
```

will remove all files from all devices.

The CHANGE, ORDER and PURGE commands can be used only on closed spool files that belong to you and have not yet been selected for processing.

TRANSFERRING SPOOL FILES (LOCALLY)

Spool files can be transferred between users via the SPOOL and TRANSFER commands.

In order to transfer printer or punch output of your virtual machine to the virtual reader of some other user use the SPOOL command with the TO option.

```
spool printer to jones
```

will transfer all subsequent printed output of your virtual machine to the designated user's (userid JONES) virtual reader, while:

```
spool 00e to jones
```

will transfer only the subsequent output of a single device.

If you only want to print or punch a file for another user and not have it transferred to his virtual reader you can use the SPOOL command with the FOR option:

```
spool punch for jones
```

Subsequent punched output of your virtual machine will be identified with the userid and distribution code of user JONES.

If you wish to transfer a file with a spoolid of 0824 from your virtual reader to user jones' virtual reader, enter the command:

```
transfer 0824 to jones
```

You can also transfer all files of a particular class:

```
transfer class b to jones
```

If, for some reason, you wish to reclaim a file or files that you have spooled or transferred to some other user, you can use the TRANSFER command with the FROM option.

```
transfer 0824 from jones
```

will reclaim a specific file.

```
transfer class b from jones
```

will reclaim all files with a spool class of 'B'.

```
transfer all from all
```

will reclaim all the non-processed files that you had previously spooled or transferred to others.

Notes:

1. Files that have been spooled FOR another user, are not in that user's virtual reader and therefore cannot be reclaimed by you.

2. You can transfer any file that is queued on your virtual reader; however, you can reclaim only those files for which you were the originator.

TRANSFERRING SPOOL FILES (REMOTELY)

If the VM/370 system on which you are logged is running the Remote Spooling Communications Subsystem (RSCS), you can transfer your output files to:

- Remote non-programmable terminals
- Remote programmable stations
- Remote HASP/ASP type batch systems

All VM/370 output spool files contain a 136-byte information field called the spool file tag. When a file is to be transferred to a remote location, you must use the TAG command to insert destination and other information about the remote location into this spool file tag. The information required by RSCS and it's format is contained in the VM/370: Remote Spooling Communication Subsystem (RSCS) User's Guide. The minimum requirement, the location identifier, will be used in the following examples.

The TAG command can preset the spool file tag for a class of devices or a particular device as follows:

```
tag dev printer houston
```

will insert the location identifer, HOUSTON, into the spool file tag of all subsequent printer output files, while:

```
tag dev 00e houston
```

will do the same, but only for output files produced on the device at the address 00E.

When you have set the spool file tag with the appropriate location identifier, you must spool the files to the virtual reader of the RSCS virtual machine for actual transmission.

```
spool printer to net
```

will spool the files to be transferred to the virtual reader of the RSCS virtual machine whose userid is NET.

For punch files, representative entries can be:

```
tag dev 00d paloalto
spool 00d to net
```

To check the spool file tags for the above devices you can request a display of the TAG information for the device at 00E:

```
tag query dev 00e
```

You will receive the response:

```
PRT 00E TAG:
HOUSTON
```

For the punch, enter:

```
tag query dev 00d
```

and you will receive the reply:

```
PUN 00D TAG:
PALOALTO
```

Once you have closed a spool file that is spooled to another user, it is no longer available for you to change or query the spool file tag. If the file has not been selected for processing by the RSCS virtual machine, you can use the TRANSFER command to get the file back to your virtual reader. You can then use the FILE option of the TAG command to change a tag that was previously set. Determine the spoolid via the QUERY RDR ALL command. If the spoolid turns out to be 0834, enter:

```
tag file 0834 newdest
```

and the previous location identifier will be replaced with NEWDEST. To confirm the change you can enter:

```
tag query file 0834
```

and the response will be:

```
NEWDEST
```

DEDICATED DEVICES

A device is referred to as being dedicated if its use is restricted to a single virtual machine. Some devices, like disks and unit record

equipment can function in dedicated mode at one time and in shared or non-dedicated mode at other times. Devices such as magnetic tape drives require a one-to-one relation between the virtual and real device and therefore can only be used as dedicated devices. Still other devices, like the 1287 Optical Reader, are not supported by VM/370 and the virtual machine's operating system must handle all error recovery and error recording procedures; running in dedicated mode will allow this.

A device can be dedicated via a DEDICATE control statement in the VM/370 Directory entry for a virtual machine or it can be dedicated dynamically via the system operator's ATTACH command.

If the directory entry for your virtual machine includes a statement such as:

```
DEDICATE 495 295
```

the device at real address of 295 will be made available to your virtual machine as virtual address 495 when you log on to the system. It will remain dedicated to your virtual machine until you either log off, or release the device via the DETACH command:

```
detach 495
```

The format of the DEDICATE control statement is described in detail in "Part 2: Defining your VM/370 system" of the VM/370: Planning & System Generation Guide. More than one user can have the same real device specified as being dedicated to his virtual machine. However, the first user to log on will gain access to the device and others will have to wait until the current user either logs off or releases the device.

If you are using the DEDICATE statement for a DASD, you can specify a particular disk volume by its volume serial number rather than by its real device address. This has the effect of not tying you down to a particular real device. In the event that a malfunction precludes the use of the volume on one DASD, the disk pack can be transferred to another DASD and still be accessed by its volume serial number.

If a particular type of device that cannot be shared, such as a magnetic tape, is not required for the entire terminal session, it may be more practical to dedicate it as required. Since the ATTACH command is a class B command and not usually available to the general user, you can send the system operator a message:

```
msg operator pls attach 281 to smith as 181
```

The operator will issue the command:

```
attach 281 to smith as 181
```

If the device (assumed to be a magnetic tape) was available and the command was completed successfully, you will receive the verification response:

```
TAPE 181 ATTACHED
```

When you have no further use for the device, you should issue the command:

```
detach 181
```

You will receive the acknowledgement:

```
TAPE 181 DETACHED
```

and the device is now available to be attached to some other user. If you do not DETACH the device, it will remain dedicated to your virtual machine until you log off.

DEDICATED CHANNELS

A user can have an entire channel with all its devices dedicated to his virtual machine. CP will not translate device addresses, since the virtual addresses must be the same as the real device addresses. Contention for use of the channel is minimized, since all of the channel resources are dedicated to a single virtual machine. The ATTACH CHANNEL command is a class B command and is not usually available to the general user. However, you can send a message to the system operator:

```
msg operator pls attach channel 2 to smith
```

If the channel is available, the operator will issue:

```
attach channel 2 to smith
```

and when the command has been successfully completed, you will be notified by the response:

```
CHANNEL 2 ATTACHED
```

VM/370 provides several commands with which you can simulate hardware interruptions to your virtual machine, enter CP commands while in a virtual machine environment, and reconfigure your virtual machine dynamically, during a terminal session.

SIMULATING INTERRUPTS

The EXTERNAL command can be used to simulate an external interrupt to the virtual machine and return control to that virtual machine. If you enter the command:

```
external
```

a default interrupt code of x'40' is assumed; this corresponds to pressing the External Interrupt button on the real system control panel. You can also specify an interrupt code within the range of x'01' through x'FF':

```
external A8
```

If your virtual machine has the ECMODE option specified in the VM/370 directory, you can also code, x'1004' (Clock Comparator interrupt) and x'1005' (CPU Timer interrupt). The interrupt code that you enter or default to, is placed in position 16 through 31 of the PSW if you are operating in BC mode. If your machine is in EC mode, the interrupt code is placed in the two bytes at location x'84'. An external interrupt is then presented to your virtual machine and subsequent action is determined by your operating system.

The Attention or Request buttons on the real console can be simulated by the CP commands:

```
  attn
-- or --
request
```

In either case the effect is to interrupt the running condition of the virtual machine and ready it for console input.

ENTERING CP COMMANDS

There are several ways that you can interrupt the running of your virtual machine in order to execute CP commands. You can stop the virtual machine and place it in a virtual console read environment; you can go from a virtual console read environment to the CP environment; and, you can stop a virtual machine and place it directly into the CP environment.

ENTERING CP COMMANDS WHEN THE VIRTUAL MACHINE IS RUNNING

If your terminal mode is set to VM, pressing the Attention key (or its equivalent) once, will stop the virtual machine and place it into the virtual console read environment. You can now execute CP commands by entering one or more command lines as operands of the #CP command; for example:

```
        #cp query time
           -- or --
#cp query time#query users
```

where the latter example shows how multiple CP command lines can be entered.

Note: The pound sign (#) in these examples represents the logical line end character currently in effect.

If you are using a 3270 display terminal where the keyboard is not locked when the virtual machine is running, you can enter either of the following commands:

```
 #cp attn
 -- or --
#cp request
```

to both stop the virtual machine and place it in a virtual console read environment.

On a 3270 terminal, you can also enter a command such as:

```
#cp query time
```

while the virtual machine is running. The control program (CP) will interrupt the virtual machine, execute the CP command, and restart the virtual machine.

The #CP command, entered without any command line operands, will place the virtual machine in the CP environment. You can then enter CP commands directly. For example, the following sequence:

```
#cp
query time
query users
```

will first place you in CP mode and then execute the two CP commands.

To restart the virtual machine, enter:

```
begin
```

ENTERING CP COMMANDS FROM THE VIRTUAL CONSOLE READ ENVIRONMENT

When your virtual machine is stopped and in a console read environment you can either enter CP commands directly or you can place the virtual machine in the CP console function mode.

To execute CP commands directly, enter one or more CP command lines as operands of the #CP command. For example, if you enter:

```
#cp query time#query users
```

the virtual machine will execute the QUERY TIME and QUERY USERS commands and return to the virtual console read environment. You can restart the virtual machine by entering the command

```
#cp begin
```

If you enter the #CP command without any operands, the virtual machine will be placed in CP console function mode. You can then enter CP commands such as:

```
query time
 -- or --
query users
```

directly. After each command, the virtual machine returns to the CP console function mode. Entering the command:

```
begin
```

will return the virtual machine to the environment from which the #CP command was issued, in this case, the virtual console read environment.

ENTERING CP COMMANDS FROM CP CONSOLE FUNCTION MODE

When you are in CP console function mode, you can enter CP commands such as:

```
query time
 -- or --
query users
```

directly, and you remain in CP mode until you issue the BEGIN command. The command:

```
begin
```

returns you to the environment from which you entered the #CP command that placed you in CP mode. This would be either the virtual machine running or the virtual console read environment.

RECONFIGURING THE VIRTUAL MACHINE

The configuration of your virtual machine that is stored in the VM/370 directory can be altered during a terminal session to conform to specific situations that arise. A seldom used compiler may need additional disk work area or a larger virtual storage. A new application program in test status may need additional I/O devices or require a different channel mode of operation.

The CP DEFINE command allows you to change your configuration temporarily, for the current terminal session only. For example:

```
define reader 00b
```

temporarily adds a card reader at virtual address 00B. The entry:

```
define t3330 as 291 cyl 125
```

adds a temporary virtual disk, containing 125 cylinders, at virtual address 291.

If your virtual machine is operating in basic control (BC) mode, you can only define virtual devices with addresses up through 5FF; in extended control (EC) mode, you can use all addresses through FFF.

You can enter:

```
define storage as 768k
```

to temporarily change the virtual storage size of your virtual machine to 768K (K=1024 bytes). When you redefine storage, your virtual machine is automatically reset and you must reload your operating system.

Note: When defining virtual devices, you must be careful to specify virtual address that will not result in conflict or contention in the virtual control unit interface. See the "Control Statements in General" in "Section 7. Configuring Your Virtual Machine."

VM/370 has several CP commands that you can use in addition to the testing and debugging facilities that are contained in your virtual machine's operating system. These instructions can be used to locate, display, trace, and alter the program instructions that are running in your virtual storage.

In the following discussions of the ADSTOP, DISPLAY, DUMP, and STORE commands, it is important that you understand what levels of storage can be specified on the command line.

Only first level storage (storage that is real to the virtual machine) can be handled directly. This includes the V=R partitions or regions of DOS/VS and OS/VS as well as storage in OS/PCP, MFT, and MVT. The V=V partitions or regions of DOS/VS and OS/VS constitute second level storage and cannot be specified directly. The user, or the virtual operating system, is responsible for converting any second level storage addresses to first level storage addresses before including them in the specific command line.

STOPPING EXECUTION OF YOUR VIRTUAL MACHINE

In order to stop execution of your virtual machine at a given address in virtual storage use the ADSTOP command and specify the hexadecimal address of a virtual instruction. The command:

```
#cp adstop 3000
```

will stop the virtual machine when the instruction at hexadecimal location 3000 is the next instruction to be executed. When the machine stops running, you will receive the message:

```
ADSTOP AT 3000
```

and your terminal will be placed in CP console function mode. At this point you can enter other CP debugging commands to display and alter storage or to trace certain instructions. When you want to resume running your virtual machine, enter:

```
begin
```

DISPLAYING VIRTUAL STORAGE

The contents of virtual storage, storage keys, general registers, floating point registers, control registers (if in EC mode), PSW, CAW, and CSW can be displayed on your terminal via the DISPLAY command. All but the CAW and CSW can be printed on the spooled virtual printer via the DUMP command.

The DISPLAY and DUMP commands are fully described in the VM/370: CP Command Reference for General Users. The examples that follow do not attempt to show all possible uses.

TERMINAL OUTPUT

With the DISPLAY command, you can display virtual storage at your terminal in either of the following formats:

- Four byte groups, aligned on full word boundaries, hexadecimal format, printed four fullwords per line.
- 16 byte groups, aligned on 16 byte boundaries, hexadecimal format, printed four fullwords plus EBCDIC translation per line.

For the first format, enter the DISPLAY command as:

```
display 1026-102c
```

you will receive the response:

```
001024 xxxxxxxx xxxxxxxx xxxxxxxx
```

For the second format, enter the command as:

```
display t1026-102c
```

and the response will be:

```
                                                  (EBCDIC trans.)
001020 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx *................*
```

You can also specify the area of storage to be displayed by entering a hexadecimal byte count such as:

```
display 1024.12
```

The response will display 20 bytes as follows:

```
001024 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
001034 xxxxxxxx
```

Byte Alignment on Terminal Output

The above responses illustrate the byte alignment that takes place in each of the two display formats.

If the first location to be displayed is not on the appropriate 4 or 16 byte boundary, it is rounded down to the next lower boundary that applies.

If the last location to be displayed does not fall at the end of the appropriate 4 or 16 byte group, it is rounded up to the end of that group.

If you enter:

```
display k1024-3200
```

the storage keys that are assigned to each 2K segment of the specified storage area are displayed. Contiguous 2K segment with identical storage keys are combined; for example, the response to the above could be:

```
001000 TO 0027FF  KEY=F0
002800 TO 003800  KEY=E0
```

To display all storage keys, enter:

```
display k
```

If your virtual machine is in extended control mode, you can interrogate any of the control registers:

```
display X1 4 A
```

and receive the response:

```
ECR  1 = xxxxxxxx
ECR  4 = xxxxxxxx
ECR 10 = xxxxxxxx
```

However, the same command entered while your virtual machine is not in extended control mode will result in the response:

```
ECR 0 = xxxxxxxx
ECR 0 = xxxxxxxx
ECR 0 = xxxxxxxx
```

As each operand in the command line is processed, VM/370 determines that ECMODE is OFF and replaces any reference to a control register with ECR 0, the only control register available in basic control mode.

PRINTER OUTPUT

With the DUMP command you can dump the contents of all registers, the PSW and the storage keys, along with any specified area of virtual storage, to the virtual machine's spooled printer. The printer format for storage locations is 8 fullwords per line plus the EBCDIC translation on the right.

To print only the registers, PSW, and storage keys, you need only enter:

```
dump 0
```

To also print an area of virtual storage, you can specify the beginning and ending hexadecimal locations:

```
dump 1064-10ff
```

You can also specify the beginning location and the number of bytes to be dumped; both values are entered in hexadecimal:

```
dump 1064.9b
```

If you are outputting a series of dumps, you can identify each one by including its identification on the DUMP command line, following an asterisk:

```
dump 1000-2000 * dump no. 1
```

In order to print the dump data on the real printer you must first close the virtual printer. Issue the command:

```
close printer
```

and the dump data spool file is placed on an appropriate system printer queue.

ALTERING VIRTUAL STORAGE

You can alter the contents of your first level virtual storage, general registers, floating point registers, control registers (if available), and the PSW with the STORE command.

Virtual storage can be altered in either fullword or byte units.

When using fullword units, the address of the first positions to be stored must have either an L or no prefix:

```
store 1024 46a2
   -- or  --
store l1024 46a2
```

will result in 000046A2 being stored in locations 1024 through 1027.

```
store 1024 46 a2
```

on the other hand, implies storing 2 fullwords and will result in the storing of 00000046000000A2 in locations 1024 through 102B.

If the starting location is not a multiple of a full word, it is automatically rounded down to the next lower fullword boundary. Each fullword operand can be from 1 through 8 hexadecimal characters in length. If less than 8 characters are specified, they will be right justified in the fullword unit and padded to the left with zeroes.

You can store in byte units by prefixing the start address with an S.

```
store s1026 d1d6c5
```

will store D1D6C5 in locations 1026, 1027 and 1028. Note that the data storage is byte aligned. If an odd number of hexadecimal characters are specified, the last one is ignored.

You can store data into one or multiple consecutive registers.

General and control registers are loaded in fullword units.

```
store g4 123456
```

loads general register 4 with 00123456.

```
store g4 12 34 56
```

loads general registers 4, 5, and 6 with 00000012, 00000034, and 00000056, respectively.

Floating point registers are loaded in doubleword units. Each doubleword operand can be from 1 through 16 hexadecimal characters in length. If less than 16 characters are specified, they will be left justified in the doubleword unit and padded to the right with zeroes. For example:

```
store y2 00123456789
```

will load floating point register 2 with the value 0012345678900000.

You can use the STATUS operand of the STORE command to simulate the hardware store status facility. Selected virtual machine data is stored in permanently assigned areas in low storage. Your virtual machine must be in extended control mode in order for the command:

```
store status
```

to be accepted. To place your virtual machine in extended control mode, issue the command:

```
set ecmode on
```

Be aware that this command will reset your virtual machine and you must reload your operating system.

The data that is stored by the STORE STATUS command is summarized in the following table:

Virtual Address Dec.	Hex.	No. of bytes	Data
216	D8	8	CPU Timer
224	E0	8	Clock Comparator
256	100	8	Current PSW
352	160	32	Floating Point Registers (0,2,4,6)
384	180	64	General Registers (0-15)
448	1C0	64	Control Registers (0-15)

Note: If the operating system that is running in your virtual machine operates in the basic control mode, the above areas of low storage may be used for other purposes. You should not use this facility under these conditions.

TRACING VIRTUAL MACHINE ACTIVITY

The TRACE command allows you to trace certain virtual machine activities and record the results on either your terminal, the virtual printer, or on both.

You can trace any one or more of the following virtual machine activities:

- SVC interruptions
- I/O interruptions
- Program interruptions
- External interruptions
- Privileged instructions
- PSW instructions
- Successful branches

Whenever you are recording trace output at your terminal, the virtual machine will stop execution and enter the CP console read environment after each output line. This is the default mode of operation when, for example, you enter:

```
trace all
-- or --
trace svc program branch
```

If you only want to record the trace and not stop after each output line, add the RUN operand as the last entry on the command line.

If, having specified the above multiple activities to be traced, you decide to stop tracing one or more of them, enter:

```
trace program branch off
```

and tracing will now be confined to SVCs only.

To trace all activity with the output directed to the virtual printer, enter:

```
trace all printer
```

When you stop tracing, you must also issue the CLOSE command for the printer in order for the spooled trace output to be printed on the real printer:

```
trace end
close printer
```

If your virtual machine configuration contains only one printer, trace output will be intermixed with application output. You should define another virtual printer with an address lower than the previously defined printer. Application output will still be directed to the original printer; however, trace output is always directed to the printer with the lowest address. For a complete description of the TRACE command along with the trace output formats, see the VM/370: CP Command Reference for General Users.

You may require the facilities of more than one operating system during a single terminal session.

When you run an operating system such as OS, DOS, OS/VS1, OS/VS2, or DOS/VS from a terminal, you can use the CMS Editor facilities to create and modify job streams and to analyze the result and output.

If you are an applications programmer who normally uses CMS to interactively create, modify, and test your programs, you may require facilities for compilation or execution that are not supported or available in CMS.

The following technique uses multiple operating systems consecutively. Job control cards, compiler or assembler source programs, and test data streams are created and modified at the terminal under control of the CMS Editor. The job stream is then executed, by passing control to an appropriate operating system with the necessary facilities.

In this way, you are using the terminal-oriented facilities of CMS to create and update source programs and JCL. When you are ready to compile or test, you give control of your virtual machine to DOS or OS. After execution is finished, you can give control back to CMS for selective scanning and displaying of the printer and punch output at the terminal.

This approach assumes you have created source program files and data files under CMS. To execute under another operating system (in this example, OS) you must also create JCL records that specify the compilation, link edit, or execution, as appropriate. These records are created under CMS and named with a distinctive filename and filetype (for example, PLICOMP JCL). Job control records, source program files, and data files can then be merged together in the virtual card reader to form a single OS job stream. The CP and CMS commands (shown in Figure 4) create and transfer this job stream.

TRANSFERRING OUTPUT

The CP SPOOL command transfers card images from the virtual card punch of one virtual machine to the virtual card reader of that same or some other virtual machine. During this time, no real cards are punched or read; CP manages the transfer of CMS card-image data files through disk spooling operations only.

Figure 4 shows how you can punch files to yourself; or, to be more exact, to your virtual card reader. The virtual machine is in the CMS environment at the start of the example. The command "SPOOL 00c cont eof" specifies that reading be continuous until all files spooled to the virtual machine are exhausted and the virtual end-of-file button on the reader is pushed. NOHEADER specifies that no special control cards are to be inserted in front of each punched file. Virtual device 230 is an OS system volume. Virtual device 231 contains the OS job queue, SYS1.SYSJOBQE. All standard CMS and OS responses are omitted from the example; however, the OS READY message is included to more fully illustrate the IPL sequence. Also, assuming you are using a 2741, you must press the Attention key before entering each OS command. The attention interrupts are not shown in Figure 4.

```
CMS
cp close 00c
cp purge 00c all
cp close 00d purge
cp spool 00d to * cont
punch jobcard jcl (noheader)
punch plicomp jcl  (noheader)
punch plimain pli  (noheader)
punch asmcomp jcl  (noheader)
punch asmsub assemble  (noheader)
punch linkgo jcl  (noheader)
punch godata dat (noheader)
punch slshstar jcl (noheader)
cp spool 00d nocont
cp close 00d
cp spool 00c cont eof
cp ipl 230

Note:  The following are issued once under OS control:

IEE007A READY
set date=xx.355,Q=(231)
start rdr,00c
start wtr,00e
start
```

Figure 4. OS Job Stream Transfer

To transfer files between systems, you must have access to both operating systems being used. Access to both systems can be provided either in your virtual machine's VM/370 directory entry, or dynamically before loading the new system.

Figure 5 illustrates a virtual machine configuration and the corresponding VM/370 directory control statements. Virtual device addresses 190 and 191 contain the CMS system and user disk area. Virtual device addresses 230 and 231 contain the OS system and user disk area. The two systems use a common card reader, card punch, printer, and console.

```
USER OS2 PASSWORD
 ACCOUNT NUMBER BIN16
  CONSOLE 01F 3215
  SPOOL  C 2540 READER
  SPOOL  D 2540 PUNCH
  SPOOL  E 1403
  LINK JFK 230 230 R
  LINK CMSSYS 190 190 RR
  MDISK 231 2314 120 82 UDISK1 WR
  MDISK 191 2314 101 10 UDISK1 WR RPASS WPASS
```

Figure 5. Directory Entry for Alternating Operating Systems

CONFIGURATIONS FOR ALTERNATING OPERATING SYSTEMS

You can alternate operating systems more simply if:

- Devices used by both systems are supported at the same device address, and
- Common addresses are not used to support different devices

If these two conditions are not met, you must modify the virtual machine configuration before each IPL of a new system.

If the two systems require online typewriter keyboards at different addresses, the CP DEFINE command can be used to change the address of the virtual system console. For example, the OS system specified above requires an online typewriter keyboard at address 01F while the CMS system probably will have its console at address 009. In this case, you must issue the command:

```
cp define 009 as 01f
```

before loading 230. Since CMS automatically communicates with any valid multiplexer address, you can leave the console at 01F and satisfy both systems.

If the systems expect different device types at the same address, the common address must be assigned to the appropriate device each time a new system is loaded. If CMS is running with a disk at address 191 and OS is generated to support a 3330 at that address, you should issue the following command before loading OS:

```
cp detach 191
```

An appropriate device can then be added to the virtual machine at address 191 either before loading, or in response to a mount request from the OS system.

Note: For direct access storage devices, the above procedure is necessary even if both systems support the same device type at the same address. Except for VSAM disks, the disk format used by CMS is unique, and is not compatible with that of other operating systems. Files can be shared between CMS and OS or DOS only through the spooling facilities of CP or through VSAM data sets.

Performance measurements apply to both the VM/370 system and the individual virtual machine. How well the system responds to the needs of the users is of prime importance to the system analyst. How efficiently the individual virtual machine makes use of the allotted storage, CPU, and I/O facilities is of prime importance to the general user.

CP provides certain commands that will allow both VM/370 and virtual machine performance to be tracked and measured; other commands allow the setting of certain options to improve performance.

For a complete description of the INDICATE and MONITOR commands that will be discussed in the following sections, see the "Performance Observation and Analysis" sections in the VM/370: System Programer's Guide.

VM/370 SYSTEM PERFORMANCE

THE INDICATE COMMAND FOR THE GENERAL USER

The general user can request limited system performance information by issuing the:

```
#cp indicate load
```

command. The response will reflect values of CPU usage, number of users being serviced, main storage usage, and main storage contention. Since these values are constantly fluctuating, CP takes periodic samplings and the displayed values are smoothed according to the frequency of the sampling. The response format is:

```
CPU nnn%  Q1-nn  Q2-nn  STORAGE-nnn%  RATIO-n.n
```

CPU nnn% represents the time that the system is running as opposed to being in a wait state.

Q1-nn and Q2-nn represent the number of users in the interactive and noninteractive queues, respectively, that are currently being serviced.

STORAGE-nnn% is a measure of real storage usage. It is the ratio of the sum of the estimated working sets of the users being serviced (Q1+Q2) to the number of pageable pages in the system. Due to the scheduling algorithm used, this value can exceed 100% at times.

RATIO-n.n is the ratio of the total number of active users to the number of users actually being serviced. The average number of users that are currently eligible but not being serviced can be calculated as:

```
(Q1+Q2) x (RATIO-1.0)
```

THE INDICATE COMMAND FOR THE SYSTEM ANALYST

If you are a system analyst with the privilege class E subset of commands, you can use the full set of operands of the INDICATE command. In addition to the system load values that are available to the general user, you can issue:

```
#cp indicate user userid
```

and determine, for the userid specified, the virtual machine's activity in terms of resources used and occupied as well as I/O events that have taken place. You can issue:

```
#cp indicate queues
```

and display each active user's identification, the queue he is in, his current status, the number of pages resident in real storage, and his working set size.

If you suspect excessive I/O contention, you can issue:

```
#cp indicate i/o
```

and the response will list all users that are in an I/O wait state at that time, along with the address of the real device to which the most recent SIO was mapped. If I/O contention seems to be a problem, you can use the SEEKS option of the MONITOR command to conduct a thorough trace of I/O activity. This command will be discussed in a subsequent section.

If your installation has both primary high speed and secondary lower speed paging devices, and the INDICATE QUEUES command response shows an excessive number of users in persistent page wait, you can use the command:

```
#cp indicate paging wait
```

to further analyze the condition. The response will list each user that is in page wait along with the hexadecimal number of pages that are allocated on the primary paging device and the number allocated on the secondary paging device. With this information, you can determine which users are monopolizing primary paging space. You can then use the INDICATE USER USERID command to determine if these users are still active. If not, they are occupying a critical resource that could be put to better use.

If you issue the command:

```
#cp indicate paging all
```

you will receive the page residency data for all users of the system.

THE MONITOR COMMAND FOR THE SYSTEM ANALYST

The Monitor command (privilege class E) is a data collection tool that performs the following:

- Stops and starts CP internal trace table data collection.
- Enables one or more classes of Monitor Call (MC) or timer driven interrupts.

- Starts and stops the recording of MC and timer driven interrupt data on to tape.
- Specifies or changes the interval to be used for timer driven interrupts.
- Displays the status of the internal trace table and each implemented class of data collection.

The START and STOP CPTRACE Operands

The CP internal trace table contains a chronological recording of certain key events that occur on the real machine. The contents are used mainly in debugging procedures when the system aborts. CP internal tracing is active whenever the system is initially loaded. To subsequently stop and restart tracing issue:

```
#cp monitor stop cptrace
        --and--
#cp monitor start cptrace
```

The ENABLE Operand

There are eight classes of data collection that you can enable with the MONITOR ENABLE command. See the VM/370: System Programmer's Guide for descriptions of each class.

Five of these classes are referred to as event data and are triggered by MONITOR CALL (MC) instructions embedded in the VM/370 code. Event data consists of terminal I/O, scheduler queue management, scheduling statistics, privileged instruction simulation, and DASD I/O requests.

The other three classes are referred to as sampled data and are triggered by timer interruptions at specified intervals of time. Sampled data consists of system resource utilization, user resource utilization, user status, and SIOs for DASD and tape devices only.

As an example, to enable the data collection class that monitors the activity of the scheduler, issue the command:

```
#cp monitor enable schedule
```

The START and STOP TAPE Operand

Enabling one or more data collection classes does not automatically start data collection. You must issue the command:

```
#cp monitor start tape raddr
```

where 'raddr' is the real address of the tape drive you are using to accumulate the collected data. The data is organized into variable length records and stored in two alternating page size buffers. As soon as one buffer is filled, it is written out to tape as the other buffer is being filled. The command:

```
#cp monitor stop tape
```

will stop data collection and write out the last partially filled buffer page. The tape file is then closed and the tape is rewound and unloaded.

Setting the Time Interval

There is a default 60 second time interval used for the three timer driven classes of data collection. If you wish to set a new time interval of 30 seconds you can issue the command:

```
#cp monitor interval 30 sec
        --or--
#cp monitor interval 30
```

To change the interval to 4 minutes, enter:

```
#cp monitor interval 4 min
```

The maximum allowable interval is 9 hours, entered as 540 minutes or 32,400 seconds. The minimum interval is 30 seconds. The interval can be changed at any time; however, if data collection is in progress, the new interval takes effect after the current interval has elapsed.

The interval is automatically reset to 60 seconds when any of the following occurs:

- The user issues the MONITOR STOP TAPE command.
- An unrecoverable I/O error stops the system.
- The end of tape is reached.

The DISPLAY Operand

The status of each data collection class including CP internal tracing can be interrogated with the command:

```
#cp monitor display
```

The response will itemize all classes by number and key word and tell you whether each class is enabled or disabled.

VIRTUAL MACHINE PERFORMANCE

As a general user, you can issue the INDICATE command to obtain data on your virtual machine's use of system resources. Each time you issue the command:

```
#cp indicate user
```

you will receive a two line response that will contain the following:

RES-nnnn the current number of virtual storage pages resident in main storage.

WS-nnnn — the most recent estimate of your working set size.

READS=nnnnnn[1] — the total number of page reads that have occurred since you logged on.

WRITES=nnnnnn[1] — the total number of pages written for you since you logged on.

DISK-nnnn — the current number of virtual pages allocated for you on the system paging disk.

DRUM-nnnn — the current number of virtual pages allocated for you on the system paging drum.

VTIME=nn:nn[1] — the total virtual machine time used since you logged on.

TTIME=nn:nn[1] — the total virtual machine time plus the total CPU time (virtual

SIO=nnnnnn[1] — the total number of non-spooled I/O requests that you have issued since you logged on.

RDR=nnnnnn[1] — the total number of virtual cards read since you logged on.

PRT=nnnnnn[1] — the total number of virtual lines printed since you logged on.

PCH=nnnnnn[1] — the total number of virtual cards punched since you logged on.

As a system analyst, with class E command privileges, you can issue the INDICATE command and specify any user that is currently active on the system. For example:

```
#cp indicate user smith
```

will return a two line response containing resource usage data pertaining to user SMITH. The format is the same as for the class G INDICATE USER command described above.

PERFORMANCE OPTIONS

After measuring the performance of both the VM/370 system and the virtual machines that it is supporting, you may want to redistribute system resources either to balance out any inequities or to favor one virtual machine over another. If you do, you must remember that the system consists of a finite amount of system resources; and, if more of a resource is given to one virtual machine, the rest must share whatever is left.

---

[1]Means that this value is one of the VM/370 accounting data items. As such, the value is reset to zero whenever the system operator issues the ACCT command as well as when the user logs off.

OPTIONS CONTROLLED BY THE GENERAL USER

As a general user, you have limited direct control over any of the performance systems available to your virtual machine.

Virtual Machine Assist Feature

You can invoke the virtual machine assist feature, providing that (1) the hardware feature is available on the System/370[1] and (2) the system operator has not turned the feature off with the class A command:

```
set sassist off
```

If neither of the above conditions preclude your use of the virtual assist feature, you can enter:

```
#cp set assist on svc
      --or--
#cp set assist on nosvc
```

and the CPU will intercept and handle interruptions caused by invalid page conditions, and several privileged instructions. If the SVC operand is specified, the CPU will also handle all SVC interruptions except SVC 76. The NOSVC operand specifies that CP is to handle all SVC interrupts. SVC 76 interruptions are always handled by CP. With the virtual machine assist feature handling the above interruptions, rather than CP, the performance of VM/370 is improved and the overhead in servicing users is reduced. Your virtual machine can also realize greater productivity during its alloted time slice; however, this will depend on the number of intercepted SVCs and privileged instruction.

Virtual = Real Option

The virtual=real (V=R) option, although available to any number of users, can be in effect for only one user at a time. For the V=R user, all pages of his virtual machine, except page 0 which is relocated, are locked in the real storage locations they would normally occupy on a real computer. This eliminates paging overhead for that virtual machine since all pages are in storage. It also eliminates the need for CCW translation since the virtual addresses are equal to the real addresses.

VM/370 systems that are to provide the virtual=real facility for their users should be generated with a real V=R storage area large enough to contain any one of the virtual machines that will be running in V=R mode. If the storage sizes of the V=R machines vary greatly, there may be times when part of the V=R area is not being utilized. If a user with storage requirements less than the reserved V=R area is logged on, the system can not use any of the idle V=R area for system or user paging space. Under these circumstances, it may be expedient to generate multiple VM/370 systems, each with a different size V=R area, and schedule production on each based on the least idle space.

---

[1]The virtual machine assist feature is available on the System/370, Models 135, 145, and 158. It is also available, via an RPQ on the Model 168.

For generating VM/370 systems with the virtual=real option, see the VM/370: Planning and System Generation Guide.

You, the general user, can avail yourself of the virtual=real option by requesting that the keyword VIRT=REAL be entered on the OPTION control statement of your virtual machine directory.

If both VM/370 and your virtual machine have the V=R option, your virtual machine will be placed into the V=R area when you log on, providing that (1) no other V=R user is logged on and occupying the V=R area, (2) the V=R area has not been released by a system operator via the UNLOCK command, and (3) your virtual machine size is less than or equal to the V=R area. If the V=R area has been released, VM/370 must be reloaded for the virtual=real option to become active again.

If another user is occupying the V=R area, or if the system is using the area, you will receive the message:

```
DMKBLD200E VIRT=REAL AREA NOT AVAILABLE; IN USE BY [userid]
           [SYSTEM]
```

and your virtual machine will be logged on but not in V=R mode.

If your virtual machine size is larger than the V=R area, you will receive the message:

```
DMKBLD202E VIRT=REAL AREA NOT AVAILABLE; INSUFFICIENT STORAGE
```

and your virtual machine will be logged on but not in V=R mode.

When the V=R area is free (and not unlocked by the system operator), you can move your virtual machine into the V=R area by either logging off and logging on, or by using the DEFINE command to redefine your virtual machine storage:

```
#cp define storage nnnnK
```

The storage size specified in the DEFINE command can be either your original size or a lesser size to comply with the V=R area. It is the command processing that effects the shift into the V=R area.

In order for CP to simulate an IPL sequence, normal CCW translation is in effect when you first log on. To stop CCW translation, the V=R user can issue the command:

```
#cp set notrans on
```

All subsequent I/O operations are then performed from the virtual CCWs in the virtual=real space, without translation.

Operating in the V=R area enhances the performance of a virtual machine due to the elimination of paging and CCW translation and is desirable when running virtual storage operating systems such as DOS/VS and OS/VS since you can eliminate double paging. It may be mandatory when running certain programs that execute self modifying channel programs or that are hardware timing dependent. For specific information, see "Dynamically Modified Channel Programs" "Appendix B: CP Restrictions."

OPTIONS CONTROLLED BY THE SYSTEM OPERATOR

The system operator can give a specific virtual machine certain performance options that will allow that user to operate in a more favorable environment.

Locked Pages Option

By locking into real storage one or more pages of a user's virtual storage, paging I/O for these pages is eliminated. This may be advantageous for sections of your operating system's nucleus that have high activity (page zero, for example). The command:

```
lock smith 6 d
```

will lock user SMITH's pages 6 through 13 while:

```
lock system 30 30
```

will lock the VM/370 system's page 30.

While this option may eliminate some paging, it is quite rigid and expects you to be able to identify the specific pages required to be resident. Note that under normal conditions, frequently used pages would tend to remain resident without the LOCK option.

Reserved Pages Option

You may request the system operator to reserve a number of pages for your virtual machine via the SET RESERVE command. The system operator can provide this option to only one virtual machine at any one time. The command:

```
set reserve smith 6
```

will ensure that user SMITH will always have at least 6 page frames available to his virtual machine. The one exception to this guarantee is when all other available page frames have been exhausted; under these circumstances, the system may temporarily retrieve one or more reserved pages.

This option is more flexible than having locked pages. You need not know in advance which pages should be resident; and, normal paging activity will result in the most active pages eventually occupying the reserved page frames.

Priority

Each virtual machine has a directory priority value assigned to it. It can be specified in the USER control statement of your VM/370 directory entry as an integer from 1 through 99. If not specified, the priority value defaults to 50. When your virtual machine is logged on it is constantly competing for an allotment of CPU time to execute its programs. When the control program decides that your virtual machine is eligible to execute instructions, your virtual machine is placed in an

eligible queue in ascending order of priority values. Users are taken from the head of the eligible list and placed in the dispatchable queue to receive their time slice of the CPU. Therefore, the lower the priority value the sooner the virtual machine is placed in the run queue.

While the priority value, defined or defaulted to in your directory entries, is in effect as soon as you log on, there may be times when you require a higher priority (a lower priority value) for a particular terminal session. The system operator can change your priority value for that session by issuing the command:

```
set priority smith 10
```

Note that priority comes into effect only in placing your virtual machine in the eligible list. When your virtual machine is taken from the eligible list, it is placed in the dispatchable queue at a position determined by your running characteristics, frequency and amount of I/O, paging activity, and terminal interactiveness. If your operating system is running a program that requires excessive I/O and paging operations, you may not realize much of an improvement with a better priority.

Favored Execution Option

The favored execution option provides the system operator with the means to alter the normal scheduling algorithms as applied to certain virtual machines. He can provide a basic favored execution option to each of a limited number of virtual machines and he can provide one virtual machine with the guaranteed percentage favored execution option.

BASIC FAVORED EXECUTION OPTION: The basic favored execution option is assigned to a virtual machine by having the system operator issue the command:

```
set favored userid
```

This option means that the specified virtual machine is not to be dropped from the dispatchable or active queue as long as it is executable. If it is dropped from the active queue, waiting for an I/O or paging operation to complete, it is immediately restored to the active queue at its priority position as soon as it becomes executable.

There is no expressed limit to the number of virtual machines that can have this option assigned to them; however, if their combined requirements for storage exceed the system's capacity, thrashing can occur and overall performance will suffer. Even if the storage requirements of the favored virtual machines do not exceed the system's capacity, they could leave so little storage for the non-favored users that they would, in effect, be prevented from ever becoming active.

GUARANTEED PERCENTAGE FAVORED EXECUTION OPTION: If one of the favored virtual machines requires an even more favored status, the system operator can assign, to that one favored machine, a guaranteed percentage of its CPU time slice. For example, the command:

```
set favored smith 80
```

will not only keep user SMITH in the active queue as long as he is executable, but will effectively assign him a priority value of zero until he has obtained 80 percent of his allotted time slice. This means that whenever SMITH becomes executable, he is the next virtual machine

dispatched. Normal scheduler algorithms determine whether the user is classified as interactive or noninteractive; the guaranteed percentage is multiplied by the corresponding time slice value.

If, in the above example SMITH is classified as an interactive user and VM/370 is running on a Model 145, the regular time slice of 300 milliseconds is multiplied by 80% and the result, 240 milliseconds, becomes the user's guaranteed time, if he can use it. Each time SMITH becomes executable, the remaining amount of his guarantee is checked. If he has not used 240 ms, he is placed at the top of the queue as the next virtual machine to be dispatched. If the guarantee has been used, he is placed in the active queue according to his normal priority. At the end of each time-slice, the guaranteed percentage is recomputed.

To remove the favored execution status from a user the system operator issues the command:

```
set favored userid off
```

Section 4. DOS/VS in a Virtual Machine

Section 4 contains operational information specific to DOS/VS and DOS.

The following topics are included:

INTRODUCTION

When you load a DOS/VS system into a virtual machine that is running under VM/370, your terminal becomes the operator's console of the DOS/VS system and you are responsible for entering all of the commands and responses normally required of the operator.

The publications Operator's Library DOS/VS Operating Procedures, GC33-5378, and DOS/VS System Control Statements, GC33-5376, contain explanations of the control statements and operator commands that you can enter from the console to start the system and control it. All of the messages and responses you receive, as well as suggestions for how you should respond, are documented in DOS/VS Messages, GC33-5379.

There are three basic techniques that you can use when you are running DOS/VS in a virtual machine:

1. Run the DOS/VS system in batch mode. The terminal is the operator's console, and other users may submit jobs either through the real system virtual card reader or from the virtual card punches of other userids.

2. Use the IPL command to alternate using the DOS/VS and the CMS system in a single virtual machine. Use CMS to prepare a job stream for DOS/VS, use DOS/VS to execute the job stream and CMS to check the output.

3. Log on to a userid and load the DOS/VS system. Once it is running, disconnect from that userid and log on to another userid while the DOS/VS userid continues working. When you want to check on the status of the DOS/VS system, disconnect from the current virtual machine and reconnect the DOS/VS virtual machine.

All three of these techniques are discussed in greater detail in the pages that follow. First, you must understand how to access the DOS/VS system residence volume, ensure that the proper I/O devices are attached to your virtual machine, and know how to IPL the DOS/VS system using VM/370.

ACCESSING THE DOS/VS SYSTEM

This section assumes that the DOS/VS system you are using in VM/370 has already been generated and that the system residence volume is available to you on a real disk or minidisk in read/write status. You can make the system residence volume available in any of these ways:

1. The DOS/VS system residence can be defined as a read/write disk in your VM/370 directory entry, or in the directory entry for the userid that you are using to run the DOS/VS system. A typical directory entry might look like the following:

```
MDISK 250 3330 101 50 VDOSYS MR RPASS WPASS
```

2. You can link to the DOS/VS system residence volume using the LINK command. For example, if the DOS/VS system residence is on the 150 disk in the directory entry for the userid DOSRES, you could enter

```
link dosres 150 250 w wpass
```

3. The VM/370 system operator can attach the volume containing the DOS/VS system residence directly to your userid, for your exclusive

use. If the operator (or another user with Class B command privileges) attaches the disk to you, no other user may access the volume.

Note: All of the console logs and command examples in this section assume a DOS/VS system residence attached to your virtual machine at virtual address 250.

SHARING THE DOS/VS SYSTEM

Only one user may access the DOS/VS system residence in read/write status at any one time, since only one user can update the label information cylinder, which is used by DOS/VS to keep a record of volumes and files needed by programs that are running.

However, if you need to share the DOS/VS system residence, you may set up a read/write core image library for link-editing. Then, you can share the SYSRES libraries, read-only, with other users. This method is acceptable if some action has been taken at your installation to provide individual standard label cylinders for each userid that accesses the DOS/VS system.

As a third alternative, you may have your own read/write copy of the system residence volume, that has a minimum of information on it, and share, in read-only status, private libraries with other users.

USING VIRTUAL UNIT RECORD DEVICES

When you use DOS/VS in a virtual machine you must have the following unit record devices, which are normally defined in your VM/370 directory entry:

- A virtual card reader, from which the DOS/VS system reads the jcb input stream. Usually, the card reader is at virtual address 00C (for a 2540) or 012 (for a 3505).

- A virtual printer, which receives all the SYSLST output generated during the operation of the DOS/VS system. Usually, the printer is at virtual address 00E (for a 1403) or 002 (for a 3211).

- A virtual card punch, which receives SYSPCH output generated during the operation of the DOS/VS system. Usually, the card punch is at virtual address 00D (for a 2540) or 013 (for a 3525).

  If you are using CMS to prepare jobs for your DOS/VS virtual machine, you use your virtual card punch, also, to spool jobs to the DOS/VS virtual machine.

The addresses indicated above are the standard addresses usually assigned in VM/370. Depending, however, on how the DOS/VS system you are using was generated, you may need a virtual device at a different address. For example, if the DOS/VS system expects a 3211 printer at address 002, and there is no printer at this address in your virtual machine configuration, you can define one with the CP DEFINE command:

```
define 3211 002
```

Before using the DOS/VS system, you should find out, from the programmer at your installation who is responsible for generating and maintaining the DOS/VS system, what the virtual device requirements are.

You can control your virtual unit record devices with the CP SPOOL command. Printed or punched output does not necessarily need to be printed or punched. For example, if you are using the flip-flop technique, you can spool your virtual printer to your card reader, as follows:

```
spool printer to *
```

so that you can read printed output onto a CMS disk and examine it. You can also use the SPOOL command to change the output spooling class of your spooled printer or punch files.

DEFINING THE OPERATOR'S CONSOLE

During DOS/VS system generation, an address is specified for the operator's console. Your terminal must also be at this address. Usually, DOS/VS expects the operator's console to be at real address 01F. The device has to be generated as a 3215, 3210, or 1052 console. However, you can use any terminal type as the virtual operator's console when you use DOS/VS in a virtual machine.

You can find out the address of your terminal by entering the command

```
query console
```

If the response indicates that your terminal is not at 01F, but at another address, such as 009 (which is a standard VM/370 console address), you can enter the command

```
define 009 as 01f
```

Your terminal can now function as the operator's console of the DOS/VS system as well as the console for CMS.

PREPARING JOBS FOR A DOS/VS VIRTUAL MACHINE

You can prepare a job stream for a DOS/VS virtual machine and place it in the DOS/VS virtual machine's card reader in one of two ways:

- Prepare a real deck of punched cards containing DOS/VS job control statements, input files, and so on. Place a CP ID card in front of this deck, indicating the userid of the DOS/VS virtual machine, for example

  ```
  ID DOSVUSER
  ```

  Put the cards in the real system card reader. Based on the userid specified on the ID card, CP directs the spool file to the virtual card reader of the DOS/VS virtual machine, which in this case is being run on the userid DOSVUSER.

- Use CMS to create a disk file containing card images, identical to the real cards you submit in a real card reader for a DOS/VS system. Use the CP SPOOL command to spool your virtual card punch to the card

reader of the DOS/VS virtual machine and use the CMS PUNCH command to punch the card images. In the CMS environment you would issue:

```
spool punch to dosvuser
punch dosjob jcl (noheader
```

You must use the NOHEADER option of the PUNCH command to suppress the punching of a CMS READ control card at the front of the card deck.

A job stream spooled to the DOS/VS system by either of the above methods remains in the card reader of the DOS/VS virtual machine until you cause the DOS/VS system to begin reading the job stream from its card reader.

You can spool the card file before or after you IPL the DOS/VS system, or any time while the system is active.

Whenever you are preparing to punch jobs to a virtual machine using your virtual punch, you should take the precaution of clearing any files or card images that may remain in it from previous jobs. The following commands ensure that the virtual punch does not have any cards in it:

```
spool punch nocont
close punch purge
```

Likewise you should clear the card reader of the virtual machine that is going to be running the DOS/VS system:

```
spool reader nocont
close reader
purge reader
```

These commands purge any existing reader files. You should take care, of course, that any files that may be in your reader are not files that you need. To obtain data about existing reader files before purging the reader, enter the command:

```
query reader all
```

Note, however, that no indication or information is available for any open files.

IPL THE DOS/VS SYSTEM

There are a number of methods that you can use to load the DOS/VS system in your virtual machine. The technique below shows how to enter the commands and control statements necessary to IPL and ready the DOS/VS system for input jobs. If is followed by an example of how to IPL the DOS system from your card reader.

IPL FROM THE CONSOLE

When you are sure that you have all of the virtual unit record devices you need, your console is properly defined, and all the DASD devices you need are attached, you should enter the command:

```
set ecmode on
```

to make extended control (EC) mode active for your virtual machine. If this option is specified in your directory entry, you do not need to enter it. Since the DOS/VS system performs paging, you must operate in this mode.

To load the DOS/VS system into your virtual machine, enter the IPL command:

```
ipl 250
```

After a few seconds, the system enters a wait state. On a real system console, the wait light goes on. On your terminal, you may want to let a few seconds pass to be sure that the wait state has been entered. To verify that the system is in a wait state, enter CP mode and use the DISPLAY command to display the PSW:

```
display psw
```

If bit 14 is a one, the system is in a wait state. If not, use the BEGIN command to resume the program execution.

When you have determined that the system is in wait state, you must cause an attention interrupt by pressing the Attention key (or equivalent). The message

```
0I03A  SPECIFY SUPERVISOR NAME
```

prompts you to enter the name of the DOS/VS supervisor. If you enter a null line, the default supervisor, $$A$SUP1, is used.

Shortly after you enter the supervisor name, the system enters the wait state again. You should allow a few seconds to be sure that the wait state has been reached, and cause another attention interrupt with the Attention key.

Next, the DOS/VS system displays a series of messages that display the status of the IPL procedure, followed by a prompting message

```
0I10A  GIVE IPL CONTROL COMMANDS
```

And you can enter:

- ADD or DEL commands (optional), as necessary, to alter the default configuration of the DOS/VS system, established at system generation.
- The command SET, to intialize the date and time clock. This command is required.
- The command CAT (optional), to define the VSAM master catalog.
- The DPD command, to define the page data set. This command is required.

After you enter the DPD command, the message

```
0I20I DOS/VS IPL COMPLETE
```

signifies that the DOS/VS system is loaded into your virtual machine.

If there is a warm start copy of the SVA (shared virtual area) available, you also receive the message

```
1T00A WARM START COPY OF SVA FOUND
```

and you must enter KEEP (or a null line) or REJ, depending on whether you want to use this copy of the SVA or not.

If there is no warm start copy of the SVA, this message does not appear. If you need to use the SVA, you can create one using a standard procedure, depending on what is available in your DOS/VS system.

When the IPL procedure is complete, the message

BG 1I00A READY FOR COMMUNICATIONS.

indicates that the background partition is running and ready to accept control commands or job control statements.

The complete VM/370 logon and DOS/VS IPL procedure as it would appear on a 2741 terminal is shown in Figure 6. The ! (exclamation marks) indicate pressing the Attention key or equivalent.

```
logon tester
ENTER PASSWORD:

FILES:  001 RDR,  NO PRT,  NO PUN        Note 1
LOGON AT 08:19:52 EST MONDAY 01/12/76
set ecmode on
link dosys 250 250 w        Note 2
DASD 250 LINKED R/W
ipl 250        Note 3
!
0I03A SPECIFY SUPERVISOR NAME
$$a$sup2
!
0I04I IPLDEV=X'250',VOLSER=DOSRE3,CPUID=FF0101530155
0I30I DATE=10/30/75,CLOCK=16/35/09,ZONE=EAST/04/00
0I10A GIVE IPL CONTROL COMMANDS
set        Note 4
dpd
0I52I PAGE DATA SET EXTENT       LOW    HIGH
                               327   0 250  11
0I20I DOS/VS IPL COMPLETE
BG
1I00A  READY FOR COMMUNICATIONS.
BG        Note 5
log
BG
assgn sysrec,x'250'
BG
set rf=yes
BG
      Note 6
BG
// JOB TEST1

DATE 10/30/75,CLOCK 16/35/47
BG
1I89A  IPL REASON CODE =        Note 7

BG
1I91A  SUB-SYSTEM ID =

BG
1I93I  RECORDER FILE IS   2% FULL
BG
// OPTION NODECK        Note 8
BG
// EXEC ASSEMBLY
BG
EOJ TEST1

DATE 10/30/75,CLOCK 16/37/49,DURATION 00/02/01
BG
1C00A  ATTN. 00C        Note 9
BG
      Note 10
BG
0P08A    INTERV REQ SYSRDR=00C
```

Figure 6. IPL DOS/VS and Execute a Job in the Card Reader

Notes

1. The reader file contains a job stream for the DOS/VS virtual machine.

2. The DOS/VS system residence is assumed to be defined at virtual address 250 in userid DOSYS's directory entry, with a write password of ALL.

3. After the IPL command, the Attention interrupt causes message 0I03A. After entering the supervisor name, a later interrupt continues the IPL procedure.

4. The SET and DPD commands are required for IPLing the DOS/VS system.

5. The background partition is active and waiting for commands. The LOG command is optional; the ASSGN and SET commands shown here may be required for system recording.

6. A null line signals an interrupt to your card reader, so that the DOS/VS system begins the job stream.

7. These are RDE messages; a null line entered in response indicates that you are taking the default values.

8. DOS/VS continues reading the job.

9. The card reader is empty; the spool file has been read.

10. A null line causes another interrupt to the card reader; if there is another file in the card reader, it is read. Otherwise, message 0P08A is issued.

IPL FROM THE CARD READER

You can place the control commands you need to perform the IPL procedure in the card reader of the DOS/VS virtual machine before you issue the IPL command to load the system, and then signal the DOS/VS system to read the control commands from the card reader. This method can be more efficient than entering all of the control commands manually, if there are many ADD and DEL commands that you must issue when you IPL DOS/VS.

The cards necessary to perform the IPL procedure must be in two separate card files: (1) a single card specifying the supervisor name, and (2) a card deck containing the IPL commands. You may follow these card files with additional files containing the jobs you want to execute in your DOS/VS virtual machine.

As in the IPL procedure outlined under "IPL the DOS/VS System," shortly after you enter the IPL command

```
ipl 250
```

the system enters a wait state. If you are IPLing DOS/VS from your card reader, you must cause a reader interrupt so that the system reads the name of the supervisor from the card reader. To cause a reader interrupt, you must enter the CP environment by pressing the Attention key twice (or the 3270's PA1 key once) to cause a CP read, and enter

```
#cp ready 00c
#cp begin
```

The READY command causes a reader interrupt; the BEGIN command returns control to DOS/VS; and, the first spool file is read from the card reader.

Shortly after this card is read, and the system enters the wait state a second time, you must again enter the CP environment and then enter:

```
ready 00c
begin
```

so that the DOS/VS system begins reading the IPL commands.

When you IPL the DOS/VS system through the card reader, you cannot supply the responses necessary to save the warm start copy of the SVA from card input. You must supply these responses from the console. If you want to create the SVA and SDL when you IPL, however, you may place the cards necessary to perform these procedures in the card reader in a separate spool file following the IPL commands. Again, you must use the READY 00C command to cause the reader interrupt to force the system to begin reading from the card reader.

BEGIN THE DOS/VS SYSTEM OPERATION

Depending on how the DOS/VS system you are using was generated, there are additional operator commands and control statements you must enter at the console before you can begin running jobs on the DOS/VS virtual machine.

If there is an active system recorder file, it is opened when the first // JOB card is encountered in the input stream. You should enter the ASSGN and SET commands that define the SYSREC device and the status of the system recorder file before this JOB card is read; otherwise, an

error is encountered opening the SYSREC file. For example, if the system recorder file is already active on the system residence volume, you enter

```
assgn sysrec,x'250'
set rf=yes
```

If you are starting a DOS/VS virtual machine to run in batch mode to process jobs from other users, you may also want to enter the operator commands necessary to:

- Allocate storage among different partitions
- Start foreground partitions in operation
- Initialize POWER/VS

These considerations are discussed under "Using DOS/VS In Batch Mode Under VM/370." If you are alternating between the CMS and DOS/VS systems, you may want to keep the IPL procedure as simple as possible.

Starting a Job Stream

If you have prepared job streams and placed them in the card reader of the DOS/VS system, then after you receive the READY FOR COMMUNICATIONS message, you can enter a null line (with the Return or Enter key) to cause an interrupt. This interrupt causes DOS/VS to begin reading from the card reader.

If your system was generated with RDE (reliability data extractor), you must respond to the following messages after the first // JOB card is read:

```
1I89A  IPL REASON CODE =
1I91A  SUB-SYSTEM ID =
```

See DOS/VS Messages for information on how to respond to these messages.

If you entered the LOG command before beginning the job stream, all the job control statements that are executed are displayed on your terminal.

As the operator of the DOS/VS system, you may enter control statements directly from the terminal. For example, if you want to execute cataloged programs or procedures, you can enter the ASSGN, DLBL, and EXEC statements necessary to execute the program directly from the console.

When a Job is Finished

When the DOS/VS virtual machine running a single partition finishes reading a spool file (which may contain one or more jobs), it posts the attention message:

```
1C00A  ATTN. 00C
```

which indicates that the reader is empty. If there are additional jobs in the card reader that were spooled as separate CP spool files, you

should enter a null line, again, to cause DOS/VS to begin reading from the card reader. If there are no more files in the reader and you enter a null line, the message

```
0P08A    INTERV REQ SYSRDR=00C
```

indicates that operator intervention is required. You can put cards into the real system card reader to direct another job stream to the DOS/VS virtual machine, if you want. When additional cards are received in the reader of the DOS/VS virtual machine, reading automatically begins.

If the card reader of the DOS/VS virtual machine is spooled with the CONT option, then any jobs received in the card reader while a job is executing are read upon completion of the current job stream. If a job stream completes, however, and the end of the spool file is reached before another job is received, an interrupt must be issued to cause the next job to be read.

If you want to terminate the DOS/VS terminal session you can use the Attention key to enter the CP command environment, where you can enter either the LOGOFF command to log off the VM/370 system, or you can use the IPL command to load another operating system into your virtual machine.

COMMUNICATING WITH CP

During the operation of the DOS/VS virtual machine, you can use CP commands to communicate with the system operator or other virtual machine users, query the status of virtual machine devices or spool files, or attach or detach devices from your virtual machine configuration.

To enter a CP command while your DOS/VS virtual machine is in operation, you can press the Attention key twice, quickly, to force a CP read (on a 3270, use the PA1 key). After entering CP commands, you can use the command BEGIN to resume the DOS/VS virtual machine execution. For example, if a PAUSE control statement is encountered that indicates that the VM/370 operator be requested to perform some action, you must enter the CP environment to send a message to the operator of the VM/370 system. For example, the following lines represent a typical sequence on a typewriter terminal:

```
// PAUSE ASK OPERATOR TO ATTACH TAPE AS 284
!!
CP
msg op please attach scratch tape as 284
sleep
TAPE 284 ATTACHED
!
CP
begin
```

The SLEEP command locks the keyboard, and places the virtual machine in a dormant status to wait for any messages. When you receive the message indicating the tape is attached, you press the Attention key once to return to CP, and then use the BEGIN command to resume virtual machine execution. The // PAUSE message is still outstanding so you should press the RETURN key to continue.

Using the #CP Function

In most cases during your virtual machine operation you can use the #CP function to enter CP commands directly from the virtual machine environment. You do not always have to use the Attention key to enter the CP environment. If your virtual machine is waiting for a read, you can enter a CP command with the #CP function, for example

```
#cp msg op please mount vol240
```

The command line is processed by CP, and the virtual machine is still waiting for a read. You may find this method convenient for entering CP commands.

Note: You cannot always enter CP commands with the #CP function, however. During the IPL procedure when the DOS/VS system is processing IPL commands, for example, the CCWs that are used for these reads expect only 3 bytes of data, so additional information on CP command lines is truncated.

Interrupting the Virtual Machine

While the DOS/VS system is running in your virtual machine, you can interrupt its execution using the Attention key (or equivalent) on your terminal. The DOS/VS attention handler responds with

```
AR
1I60A  READY FOR COMMUNICATIONS.
AR
```

and you can enter attention commands. To resume program execution, enter a null line. You should normally wait until the attention has been processed before you signal another one except when cancelling a dump.

If you are using a 2741 terminal, and you want to use the Attention key mainly to signal control program (CP) interrupts, enter the command:

```
#cp terminal mode cp
```

The first time you press the Attention key, a CP interrupt will be posted; the next depression of the Attention key will signal an interrupt to DOS/VS.

If you are in the CP environment and wish to signal an interrupt to DOS/VS, you can enter either the ATTN or REQUEST commands.

RUNNING A BATCH DOS/VS SYSTEM UNDER VM/370

If you are using DOS/VS in a virtual machine as production tool, it is likely that the virtual machine that is running DOS/VS is going to be continously logged on. This machine may be available for many users to submit jobs, or it may be used only by installation personnel responsible for running the production jobs.

In either event, it is likely that the DOS/VS system that you are using has been generated specifically for use under VM/370. In this case, you should know whether:

- It is necessary to start more than one partition, and if it is, how much virtual storage to allocate to each partition. These considerations are much the same as they would be for a user of a native DOS/VS system, who must decide how much work is going to be done and the most efficient way to do it.

- To generate POWER/VS. This program can provide many functions not available when you rely solely on CP spooling, including the ability to spool jobs via class, purge selected jobs in a job stream, and so on.

  When POWER/VS is active in your DOS/VS virtual machine, POWER controls which jobs are done in the various partitions, and reads jobs from the card reader. Since POWER is always working, there is no need to cause an attention interrupt when a new job is placed in a card reader.

  To start POWER/VS in a virtual machine, you can use the AUTOSTART procedure, and can spool the card deck necessary for the automatic start either using real cards or through the virtual card reader.

- Virtual devices are dedicated for use by the DOS/VS virtual machine, or whether devices must be shared among other virtual machines.

  For example, if a card reader has been dedicated to a DOS/VS virtual machine, then users may submit jobs through that card reader without having to preface the card with a CP ID card. If the DOS/VS virtual machine is sharing the system card reader, then any user submitting a card deck must place an ID card in front of the deck, specifying the userid of the DOS/VS virtual machine.

ALTERNATING OPERATING SYSTEMS

If you are working in a development and testing environment, rather than in a batch environment, and the work is not suitable for CMS/DOS, there are advantages to alternating the CMS and DOS/VS operating systems, among them:

- Reduced unit record output. You can examine program output and compiler listings online, check the results and resubmit the job without printing a single page on the system printer or punching card decks.

- Faster turn-around time compared to batch alone; you can see the results of program compilation and execution immediately, rather than waiting for output from a batch system.

The technique for alternating operating systems is outlined below. If you want to use this technique, you should be familiar with CMS, the conversational component of VM/370, particularly with the CMS Editor and some file system commands. Usage information about CMS is in VM/370: CMS User's Guide. For details on CMS commands and EDIT subcommands, see VM/370: CMS Command and Macro Reference.

Load CMS Into Your Virtual Machine

To load the CMS system into your virtual machine, use the IPL command. Usually, you can IPL CMS specifying the saved system name, CMS:

```
ipl cms
```

Or, you can load CMS specifying the virtual address of the CMS system, which is usually 190:

```
ipl 190
```

After you receive a message like

```
CMS VERSION 3.0
```

you can use the interactive facilities of CMS to prepare jobs to execute in your DOS/VS virtual machine.

Use the CMS Editor To Prepare Jobs

You can use the EDIT command in CMS to invoke the CMS Editor, with which you can prepare disk files of 80-character card image records. The files that you create may contain DOS/VS job control statements, source files, or even IPL decks.

For example, if you want to IPL the DOS/VS system using input statements from your card reader, you could prepare CMS files that contain:

- The DOS/VS supervisor name (required for IPL). This file might contain the record:

```
$$A$SUP1
```

- The IPL control statements that you want to supply. This file might contain:

```
set
dpd
assgn sysrec,x'250'
set rf=yes
log
```

Let's assume, for the purposes of this example that these files have CMS file identifications of SUPER JCL and START JCL. All of the statements you must enter to control the IPL of the DOS/VS system are in these files.

You could also use the Editor to prepare the job stream you want to execute. Suppose you want to assemble an Assembler Language source program. If you have the source file available as a CMS disk file, you can edit the file and insert the DOS/VS job control statements into the CMS file. The file DOSTEST JCL might contain:

```
// JOB TEST1
// OPTION NODECK
// EXEC ASSEMBLY
   (Assembler language source statements)
      .
      .
      .
/*
/&
```

Although this job stream contains only the job control statements necessary to assemble the job, you can also include the statements

necessary to link-edit the output module, catalog the program, and so on.

Issue SPOOL Commands to Control Unit Record Devices

Before sending your job to the DOS/VS virtual machine, you should check your unit record devices:

- Close and purge your card reader, to make sure there are no files left over from a previous job:

```
spool reader nocont
close reader
purge reader all
```

- Close and purge your card punch, and spool it to your card reader:

```
spool punch nocont
close punch purge
spool punch to *
```

  If you are going to use your card reader to perform the IPL of the DOS/VS system, you must be sure your punch is spooled NOCONT, since at least the first two files must be separate spool files.

- Spool your virtual printer to your card reader:

```
spool printer to *
```

  If you want DOS/VS SYSLST output to print on the system printer, you can omit this SPOOL command. You can also spool the printer file a different class, for example, L, to ensure that in the case of the printer being closed before you are ready to IPL CMS, the DOS/VS system does not read the printer file from your reader:

```
spool printer to * class l
```

Punch the CMS Files

Use the CMS PUNCH command to punch the card files. The files are spooled to your virtual card reader. For example, to punch the three files used above, enter

```
punch super jcl (noheader
punch start jcl (noheader
punch dostest jcl (noheader
```

You must use the NOHEADER option to suppress the punching of a CMS READ control card, which is punched by default when you use the PUNCH command.

IPL the DOS/VS System

Now, if you have the DOS/VS system residence attached to your virtual machine, you can use the IPL command to load the DOS/VS system:

```
ipl 250
```

You can wait a few moments, and use the DISPLAY command to see if the system is in a wait state. When you have determined that the system is in a wait state, enter an attention interrupt:

```
!!
CP
display psw
PSW = 030E000 00000000
ready 00c
begin
```

After this BEGIN command returns control to the virtual machine, the file SUPER JCL is read from the card reader, and the IPL procedure continues. You must again wait for the system to enter a wait state and ready the reader:

```
!!
CP
display psw
PSW = 030E0000 00012800
ready 00c
begin
```

When the IPL procedure is complete, messages like

```
0I52I PAGE DATA SET EXTENT      LOW    HIGH
                              327   0 250  11
0I20I DOS/VS IPL COMPLETE
BG
1C00A  ATTN. 00C
BG
```

indicate the completion of the IPL procedure. As a practical matter, your installation may want to create a saved DOS/VS system at this point so that future loadings of DOS/VS would avoid these prelimanary steps. Refer to the VM/370: System Programmer's Guide for information on creating saved systems.

Signal DOS/VS To Begin Reading the Job Stream

When the DOS/VS system has been loaded and the message BG indicates that it is waiting for communication, you can enter a null line so that the DOS/VS system begins reading the next spool reader file, which is the one that contains the job control statements, and in this example, the assembler language source program.

If you issued the LOG command during the IPl procedure, all of the job control statements that are read are displayed on your terminal as they are executed.

If the RDE option is in effect for this DOS/VS system, you must respond to the messages:

```
1I89A  IPL REASON CODE =
1I91A  SUB-SYSTEM ID =
```

before the job is executed.

When the Card Reader is Empty

When the job that you punched to the card reader has been read, the message

```
EOJ TEST1
```

is followed by a time stamp, such as

```
DATE 02/19/76,CLOCK 20/19/28 DURATION 00/07/48
```

and additional messages from the background partition:

```
BG
1C00A ATTN. 00C
BG
```

If you punched more than one job, and the spool file(s) for remaining jobs are still in the card reader, you can enter another null line to signal the DOS/VS system to read from the reader.

Reload CMS Into Your Virtual Machine

When your job or jobs are finished in the DOS/VS virtual machine, you can return to the CMS environment by entering CP mode and issuing the command

```
ipl cms
```

This IPL command closes your virtual printer, so if you have spooled your printer to your card reader, or have your printer in a hold status, you receive a message from CP, such as

```
PRT FILE 4786 TO BILBO    COPY 01  NOHOLD
```

or

```
PRT FILE 4786 FOR BILBO    COPY 01 HOLD
```

This printer file contains the SYSLST output from the job that you executed in DOS/VS.

If the file is spooled to the printer, it is queued for printing on the system printer. If the printer is spooled to your card reader, it is a reader file, available to you in your card reader.

Examine the Output From the DOS/VS Virtual Machine

If you want to examine the output from the job you executed under DOS/VS in your virtual machine, and you spooled your printer to your card reader, you can now read the reader file onto your CMS A-disk with the READCARD command, for example

```
readcard dostest output
```

This command creates a file named DOSTEST OUTPUT, which contains the spooled printer output. Now, using the CMS Editor you can examine the contents of the file, or you can use the TYPE command to display the whole file at your terminal.

You can invoke the editor with the EDIT command

```
edit dostest output
```

If you are examining the output of an assembly or compilation, you can use a LOCATE subcommand to locate a keyword in the listing file, for example, the subcommand

```
locate /diagnostics
```

locates the beginning of the diagnostic messages produced by the assembler.

If there are no errors in the assembly, you can modify the job control statements in the file and resubmit the job. Otherwise, correct the source statements and then repeat the procedure for clearing files from the card punch and card reader, punch the IPL decks and the job stream, and re-IPL the DOS/VS system.

USING EXEC PROCEDURES

If you are alternating CMS and DOS/VS in your virtual machine extensively, you can place the commands necessary to spool unit record devices, punch the IPL deck, and a job stream, into an EXEC procedure. Then, to send a job to your card reader and IPL the DOS/VS system, you only have to enter the EXEC name, and in some cases, a few additional control commands.

For example, the command sequence used above could be contained in an EXEC procedure such as:

```
CP SPOOL PUNCH NOCONT
CP CLOSE PUNCH PURGE
CP CLOSE C
CP PURGE READER ALL
PUNCH SUPER JCL (NOH
PUNCH START JCL (NOH
PUNCH DOSTEST JCL (NOH
CP IPL 250
```

If this EXEC procedure is named DOSJOB, then to execute this procedure, all you have to do is enter:

```
dosjob
```

Once the IPL command in the EXEC is performed, your virtual machine is no longer under the control of CMS, so you must enter attention interrupts and CP commands, as necessary, to begin DOS/VS system operation.

You can make an EXEC procedure more generalized, also. For example, if you want to make this EXEC capable of sending any job to your card reader, you could use the EXEC symbol &1 in place of the DOSTEST filename, for example:

```
CP SPOOL PUNCH NOCONT
CP CLOSE PUNCH PURGE
CP CLOSE C
CP PURGE READER ALL
PUNCH SUPER JCL (NOH
PUNCH START JCL (NOH
PUNCH &1 JCL (NOH
CP IPL 250
```

With this EXEC, you can send any job to your card reader that has a CMS filetype of JCL, for example, if you enter

```
dosjob prog3
```

The EXEC variable symbol &1 is replaced with the argument PROG3 and the file PROG3 JCL is punched to your card reader for execution under DOS/VS.

As an additional aid, when you are entering the commands and responses necessary to IPL the DOS/VS system from the card reader, you can use the CONT option of the SPOOL command to spool the IPL commands and your job stream as a single spool file, for example

```
CP SPOOL PUNCH NOCONT
CP CLOSE PUNCH PURGE
CP CLOSE C
CP PURGE READER ALL
PUNCH SUPER JCL (NOH
CP SPOOL D CONT
PUNCH START JCL (NOH
PUNCH DOSTEST JCL (NOH
CP SPOOL D NOCONT
CP IPL 250
```

When you spool the punch with the CONT option, the files START JCL and DOSTEST JCL are spooled as a single spool file. When the IPL commands are read, the DOS/VS system continues reading from the card reader, so that you do not have to signal (with a null line) when you want DOS/VS to being reading the job stream.

This method of spooling has an additional advantage in the IPL procedure. When you issue the IPL command, the virtual punch is closed. In this instance, since the punch was spooled NOCONT, but not closed, the IPL closes the punch and as the file is spooled to your reader, this causes a reader interrupt, which is effectively the first interrupt you have to enter (the one that causes DOS/VS to read the supervisor name).

In effect, when you execute this format of the EXEC, you are required, after the IPL command is executed, to enter only one interrupt and one READY command to execute the entire job. Your console sheet might look like this:

```
dosjob
!!
display psw
PSW = 030E0000 000128000
ready 00c
begin
```

After you issue this BEGIN command, the remainder of the job is executed without your intervention, until the card reader is empty.

The CMS EXEC Facility is described in detail in the VM/370: CMS User's Guide.

USING MORE THAN ONE VIRTUAL MACHINE

If you are running a job in a DOS/VS virtual machine, that may take a long time to execute, you may want to free your terminal for other work. In these situations, you can use the DISCONN command to disconnect the DOS/VS virtual machine and log on to some other userid.

A sample procedure, using the userids CMSPREP1 and DOSTEST1, is shown below, followed by a list of some additional things you must consider when you are disconnecting your terminal. Note that while it is not necessary to use CMS to prepare or transmit the jobs to the DOS/VS virtual machine, you may find it convenient.

Log On to a Userid to Prepare the Job in CMS

If you have available a userid with which you can access the CMS system, you can log on to that userid and load CMS:

```
logon cmsprep1
ENTER PASSWORD:

LOGON AT 10:30:41 EST TUESDAY 02/28/76
ipl cms
CMS VERSION 3.0
```

Assuming that you have a read/write CMS A-disk, you can create CMS files that contain the IPL control commands and responses, as well as the job streams to execute in DOS/VS. Then, using the SPOOL and PUNCH commands, you can place copies of these files in the card reader of the machine that is going to be running the DOS/VS system.

Suppose you have files named SUPER JCL and START JCL, which contain the supervisor name and IPL commands for the IPL procedure. In addition, the file DOSTEST JCL contains the job stream that you want to execute. To spool these files to the userid DOSTEST1 you can enter:

```
cp spool punch to dostest1
punch super jcl (noheader
punch start jcl (noheader
punch dostest jcl (noheader
```

Disconnect the CMS Virtual Machine

When you are ready to log on to the userid that is going to run the DOS/VS system, you can disconnect the CMS virtual machine as follows:

```
disconn hold
```

You should use the HOLD option of the DISCONN command if you are using a dial-up terminal and you do not want to lose the connection.

Since the CMS machine is not currently active, you do not need to disconnect it. You could just as easily log off, but if you disconnect then when you log on again, you do not have to reload CMS, respool your punch, printer, and so on.

Log On to the DOS/VS Virtual Machine

When you log on to the virtual machine that is going to run the DOS/VS system, you see, in addition to the normal log messages, a message indicating that there are files in the card reader:

```
logon dostest1
   .
FILES:  003 RDR,  NO PRT,  NO PUN
LOGON AT 10:50:34 EST TUESDAY 02/28/76
```

If this virtual machine does not have the ECMODE option set on you must issue the command SET ECMODE ON. Then, you can begin the IPL procedure as outlined under "IPL from the Card Reader."

If the RDE option is in effect for the DOS/VS system, you should wait until after you have responded to the RDE messages before you disconnect the DOS/VS virtual machine. Then, you can enter CP mode, using the Attention key, and enter the DISCONN command:

```
!!
CP
disconn
```

The DOS/VS virtual machine continues to run.

Return to the CMS Virtual Machine

You can now reconnect onto the CMS virtual machine, using the VM/370 logon procedures. When you log on, you receive the message

```
RECONNECT AT 11:05:02 EST TUESDAY 02/28/76
```

instead of the normal LOGON message. To return to the CMS environment and continue working, you can enter the command

```
begin
```

If you issue the command

```
query dostest1
```

you can see that the virtual machine that you set up to run the DOS/VS system is running, and you can continue to use CMS (or some other operating system) in this virtual machine.

If you wish, you can again disconnect the CMS virtual machine and reconnect onto the DOS/VS virtual machine, if, for example, you know that the job stream that is executing may pause to require an operator response. Or, if you use CMS to spool another job to the DOS/VS system, you may need to reconnect to the DOS/VS virtual machine in order to alert DOS/VS to begin reading from the card reader.

After reconnecting, you must issue the command

```
begin
```

before the DOS/VS virtual machine will resume execution.

CONSIDERATIONS WHEN DISCONNECTING A DOS/VS VIRTUAL MACHINE

If you are using more than one userid to alternate between two operating systems, you should keep in mind (1) how the DOS/VS system may read additional jobs from the card reader (2) what happens when a read occurs in the DOS/VS virtual machine and (3) what happens to the console log of a disconnected virtual machine.

Sending Jobs to a Disconnected DOS/VS Machine

When you spool a job or jobs to the card reader of a DOS/VS virtual machine (or to the userid for a machine that is going to run DOS/VS), you can spool your card punch with the CONT option before you punch the individual jobs so that the DOS/VS machine will, after completing one job, continue reading the next. If the jobs have been spooled separately, that is, they appear in the card reader of the DOS/VS machine as separate spool files, you must, at the completion of each job, enter a null line so that the DOS/VS virtual machine reads the next job. If you are running DOS/VS disconnected, then you must reconnect to the DOS/VS virtual machine so that you can enter the interrupt to cause it to continue reading.

If you are planning to use CMS to spool jobs to a disconnected DOS/VS virtual machine, you can spool the reader of the DOS/VS machine with the CONT option:

```
spool reader cont
```

Then, if another job is received in the card reader of the DOS/VS machine before the currently executing job stream has completed, the job just received is also read and executed. Thus, if you are running a series of jobs in a single job stream and the reader is spooled with the CONT option, you can continue to send (from the CMS virtual machine) additional jobs to the DOS/VS virtual machine for execution.

You do not, in these circumstances, have to reconnect to the DOS/VS virtual machine to signal an interrupt, unless the currently executing job stream finishes before the next job is received in the card reader.

If the DOS/VS system that is running disconnected is processing many jobs, you may want to obtain printed output for jobs that are completed without waiting for all the jobs to finish. To release spooled printer output accumulated directly or through POWER/VS, you should reconnect the DOS/VS system and issue the command:

```
close printer
```

This releases the SYSLST output accumulated thus far. However, if you are using POWER/VS and a dedicated printer, all spooled output files are under the control of POWER spooling, and not CP spooling.

When a Read Occurs in a Disconnected DOS/VS Virtual Machine

If a virtual machine is running disconnected and a console read occurs, a 15-minute timeout occurs. If there is no response in the virtual machine before the 15 minutes elapse, the virtual machine is automatically logged off.

So, for example, if you are running a DOS/VS system disconnected and a program running in the machine completes execution or issues a PAUSE request, the virtual machine is logged off after 15 minutes, unless you reconnect the virtual machine and issue the BEGIN command.

The Console Log of a Disconnected DOS/VS Machine

When a virtual machine is running disconnected, all output or "writes" to the virtual console are ignored unless you spool your console. Before you disconnect your virtual machine, issue the CP command:

```
#cp spool console start
```

to start recording all console input and output on spool file. When you log back on, issue the command:

```
spool console stop close
```

to stop console spooling and release the spool file to the real printer.

HOW CMS CAN HELP YOU DEVELOP AND TEST PROGRAMS TO RUN IN A DOS/VS VIRTUAL MACHINE

The discussions above noted how you can use the CMS Editor and the EXEC Facility to help you prepare jobs for execution in a DOS/VS virtual machine. In addition to these CMS features, there are a number of other CMS commands that you can use that may help you in developing and testing programs.

You can copy DOS sequential disk, tape, or card files into CMS disk files using the MOVEFILE command. Then, you can use the CMS Editor to modify the file, if necessary, or use it as part of the input job stream for a DOS/VS system.

If the file that you want to write into a CMS disk file is contained on cards, you can also use the READCARD command to read the file onto disk.

One of the advantages of storing your source programs on CMS disks is that you can maintain a backup copy of a program while you test and debug a second version. Using the editor, or commands like COPYFILE and RENAME, you can modify and copy CMS disk files.

USE CMS/DOS TO DEVELOP AND TEST YOUR PROGRAMS

CMS has a special environment, called CMS/DOS, which provides many commands that simulate the functions of DOS/VS. These commands are available to you in the CMS environment after you enter the command

```
set dos on
```

You can access the DOS/VS system residence from CMS/DOS. If you want to use it, you must access it in CMS, and specifying the mode letter of the disk on the SET DOS ON command line:

```
access 250 z
set dos on z
```

Some of the things you can do in CMS/DOS are:

- Create CMS macro libraries from DOS/VS macro libraries with the SSERV, ESERV, and MACLIB commands. Then use the ASSEMBLE command to assemble programs directly in CMS. Assembler diagnostic messages are displayed on the terminal.

- Compile programs written in DOS/VS COBOL or DOS/VS PL/I programming languages, using DOS/VS macro libraries.

- Display or print the directories of DOS/VS private or system core image, relocatable, source statement, or procedure libraries with the DSERV command.

- Display or print the procedure library with the PSERV command.

- Link-edit TEXT decks from CMS disks, or relocatable modules from DOS/VS system or private relocatable libraries, using the DOSLKED command. This command creates simulated core image libraries, called DOSLIBs, on a CMS disk. You can also use the RSERV command to copy relocatable modules from DOS/VS libraries.

- Load core image phases from CMS DOSLIBs or from DOS/VS core image libraries into virtual storage and execute them, using the FETCH and START commands.

- Identify system and programmer logical units for programs that you use with the ASSGN command, and list current assignments with the LISTIO command.

- Use the DLBL command to identify disk files. When you execute programs in CMS/DOS, you can read sequential disk files directly from DOS disks, but you cannot write them. Instead, for testing purposes, you can write CMS disk files, or write output files to your virtual punch, printer, or to a tape.

  You can, however, both read and write VSAM files that are located on DOS disks directly from CMS, when you execute COBOL and PL/I programs in CMS/DOS.

- Use the CMS and VM/370 debugging facilities to debug your program in CMS. You can set address stops (called breakpoints in the CMS debug environment), and temporarily halt the execution of a program to examine or change the contents of registers or specific storage locations.

When your program is tested and debugged in CMS/DOS you can prepare a job stream to catalog and execute the program in a DOS/VS virtual machine.

For complete details on how to use CMS/DOS, see VM/370: CMS User's Guide. For details on the command formats of CMS commands, see VM/370: CMS Command and Macro Reference.

Section 5. OS/VS in a Virtual Machine

Section 5 contains operational information specific to OS/VS and OS.

The following topics are included:

Note: For the purposes of the ensuing discussion, the term "OS/VS" shall be considered as a generic expression representing any or all of OS, OS/VS1, and OS/VS2 unless specified otherwise.

This discussion of OS/VS running under VM/370 is intended for the OS operator. The systems programmer is assumed to have generated an OS/VS system employing the various considerations and techniques discussed elsewhere in this document. When you log on to VM/370 with a userid established at your installation and load an OS/VS system, your terminal becomes the operator's console and you are responsible for operating the OS/VS system.

You should be equipped with the complement of documentation supporting the particular system that you are operating. For the OS/VS1 operator, the following documentation is recommended:

Operator's Library: OS/VS1 Reference, GC38-0110
OS/VS Message Library: VS1 System Messages, GC38-1001
OS/VS Message Library: VS1 System Codes, GC38-1003

If you are operating VS2, you may wish to have at least the following manuals on hand:

Operator's Library: OS/VS2 Reference (JES2), GC28-0210
OS/VS Message Library: VS2 System Messages, GS38-1002
OS/VS Message Library: VS2 System Codes, GC38-1008

The basic techniques of running an OS/VS machine in a VM/370 environment are depicted in Figure 7. The three most common techniques for running OS/VS on VM/370 are:

- "Batch mode." One user runs as the OS/VS machine (userid OSVS) and other users (like CMSID1) may submit jobs either through the virtual card reader, through the system card reader, or via JES remote stations.

- The "Flip-flop technique." The IPL command is used to alternate between OS/VS and CMS in a single virtual machine. This method requires only one directory entry, the one for the OS/VS user, but because of the lengthy IPL process, it is only practical if your installation has created a saved OS/VS system.

- The disconnected user. This technique is more appropriate for the OS/VS2 user since VS2 allows you to recall a prior system message. You can log on as OS/VS operator under the OSVS userid, start up your system, and then disconnect. While the OS/VS machine continues to run, you can log on as CMS user (CMSID1) and create and submit jobstreams and check the resulting output. If you want to check the progress of the operating system, you can disconnect from your CMS machine and reconnect to the OSVS virtual machine via the LOGON command. These techniques are discussed in greater detail below. First, you must understand how to access the OS/VS system residence volume, ensure that the proper devices are attached to your virtual machine, and IPL OS/VS using VM/370.

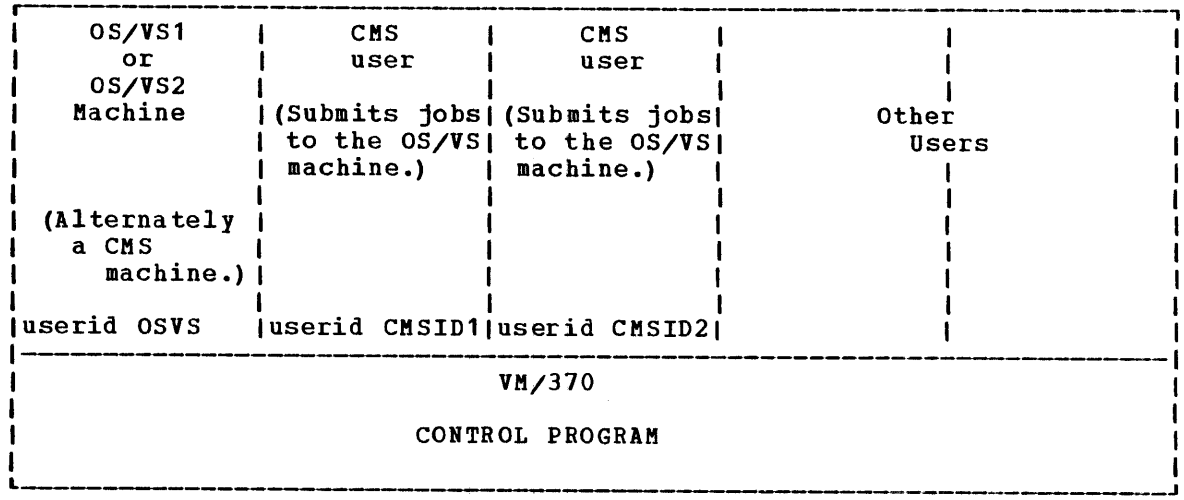


Figure 7. Running OS/VS under VM/370

THE SYSTEM RESIDENCE VOLUME

One of the things you need to know, as the OS/VS operator, is the location of the system residence (sysres) pack. Its location can be defined in your virtual machine configuration in one of three ways:

1. Your system residence volume can be defined as a read/write disk in the directory entry for your OSVS userid. Such a definition may appear as follows:

   ```
   MDISK 250 3330 0 403 VS2RES WR RALPH FARMER

        -- or --

   LINK VS2SYSP 150 250 W
   ```

   This approach of maintaining a directory definition of the system residence volume is preferred at most installations.

2. You can also define your system residence volume after you log on by using the CP LINK command. For example, if your VS2 system programmer "owns" the SYSRES pack and keeps it in his virtual machine at virtual address 150 you could gain access to it with the CP command:

   ```
   link vs2sysp 150 250 w farmer
   ```

   where VS2SYSP is his userid and FARMER is the write password.

3. The VM/370 system operator (or any privilege class B user) can exclusively attach the entire system residence volume to your userid:

   ```
   attach 152 to osvs as 250
   ```

   where 152 is the real device address on which the system residence volume is mounted.

VIRTUAL DEVICES

AS the OS operator, you must know the addresses of your virtual devices and how to use them:

1. The virtual card reader, generally at virtual address 00C (for a 2540) or 012 (for a 3505), reads the OS/VS job input stream.

2. A virtual printer, usually at virtual address 00E (for a 1403) or 002 (for a 3211), hanles the printed output generated by the OS/VS system.

3. The virtual punch, typically at virtual address 00D (for a 2540) or 013 (for a 3525), receives punched output from the OS/VS machine.

4. In addition tape and direct access storage devices are attached to your virtual machine either by directory entry control statements or by the ATTACH or DEFINE command.

The addresses indicated above are generally accepted at most installations. However, system generation specifications may require that other virtual devices exist in the configuration. If, for example, OS/VS expects a 3211 printer at device address 002 and your directory entry does not contain this assignment, the following CP DEFINE command can be used:

```
define 3211 002
```

Your operator's console must be at the address specified during the OS/VS system generation. The easiest way of ensuring that this holds true is to define the appropriate console address in your directory entry. For example, the directory control statement:

```
CONSOLE 01F 3210
```

defines a virtual 3210 console at virtual address 01F.

Before using OS/VS, you should find out from your OS/VS system programmer what its virtual device requirements are.

You can take advantage of VM/370's spool file system to handle printer or punch output that does not necessarily need to be printed or punched. For example, if you are using the "flip-flop" technique, you can route your print output to your virtual card reader by using the CP SPOOL command:

```
#cp spool printer to *
```

You can subsequently load the CMS system, create a CMS file from the data in your virtual reader via the CMS READCARD command, and scan the contents on your terminal using the CMS Editor facility.

PREPARING JOBS FOR AN OS/VS VIRTUAL MACHINE

You can prepare and submit a job stream to an OS/VS virtual machine in one of two ways:

- A deck of real punched cards containing the appropriate job control, program and data can be placed in the real card reader. Place a CP ID card in front of your jobstream deck indicating your OS/VS userid.

For example, either:

```
ID OSVS

-- or --

USERID OSVS
```

are valid ID statements directing the input that follows to user OSVS's virtual reader (the reader with the lowest virtual device address).

- Use the CMS system to create a CMS file containing card images of what you would normally submit through a card reader on a real System/370. Enter the CP SPOOL command to cause subsequent punched output to be directed to the virtual card reader of the OS/VS machine and enter the CMS PUNCH command to generate the virtual card deck:

```
cp spool punch to osvs
punch vsjob27 jcl (noheader
```

The NOHEADER option of the PUNCH command suppresses the punching of a CMS READ control card at the front of the deck.

A job stream spooled to the OS/VS system by either of the above methods remains in the card reader of the OS/VS virtual machine until an OS/VS reader is started.

When you spool jobs to a virtual machine, you should take the precaution of clearing any data that may remain in your virtual punch from previous jobs. Issue the CP commands:

```
cp spool punch nocont
cp close punch purge
```

to ensure that the virtual punch is purged. The first command is required if the punch had been originally spooled with the CONT option.

When running OS/VS in a virtual machine environment, a prime consideration is job entry and output retrieval. There are several techniques that can be employed in this regard:

- You can use the OS/VS virtual machine in batch mode, where it operates as a native OS/VS system. Job streams can be read in through a dedicated card reader. Printed output can be generated by the virtual machine on a dedicated printer.

- A single directory entry can contain a configuration sufficient for running both CMS and OS/VS. You can load CMS to create and edit OS/VS job streams and to check the OS/VS output. You can load OS/VS to run the OS/VS job streams. This method of using two different systems under a single userid is called the "flip-flop" technique. This technique is discussed in greater detail below.

- Two different userids can be used to keep two virtual machines running. One userid can be used for the OS/VS machine while the other can be used for a CMS machine to create jobstreams and inspect output. With the CP DISCONN command, you can run both virtual machines from the same terminal. In effect, both machines are running concurrently but you communicate with only one at a time. If two terminals are available, each system can run independent of the other; this is the optimum environment.

LOADING THE OS/VS SYSTEM

Log on to the userid that is going to be running the OS/VS system. It may be your own userid if you are running "flip-flop" or concurrent systems. It may be a special userid that your installation has set aside exclusively for the OS/VS machine.

```
logon osvs
```

Since extended control (EC) mode is required for the operation of OS/VS, your directory entry should contain the ECMODE specification on the OPTION control statement. If it is not, you must enter the CP command:

```
set ecmode on
```

to enable extended control mode simulation. If you are going to run OS/PCP, OS/MFT, or OS/MVT, you do not need ECMODE; the REALTIMER option, however, is normally required.

At this point, between logging on and loading of your operating system you may find it desirable or necessary to alter your virtual machine's storage size. If, for example, your directory entry specifies one megabyte of storage and for this particular terminal session you need two megabytes, issue the CP DEFINE command:

```
define storage as 2m
```

Virtual machine storage can be redefined up to the limit set in the USER control statement of your directory entry.

Once the IPL volume is made available, you can enter the command to load OS/VS:

```
ipl 250
```

OS/VS will respond with a "SPECIFY SYSTEM PARAMETERS" message. The proper use of a system parameter list (IEASYS00 or IEASYSxx), created during or subsequent to the OS/VS system generation process, can result in a significant saving of time.

Commonly used operator commands can be placed in the SYS1.PARMLIB data set to shorten the IPL process.

SYSTEM OPERATION

Figure 8 shows how an OS/VS1 system is loaded under VM/370. In the example, you will note that the virtual selector channels have been redefined to block multiplexer. The VM/370 directory entry should also have the BMX option specified on the OPTION control statement. This is a performance consideration designed to improve the processing of I/O operations.

Another performance consideration is the size of the virtual machine's storage relative to an OS/VS1's virtual storage size. VS1 has been shown to operate more efficiently in a virtual environment if it is not forced to do any paging of its own. To force VS1 into a non-paging mode, use VM/VS Handshaking and define the storage size for the virtual machine to be the same as the virtual storage size requested by VS1. VS1, in non-paging mode, avoids many privileged instructions thereby reducing CP's overhead.

```
LOGON AT 01:96:14 EST WEDNESDAY 10/29/75
def stor 3072k
STORAGE = 03072K
q set
MSG ON , WNG ON , EMSG TEXT , ACNT ON , RUN OFF
LINEDIT ON , TIMER REAL , ISAM OFF , ECMODE ON
ASSIST ON SVC , PAGEX OFF
IMSG ON
def 009 01f
CONS 01F DEFINED
q chan
CHANNELS = SEL
def chan bmx
CHANNELS = BMX
i 250
IEA760A SPECIFY VIRTUAL STORAGE SIZE
r 00,'3072'
IEA788I NON-PAGING MODE OF VS UNDER VM/370
IEE054I DATE=75.300,CLOCK=009.00.00
IEE054I DATE=75.300,CLOCK=009.00.02,GMT
IEA764I NIPNULL,CMDMON,DFNPARM,JESNULL,,,,SETPARM,,
IEA101A SPECIFY SYSTEM AND/OR SET PARAMETERS FOR RELEASE 04.0 OS/VS1
r 00,'u'
IEA106I IEAAPF00 NOT FOUND IN SYS1.PARMLIB
 IEE140I SYSTEM CONSOLES
   CONSOLE/ALT CMD AUTH     ID       ROUTCP
     01F/01F    M   ALL     01       1-10,10-16
IEF031I SYSGEN VALUES TAKEN FOR JES
IEF866I DEFINE COMMAND BEING PROCESSED
IEE804I PO=(C=AOB,576'K,A,I),P1=(C=AOB,576K,A,E,LAST),
IEE804I P2=(INACTIVE),P3=(INACTIVE),
IEE804I P4=(INACTIVE),P5=(INACTIVE),
IEE804I P6=(INACTIVE),P7=(INACTIVE),
IEE543I 250K BYTES FREE SPACE
IEE817I TMSL=NONE
IEE805I DEFINITION COMPLETED
IEE101A READY
IEE009I JLPRM=(U)
IEE050I MN JOBNAMES,T
IEE048I INITIALIZATION COMPLETED -
```

Figure 8. Sample IPL of VS1 under VM/370.

You can use other OS/VS operator commands to control the OS/VS system, hold and release queues and jobs, start initiators or define partitions. For example:

```
s wtr,00e,,c
```

will start a system writer for class C output on virtual device 00E. The progress of the command's execution can be observed via the OS/VS messages:

```
IEF403I WTR STARTED TIME=05.21.58 P00
IEF236I ALLOC FOR WTR     00E P00
IEF237I 00E ALLOCATED TO IEFRDER P00
IEF049I JOB27             ON DEVICE 00E
IEF868I 00E WTR WAITING FOR WORK
```

If you are using CMS, initial operator commands can be incorporated into a CMS EXEC procedure as part of the job stream. For example, you may want to create the following EXEC procedure called SETUPVS:

```
CP LINK VSSYS 250 250 RR OSPASS
CP DEFINE 009 AS 01F
CP IPL 250
```

After starting the appropriate OS/VS readers, the virtual machine is ready to receive input from card readers, DASD, or tape drives.

COMMUNICATING WITH CP

During the operation of the OS/VS virtual machine, you can use CP commands to communicate with the VM/370 system operator or other virtual machine users, or query and alter the status of your configuration and spool files. In general, you can enter any of the CP commands normally permitted under your userid's privilege class.

How you enter a CP command while your OS/VS virtual machine is running depends on your terminal mode as defined by the CP TERMINAL command (or its default value). If you are not running as the VM/370 system operator, your default terminal mode is VM. In this case, pressing the attention key (or its equivalent) once passes an interrupt pending condition to your virtual machine operating system. Pressing the attention key twice places your virtual machine in the CP command environment from which you can enter CP commands. for a complete description of the use of the attention key, see the section on "Interrupting the Execution of a Command" in the VM/370: CP Command Reference for General Users.

Using the #CP Function

In most cases during your virtual machine operation, you can use the #CP function to enter CP commands directly from the virtual machine environment. If your virtual machine has issued a read to your terminal, you can enter a CP command with the #CP function. For example if you are no longer using a virtual tape drive 397 that was mapped to a real tape drive 492, you could issue:

```
#cp detach 397
#cp msg op i am done with real drive 492
```

The command lines are processed immediately by CP and the virtual machine read remains outstanding.

Note: You may not always be able to enter CP commands with the #CP function. The read issued by OS/VS at the terminal must be for at least as many bytes as you enter in the #CP command line; any additional information is truncated. If the read is for at least three bytes, you can enter:

```
#cp
```

This will place you in CP command mode from which you can enter CP commands directly. To return to your virtual machine environment, enter:

```
begin
```

USING OS/VS IN BATCH MODE UNDER VM/370

If many users submit jobs to a single OS/VS virtual machine, someone is generally needed to tend the machine as an operator. As the virtual machine operator, you must make those decisions required of an operator on a real machine, i.e., deciding what work is going to be done and what is the most efficient way of doing it.

Jobs for the OS/VS machine can be read in through a real card reader. If the real card reader is not dedicated to your OS/VS virtual machine, all that is required is that each job stream be preceded by an ID card:

```
USERID OSVS A
```

where USERID (or ID) indicates the valid beginning of an ID card, OSVS is the name of the VM/370 user to receive the card input in his virtual reader, and A is the VM reader class.

If the VM/370 system operator has dedicated a card reader to your OS/VS virtual machine, then the ID card must be omitted from the front of the deck. Jobs can be sent to the OS/VS machine from other virtual machines. To do this, a user needs to spool his punch to the OS/VS userid:

```
#cp spool punch to osvs
```

Entering this statement causes subsequent punched output to appear in OSVS's virtual card reader.

ALTERNATING OPERATING SYSTEMS

If you are working in a program development environment, rather than a production environment, and you are unable to test your programs directly under CMS, you can alternate the OS/VS and CMS operating systems in a single virtual machine. The advantages are:

- Reduced unit record output. You can examine program output and compiler listings online, check the results and resubmit the job without producing any output on the system unit record devices.
- Generally faster turnaround time than in a batch environment.

The technique for alternating operating systems is outlined below. To use this technique, you should be familiar with the CMS Editor and file manipulation commands found in the VM/370: CMS User's Guide.

LOAD CMS INTO YOUR VIRTUAL MACHINE

To load CMS into your virtual machine, use the CP IPL command and specify either a saved system name or a device address:

```
ipl cms

-- or --

ipl 190
```

When CMS responds with a message like:

```
CMS VERSION 3.0
```

you can then enter the CMS commands to create an OS/VS job stream.

USE THE CMS EDITOR TO PREPARE JOB STREAMS

Suppose you want to compile a PL/I program under OS/VS, and your PL/I source file is available as a file called PLI27 DECK in the CMS file system.

The following CMS procedure will create an OS/VS job stream that can be passed to the OS/VS virtual machine's reader:

```
edit pli127 jcl                          open a CMS file by name
input                                    enter input mode
//pli127 job cps,fred,msglevel=1         enter jcl entries
//cat exec plifc                         |
//sysin dd *                             V
(null line)                              return to edit mode
getfile pli127 deck                      user source deck
input                                    enter input mode
/*                                       enter jcl entries
//                                       V
(null line)                              return to edit mode
file                                     write the file to disk
```

ISSUE SPOOL COMMANDS TO CONTROL UNIT RECORD DEVICES

Spool your virtual punch to yourself:

```
cp spool punch to *
```

Subsequent punched output will appear in your own virtual card reader. Thus, to submit a job to your OS/VS machine, punch the JCL and associated card data.

Spool your virtual printer to your virtual card reader:

```
cp spool printer to *
```

so that printed output is not routed to the real printer, but, instead, appears in your virtual card reader. Each "print" file can then be read by your CMS machine, examined, and, either purged or printed at the real printer.

Note: Since you may find both punch and printer files in your virtual reader, you may want to consider using the spool file class attribute to control which files are to be processed at any one time.

PUNCH THE CMS FILES

Use the CMS PUNCH command to transfer the jobstream to the virtual card reader of the OS/VS virtual Machine:

```
punch pli27 job (noheader
```

The NOHEADER option of the PUNCH command is used to suppress the punching of a CMS READ control card at the front of the output deck.

IPL THE OS/VS SYSTEM

Load the OS/VS system into your virtual machine:

```
cp ipl 250
```

When an OS/VS reader is started, it picks up the job stream that you had previously punched with your punch spooled to your own userid. For example, the OS command

```
s rdr,00c
```

starts a reader on virtual device 00c and reads those cards which appear in the reader queue.

RELOAD CMS INTO YOUR VIRTUAL MACHINE

When the job stream has been processed, reload CMS, and use the READCARD command to create a CMS file from the printed output onto a CMS disk. The output can now be examined with the CMS Editor or TYPE command. If you find that you need hardcopy output, the file can be printed via the CMS PRINT command. When you use the READCARD command to create a CMS file, do not use a filetype of LISTING. If you do, the first character of each line would then be assumed to be a control character and would not be printed.

EXAMINING THE OUTPUT FROM THE OS/VS VIRTUAL MACHINE

If you want to examine the output from the job you executed under OS/VS in your virtual machine, and you spooled your unit record output to your own userid, you can now read the file in your virtual card reader onto a CMS disk with the CMS READCARD command.:

```
readcard pli27 job
```

The CMS TYPE command or Editor can then be used to scan such a file.

If programming errors occurred during the execution of your jostream, you can make corrections to your source program, correct job control statements, or resolve any other execution problems with the CMS Editor and resubmit the job stream.

USING MORE THAN ONE VIRTUAL MACHINE FROM ONE TERMINAL

You can run multiple systems, each in its own virtual machine, and each controlled from its own terminal. However, if multiple terminals are not available, you can use one terminal to control all systems but only one at any one time.

This approach represents a combination of the alternating system and batch techniques. Separate userids are used to run the OS/VS system in one virtual machine and CMS in another virtual machine to prepare jobs and examine OS/VS output.

After submitting a jobstream under the CMS userid, issue the DISCONN or LOGOFF command (depending on how soon you intend to log on to the CMS ID again). Your terminal is now free to be used for running the OS/VS job stream under the OSVS userid:

```
logon cmsid
.
.
.
(route jobs to OSVS user)
.
.
.
disconn
logon osvs
.
.
.
(run jobs)
```

The procedures for running with two virtual machines are much the same as those used in running a single virtual machine in alternating system mode. The primary difference is that you now spool your virtual punch and printer to the other virtual machine instead of spooling them to your own userid:

```
logon cmsid
sp pun osvs
.
. (route jobs to OSVS user)
.
disconn
logon osvs
#cp sp prt cmsid
#cp sp pun cmsid
.
. (run OS/VS jobs)
.
```

CONSIDERATIONS FOR DISCONNECTING AN OS/VS VIRTUAL MACHINE

If you are using more than one userid to alternate communications with operating systems, you should consider:

- How the OS/VS system may read additional jobs from the card reader.
- What happens when a read is issued at the disconnected OS/VS virtual console.

Sending Jobs to a Disconnected OS/VS Machine

If you are planning to use CMS to route jobs to a disconnected OS/VS virtual machine, you should spool the OS/VS reader with the CONT option of the CP SPOOL command:

```
spool reader cont
```

This will allow the OS/VS reader to read more than a single job at a time without operator intervention.

When a Console Read Is Issued to a Disconnected Virtual Machine

If a virtual machine is running disconnected and a read is issued to its console, a 15-minute timeout occurs. If the read is not satisfied within the 15 minutes, the virtual machine is automatically logged off.

It is suggested that a console log be created whenever you are running disconnected. This will provide you with a history of what jobs were run as well as an indication of any unusual circumstances. To start console spooling, issue:

```
sp cons start
```

To stop console spooling and to print the log, issue:

```
sp console stop close
```

HOW CMS CAN HELP YOU DEVELOP AND TEST PROGRAM TO RUN IN AN OS/VS VIRTUAL MACHINE

The above discussions demonstrated how the CMS Editor and EXEC facility can help you prepare jobs for execution in an OS/VS virtual machine. In addition to these CMS features, there are a number of other CMS commands that you can use that may help you in developing and testing programs.

You can use the CMS READCARD and MOVEFILE commands to create CMS files from source programs or existing JCL that you have on cards or magnetic tape.

One of the advantages of storing your source programs on CMS disks is that you can maintain a backup copy of a program while you test and debug a second version. Using the Editor, or commands like COPYFILE, SORT, and RENAME, you can modify and copy CMS disk files.

Refer to the VM/370: CMS User's Guide for information on how to compile and execute many types of OS programs under CMS.

Part 2. Planning and System Generation Information

Part 2 contains information applicable to the system programmer and system analyst when they plan and perform the system generation of the operating systems to be run in virtual machines under VM/370.

The four sections of Part 2 contain the following major topics:

Section 6. System Planning Considerations

- Performance Guidelines
- VM/370 Performance Options
- Virtual Machine Resources
- System Generation Recommendations

Section 7. Configuring Your Virtual Machine

- Directories in General
- Defining Virtual Storage, CPU, and Console
- Defining Direct Access Storage Devices
- Defining Unit Record Devices
- Defining Other Devices
- Other Directory Control Statements
- Control Statements in General
- Representative Directory Entries

Section 8. Generating Your Operating System Under VM/370

- Generation Procedures Under VM/370
- Generating DOS/VS Under VM/370
- Generating OS/VS Under VM/370

Section 9. Special Considerations

- Defining Multiple Consoles
- VM/VS Handshaking
- Using the Diagnose Interface to CP
- Multiple-Access Virtual Machines
- The ASP Virtual Machine
- Channel Switching
- DASD Reserve/Release

Section 6. System Planning Considerations

Section 6 contains information of a system nature rather than operational data as presented in Part 1 of this manual. It is primarily intended as reference material for the system programmer and system analyst when planning the system generation of operating systems to be run in a virtual machine under VM/370.

Section 6 includes the following topics:

Performance Guidelines

- General Information
- VM/370 Performance Considerations

VM/370 Performance Options

- Favored Execution
- Reserved Page Frames
- Virtual=Real
- Priority
- Virtual Machine Assist Feature
- Locked Pages
- Virtual Block Multiplexer Channel

Virtual Machine Resources

- Virtual Machine I/O
- CPU Resources
- Timers in a Virtual Machine

System Generation Recommendations

- DOS/VS System Generation Recommendations
- DOS/VS Accounting
- DOS/VS in a V=R Virtual Machine
- OS/VS System Generation Recommendations
- OS/VS1 Performance
- OS/VS1 in a V=R Virtual Machine
- Miscellaneous Recommendations

GENERAL INFORMATION

The performance characteristics of an operating system when it is run in a virtual machine environment are difficult to predict. This unpredictability is a result of several factors:

- The System/370 model used.
- 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 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/370 performance options in use by one or more virtual machines.
- Whether VM/VS Handshaking is used (VS1 only).
- Whether the Virtual Machine Assist feature or RPQ is installed on the hardware and enabled.
- The type of operating system being used.

Performance of any virtual machine may be improved up to some limit by the choice of hardware, operating system and VM/370 options. The topics discussed in this section address:

1. Some performance suggestions for the placement of CP and CMS resources.
2. The performance options available in VM/370 to improve the performance of a particular virtual machine.
3. The system options and operational characteristics of operating systems running in virtual machines that will affect their execution in the virtual machine environment.

VM/370 PERFORMANCE CONSIDERATIONS

For the best CP paging performance, the CP paging device should be on the highest performance DASD device, should have no other files on the disk, and should be alone on the channel and control unit. If that is not feasible, it is desirable that all the other files on the paging disk, all other disks on the control unit, and all other I/O devices on the channel be relatively inactive so that they do not create a significant bottleneck to paging operations.

A fixed head DASD device such as the 2305 eliminates all seek delays and allows the fastest effective paging rate. If a movable head device

such as the 3330 or 3340 is used, place the CP paging space in the middle cylinders of the disk, since CP's paging cylinder allocation algorithm is most effective at minimizing arm motion there.

The fixed head area of the 3340-70F is not large enough to be useful for CP paging. It may be more worthwhile to place your CMS system there (at least the most active portions of CMS).

All other things being equal, for a given disk type, the smaller capacity disk is preferred for paging to avoid wasting non-paging-DASD space on that disk.

It has been suggested that obtaining more real main storage (up to 1.5 or 2 MB) is a cost-effective way to reduce paging activity.

There does not appear to be any significant advantage to splitting paging over two disks and two channels. If possible, put spooling on a different disk, and preferably, a different channel.

Don't place CP paging space on a 3344 since CP considers the 3344 to be four logical 3340 disks. The paging algorithm will not work efficiently and arm contention may become a significant problem.

Don't use a 3350 in 3330 compatibility mode for use by CP paging and spooling, or for the CMS SYSRES. It is an inefficient use of space and arm contention would again become a problem. Whenever possible, use the 3350 in native mode.

The CP system residence disk space has relatively low activity so its placement is relatively unimportant.

On the other hand, if you have many CMS users, the CMS system residence minidisk has relatively high activity and, if possible, should not be placed on the paging or spooling disk. If you have many CMS users active at the same time, you should consider creating a named CMS system as well as a shared discontiguous segment for the CMS Editor, the EXEC processor, and OS simulation to reduce the paging activity. If your number of active CMS users is "large" ("large" depends on your CPU type, storage size, and DASD configuration), you might consider generating two or more named CMS systems and spreading your CMS users among those CMS systems to reduce DASD arm contention.

SCRIPT files are normally low activity files and can be placed anywhere.

The performance of a specific virtual machine may not equal that of the same operating system running standalone on the same System/370, but in some situations the total throughput obtained in the virtual machine environment can equal or better that obtained on a real machine.

When executing in a virtual machine, any function that cannot be performed wholly by the hardware causes some degree of degradation in the virtual machine's performance. As the control program for the real machine, CP initially processes all real interrupts. A virtual machine operating system's instructions are always executed 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 upon 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, as in the case of a Start I/O (SIO) interrupt (which initiates extensive CP processing).

When planning for the virtual machine environment, consideration should be given to the number and type of privileged instructions to be executed by the virtual machines. Any reduction in the number of privileged instructions issued by a virtual machine's operating system will reduce the amount of extra work CP must do to support that machine.

VM/370 provides certain functions that create a special envionment for one or more virtual machines.

Each of the following functions, (or performance options) can be applied to only one virtual machine at a time:

- Favored execution with guaranteed percentage
- Reserved page frames
- Virtual=real option

Note: The first option and either of the next two can be applied to the same virtual machine.

The following functions can be applied to as many virtual machines as desired:

- Basic favored execution (without guaranteed percentage)
- Priority
- Virtual machine assist
- Locked pages

FAVORED EXECUTION

The favored execution options allow an installation to modify the normal scheduling algorithms and force the system to devote more of its CPU resources to a given virtual machine than would ordinarily be the case. The options provided are:

- The basic favored execution option (no percentage specified).

- The favored execution percentage option.

The basic favored execution option means that the virtual machine so designated is not to be dropped from the active (in queue) subset by the scheduler, unless it becomes non-executable. When the virtual machine is executable, it is to be placed in the dispatchable list at its normal priority position. However, any active virtual machine represents either an explicit or implicit commitment of main storage. An explicit storage commitment can be specified by either the virtual=real 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 work-set at what it would normally have been at queue-drop time. Multiple virtual machines can have the basic favored execution option set. However, if their combined main storage requirements exceed the system's capacity, performance can suffer because of thrashing.

If the favored task is highly compute bound and must compete for the CPU with many other tasks of the same type, an installation can define the CPU allocation to be made. In this case, the favored execution percentage option can be selected for one virtual machine. This option specifies that the selected virtual machine, in addition to remaining in queue, is guaranteed a specified minimum percentage of the applicable time slice, if it can use it. The guarantee is effected by assigning the virtual machine a run priority of zero which keeps it at the top of the run list. When the guarantee has been used up, the normal

calculated run priority is in effect. The favored execution option can only be invoked by a system operator with command privilege class A. The format of the command is as follows:

```
                    ┌   ┐
SET FAVORED userid  |nn |
                    |OFF|
                    └   ┘
```

where:

userid identifies the virtual machine to receive favored execution status.

nn is any value from 1 through 99 and specifies the percentage of the in-queue time slice that is guaranteed to this virtual machine.

OFF specifies that the virtual machine is to be removed from favored execution status.

The percentage option of the SET FAVORED command is administered as follows:

1. The in-queue time slice is multiplied by the specified percentage to arrive at the virtual machine's guaranteed CPU time.
2. The favored virtual machine, when it is executable, is always placed at the top of the dispatchable list until it has obtained its guaranteed CPU time.
3. If the virtual machine obtains its guaranteed CPU time before the end of its in-queue time slice, it is placed in the dispatchable list according to its calculated dispatching priority.
4. In either case (2 or 3), at the end of the in-queue time slice the guarantee is recomputed as in step 1 and the process is repeated.

Whether or not a percentage is specified, a virtual machine with the favored execution option active is kept in the dispatching queues 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 an executable list. If dropped from Q1, the virtual machine is placed directly in Q2 and remains there even though it may exhaust its allotted amount of CPU usage. Virtual machine with this option are thus considered for dispatching more frequently than other virtual machines.

Note, however, that these options, can impact the performance of other virtual machines and the response times of interactive users.

RESERVED PAGE FRAMES

The reserved page frame option provides a specified virtual machine with an essentially private set of real page frames, the number of frames being designated by the system operator, when he issues the CP SET RESERVE command line. Pages will not be locked into these frames. They can be paged out, but only for other active pages of the same virtual machine. When a temporarily inactive virtual machine having this option is reactivated, these page frames are immediately available. If the program code or data required to satisfy the request was in real storage at the time the virtual machine became inactive, no paging activity is required for the virtual machine to respond.

The maximum number of reserved page frames is specified by a class A command of the following format:

```
SET RESERVE userid xxx
```

where xxx is the maximum number required. If the page selection algorithm cannot locate an available page for other users because they are all reserved, the algorithm forces the use of reserved pages. This function can be specified in only one virtual machine at any one time.

Notes:

1. xxx 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.

2. Generally, in order to achieve better performance in a virtual machine, the reserved page frame option should be used in conjunction with the SET FAVORED option. The availability of page frames will not be of much use without the additional access to the CPU.

This option is usually more efficient than locked pages (discussed later in this section) in that the pages that remain in real storage are those pages with the greatest amount of activity at that moment, as determined automatically by the system. Although multiple virtual machines may use the LCCK option, only one virtual machine at a time may have the reserved page frames option active.

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 data communications applications where response times must be kept as short as possible.

VIRTUAL=REAL

Although the VM/370 virtual=real option eliminates CP paging for the selected virtual machine, its main function is to bypass CP's CCW translation. This is possible because I/O from a virtual machine occupying a virtual=real space contains a list of CCWs whose data addresses correspond, except for page zero, to the real storage addresses. 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 virtual=real 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 virtual=real machine is not logged on. This option often increases the paging activity for other virtual machine users, and in some cases for VM/370. (Paging activity on the system may increase substantially, since all other virtual machine storage requirements must be managed with fewer remaining real page frames.)

Except when running OS/VS1 with VM/VS Handshaking which has a non-paging mode of operation, the virtual=real option may be desirable or mandatory in certain situations. The option must be used to allow programs that execute self-modifying channel programs or have a certain degree of hardware timing dependencies to run under VM/370.

The virtual=real area is set up at VM/370 initial program load (IPL). It can be released by the primary VM/370 system operator to be used as part of the dynamic paging area. Once released, it cannot be reclaimed except by reloading VM/370. The virtual=real area must be released in total, that is, unused pages of the area cannot be selected for release.

To use the virtual=real option effectively on a multiport data communication system with no CCW translation (SET NOTRANS ON), either the transmission control unit or communications lines must be dedicated to that system via the ATTACH command or by VM/370 directory assignment. Conversely, on a multiport data communication virtual=real operation, virtual 2701/2702/2703 lines, (that is, lines assigned and used by CP's DEFINE and DIAL commands) operate with CCW translation. The SET NOTRANS ON command is nullified when the use of non-dedicated communication lines is detected.

You cannot load (via IPL) a shared system into a virtual machine with the virtual=real option.

To generate CP so that it properly supports a virtual=real area you must do the following:

- Specify the VIRT=REAL option in the VM/370 directory for all the virtual machines in your installation that you plan to run in the virtual=real area.
- Reserve enough DASD space for the CP nucleus. A CP nucleus that supports a virtual=real area is larger than one that does not.
- Make sure the virtual machine you are using to generate CP has sufficient virtual storage.
- Specify the amount of storage you want reserved for a virtual=real area.

For more information on the VIRT=REAL option, refer to VM/370: Planning and System Generation Guide, "Part 2: Defining Your VM/370 System". This section describes the Directory program, including information about the VIRT=REAL operand of the OPTION control statement.

Note: Portions of the DOS/VS supervisor and OS/VS nucleus must be relocated above page zero to keep CP's page zero from being compromised by input operations. Refer to the "System Generation Recommendations" section for a more detailed description of the problem and the suggested solutions for DOS/VS and OS/VS.

Figure 9 is an example of a real storage layout with the virtual=real option. The V=R area is 128K and real storage is 512K.

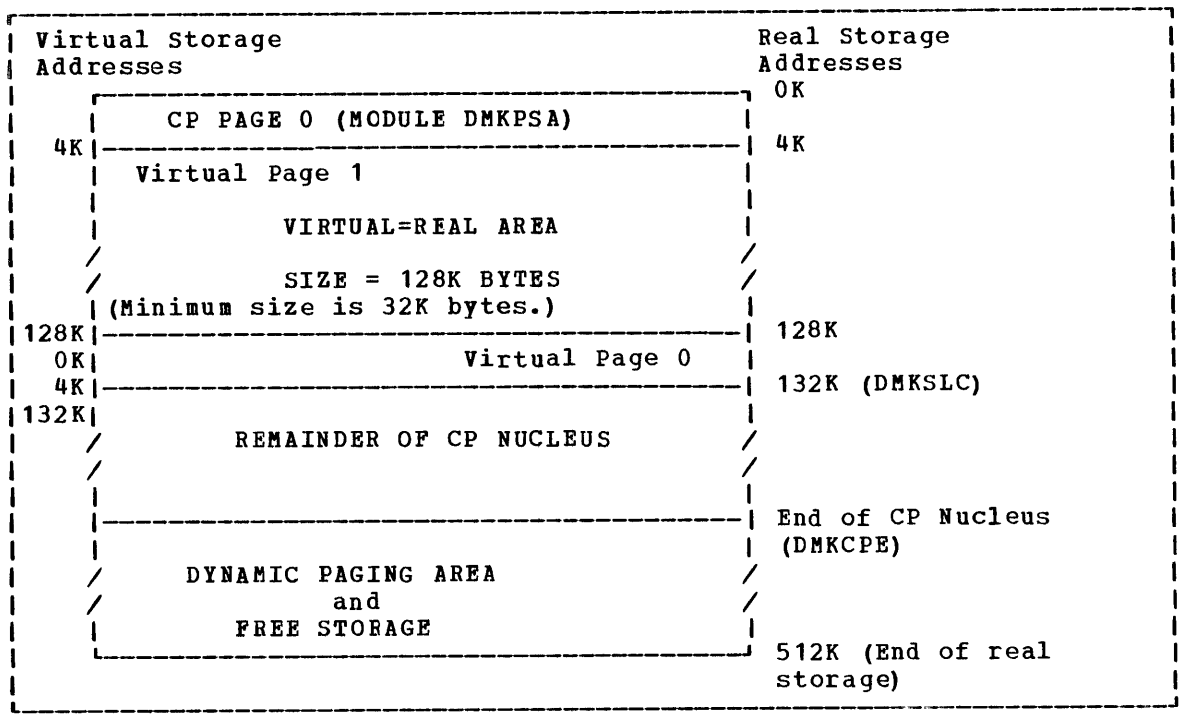


Figure 9. Storage in a Virtual=Real Machine

There are several considerations for the virtual=real option that affect overall system operation:

a. The area of contiguous storage built for the virtual=real machine must be large enough to contain the entire addressing space of the largest virtual=real machine. The virtual=real storage size that a VM/370 system allows is defined during system generation when the option is selected.

b. The storage reserved for the virtual=real machine can only be used by a virtual machine with that option specified in the VM/370 directory. It is not available to other users for paging space, nor for VM/370 usage until released from virtual=real status by a system operator via the CP UNLOCK command. Once released, VM/370 must be loaded again before the virtual=real option can become active again.

c. The virtual machine with the virtual=real option operates in the pre-allocated 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 virtual=real space without translation. The exceptions occur under any of the following conditions:

   - SIO tracing active
   - CAW is in page zero or 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 non-dedicated device (spooled unit record, console, virtual CTCA, or minidisk)

   Any of the above conditions will force CCW translation. Since minidisks are non-dedicated devices, they may be used by programs

running in the V=R region even though CP SET NOTRANS ON is in effect.

d. If the virtual=real machine performs a virtual reset or IPL, then 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 virtual=real virtual machine can issue the command. A message is issued if normal translation mode is entered.

e. A virtual=real machine is not allowed to IPL a named or shared system. It must IPL by device address.

If the virtual machine assist feature is available on a CPU, the system operator can turn the feature off, and on again, for the entire VM/370 system. Also, if the feature is available to VM/370, each virtual machine user can turn the feature off, and on again, for his own virtual machine. When you create your VM/370 directory, you can set off the SVC-handling portion of the virtual machine assist feature for various virtual machines by specifying SVCOFF on the OPTION control statement.

PRIORITY

The VM/370 operator can assign specific priority values to different virtual machines overriding the priorities specified in the VM/370 directory entries. In doing so, the virtual machine with a higher priority (lower numerical value) is considered for addition to the dispatching queues before a virtual machine with a lower priority. User priorities can be altered or set by the following class A command:

```
SET PRIORITY userid nn
```

where userid is the user's identification and nn is an integer value from 1 to 99. The value of nn affects the user's eligible list priority in relation to other users in the system. The priority value (nn) is one of the factors considered in VM/370's dispatching algorithm. Generally, the lower the value of nn, the more favorable the user's position in relation to other users in VM/370's eligible lists.

VIRTUAL MACHINE ASSIST FEATURE

The virtual machine assist feature is a combination of a CPU feature and VM/370 programming. It improves the performance of VM/370. Virtual storage operating systems which run in problem state under the control of VM/370 use many privileged instructions and SVCs that cause interrupts which VM/370 must handle. When the virtual machine assist feature is used, many of these interrupts are intercepted and handled by the CPU; and, consequently, VM/370 performance is improved.

The virtual machine assist feature is available with System/370 Models 135, 145, and 158. It 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 assist feature; it is always handled by CP. The processing of the following privileged instructions are handled by this feature:

```
LRA     (load real address)
STCTL   (store control)
RRB     (reset reference bit)
ISK     (insert storage key)
SSK     (set storage key)
IPK     (insert PSW key)
STNSM   (store then and system mask)
STOSM   (store then or system mask)
SSM     (set system mask)
LPSW    (load PSW)
SPKA    (set PSW key from address)
```

Although the assist feature was designed to improve the performance of VM/370, virtual machines may see a performance improvement because more resources are available for virtual machine users.

USING THE VIRTUAL MACHINE ASSIST FEATURE

Whenever you IPL VM/370 on a CPU with the virtual machine assist feature, the feature is available for all VM/370 virtual machines. However, the system operator's SET command can make the feature unavailable to VM/370, and subsequently available again. The format of the system operator's SET command is:

```
SET SASSIST { ON  }
            { OFF }
```

If you do not know if the virtual machine assist feature is available to VM/370, use the class A and E QUERY command. See the VM/370: Operator's Guide for a complete description of the Class A and E QUERY and SET commands.

If the virtual machine assist feature is available to VM/370 when you log on your virtual machine, it is also supported for your virtual machine. If your VM/370 directory entry has the SVCOFF option, the SVC handling portion of the assist feature is not available when you log on. The class G SET command can disable the assist feature (or only disable SVC handling). It can also enable the assist feature, or if the assist feature is available, enable the SVC handling. The format of the command is:

```
SET ASSIST { [ON] [SVC  ] }
           {     [NOSVC] }
           { OFF         }
```

You can use the class G QUERY SET command line to find if you have full, partial, or none of the assist feature available. See the VM/370: CP Command Reference for General Users for a complete description of the Class G QUERY and SET commands.

RESTRICTED USE OF THE VIRTUAL MACHINE ASSIST FEATURE

Certain interrupts must be handled by VM/370. Consequently, the assist feature is not available under certain circumstances. VM/370 automatically turns off the assist feature in a virtual machine if it:

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

Since an address stop is recognized by an SVC interrupt, VM/370 must handle SVC interrupts while address stops are set. Whenever you issue the ADSTOP command, VM/370 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 line while an address stop is in effect, the response will indicate that the SVC handling portion of the assist feature is off.

Whenever 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 will indicate the assist feature is off.

LOCKED PAGES

The LOCK command, which is available to the system operator (with privilege class A), can be used to permanently fix or lock specific user pages of virtual storage into real storage. In doing so, all paging I/O for these page frames is eliminated. This option is less flexible than the reserved page frame option, but, it is useful if the reserved page frame option is already used by some other virtual machine.

Since this facility reduces total real storage resources (real page frames) that are available to support other other virtual machines, only frequently used pages should be locked 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), it should be the first page of a particular virtual machine that an installation considers locking. The virtual machine interrupt handler pages might also be considered good candidates for locking.

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

The normal CP paging mechanism selects unreferenced page frames in real storage for replacement by active pages. Unreferenced page frames are those whose contents have not been referred to during the last 50 milliseconds. Page frames belonging to inactive virtual machines will all eventually be selected and paged out if the real storage frames are needed to support active virtual machine pages.

When virtual machine activity is initiated 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 must begin processing. Some pages will then have to be paged in so that the virtual machine can respond to the teleprocessing request compared to running the same teleprocessing program on a real machine. This paging activity may cause an increase in the time required to respond to the request compared to running the 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 may ease this problem, but it is not always easy or possible to identify which specific pages will always be 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 re-IPLs his system or loads another system, the locked pages will not be refreshed and a virtual system failure may occur. The SYSTEM CLEAR command will not clear the contents of any of that user's locked pages, even though it will clear virtual machine storage.

THE VIRTUAL BLOCK MULTIPLEXER CHANNEL OPTION

Virtual machine SIO operations are simulated by CP in three ways: byte-multiplexer, selector, and block multiplexer channel mode.

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

Selector channel mode, the default mode, is the mode of operation for any channel that has an attached Channel to Channel Adapter (CTCA), regardless of the selected channel mode setting (the CTCA is treated as a shared control unit and therefore it must be connected to a selector channel). The user need not concern himself as to the location of the CTCA since CP interrogates the related channel linkage and marks the channel as being in selector mode. As in real selector channel operations, CP reflects a busy condition (condition code 2) to the virtual machine's operating system if the system attempts 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. The selection of block multiplexer mode of operation may increase the virtual machine's through-put, particularly for those systems or programs that are designed to use the block multiplexer channels.

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

Selecting the channel mode of operation for the virtual machine can be accomplished via the OPTION control statement in its directory entry or by the use of the CP DEFINE command.

VIRTUAL MACHINE I/O

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, in the case of minidisks, real cylinders. When a virtual machine issues an SIO, CP must:

1. Intercept the virtual machine SIO interrupt.
2. Allocate real storage space to hold the real CCW list to be created.
3. Translate the virtual data and device addresses referred to in the virtual CCWs to real addresses.
4. 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.
5. Generate a new CCW sequence building a Channel Indirect Data Address list if the real storage locations cross page boundaries.
6. Schedule the I/O request.
7. Present the SIO condition code to the virtual machine.
8. Intercept, retranslate, and present the channel end and device end interrupts and status to the appropriate virtual machine, where they must then be processed by the virtual machine operating system.

CP's handling of SIOs for virtual machines can be one of the most significant causes of reduced performance in virtual machines.

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

- Use of large blocking factors (of up to 4096 bytes) for user data sets to reduce the total number of SIOs needed.
- Use of preallocated data sets.
- Use of virtual machine operating system options (such as chained scheduling in OS) that reduce the number of SIO instructions.
- Substitution of 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, there can be a performance gain when CP paging is substituted for virtual machine I/O operations. The performance of an operating system such as OS can be improved by specifying as resident as many frequently used OS functions (transient subroutines, ISAM indexes, and so forth) as possible. In this way, paging I/O is substituted for virtual machine-initiated I/O. In this case, the only work to be done by CP is to place into real storage the page which contains the desired routine or data.

Two CP performance options are available to reduce the CP overhead associated with virtual machine I/O instructions or other privileged instructions used by the virtual machine's I/O Supervisor:

1. The virtual=real option removes the need for CP to perform storage reference translation and paging before each I/O operation for a specific virtual machine.

2. The virtual machine assist feature reduces the real supervisor state time used by VM/370. It is available as a hardware feature on the System/370 Models 135, 145, and 158, and as an RPQ on the Model 168.

Assignment and use of these options is discussed in "Preferred Virtual Machines."

PAGING CONSIDERATIONS

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

The addressing characteristics of programs executing in virtual storage have a significant effect on the number of page exceptions experienced by that virtual machine. Routines that have widely scattered storage references tend to increase the paging load of a particular virtual machine. When possible, modules that are dependent upon each other should be located in the same page. Reference tables, constants, and literals should also be located near the routines that use them. Exception or error routines that are infrequently used should not be place within main routines, but located 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/370 system, and all other virtual machines. The total paging load that must be handled by CP is reduced, and more time is available for productive virtual machine use.

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

CPU RESOURCES

CP allocates the CPU resource to virtual machines according to their operating characteristics, priority, and the system resources available.

Virtual machines are dynamically categorized at the end of each time slice as interactive or noninteractive, depending on the frequency of

operations to or from either the virtual system console or a terminal controlled by the virtual machine.

Virtual machines are dispatched from one of two queues, called Queue 1 and Queue 2. To be dispatched from either queue, a virtual machine must be considered executable (that is, not waiting for some activity or for some other system resource). Virtual machines are not considered dispatchable if the virtual machine:

1. Enters a virtual wait state after an I/O operation has begun.
2. Is waiting for a page frame of real storage.
3. Is waiting for an I/O operation to be translated by CP and started.
4. Is waiting for CP to simulate its privileged instructions.
5. Is waiting for a CP console function to be performed.

QUEUE 1

Virtual machines in Queue 1 (Q1) are considered conversational or interactive users, and enter this queue when an interrupt from a terminal is reflected to the virtual machine. There are two lists of users in Q1, executable and nonexecutable. The executable users are stacked in a first in, first out (FIFO) basis. When a nonexecutable user becomes executable, he is placed at the bottom of the executable list. If a virtual machine uses more than 50 milliseconds (ms) of CPU time without entering a virtual wait state, that user is placed at the bottom of the executable list.

Virtual machines are dropped from Q1 when they complete their time slice of CPU usage, and are placed in an "eligible list". Virtual machines entering CP command mode are also dropped from Q1. When the virtual machine becomes executable again (returns to execution mode) it is placed at the bottom of the executable list in Q1.

QUEUE 2

Virtual machines in Queue 2 (Q2) are considered noninteractive users. Users are selected to enter Q2 from a list of eligible virtual machines (the "eligible list"). The list of eligible virtual machines is sorted on a FIFO basis within user priority (normally defined in the USER record in the VM/370 directory, but may be altered by the system operator).

A virtual machine is selected to enter Q2 only if its "working set" is not greater than the number of real page frames available for allocation at the time. The working set of a virtual machine is calculated and saved each time a user is dropped from Q2 and is based on the number of virtual pages referred to by the virtual machine during its stay in Q2, and the number of its virtual pages that are resident in real storage at the time it is dropped from the queue.

If the calculated working set of the highest priority virtual machine in the eligible list is greater than the number of page frames available for allocation, CP does not continue searching through the eligible list in user priority order, but rather, waits until the proper number of page frames become available.

There are two lists of users in Q2, executable and nonexecutable. Executable virtual machines are sorted by "dispatching priority". This priority is calculated each time a user is dropped from a queue and is based on the ratio of CPU time used while in the queue to elapsed time in the queue. Infrequent CPU users are placed at the top of the list and are followed by more frequent CPU users. When a nonexecutable user becomes executable, he is placed in the executable list based on his dispatching priority.

When a virtual machine completes its time slice of CPU usage, it is dropped from Q2 and placed in the eligible list by user priority. When a user in Q2 enters CP command mode, he is removed from Q2. When he becomes executable (returns to virtual machine execution mode) he is placed in the eligible list based on user priority.

If a user's virtual machine is not in Q1 or Q2, it is because:

1. The virtual machine is on the "eligible list", waiting to be put on Q1 or Q2, or

2. The virtual machine is not runnable because a wait state PSW has been loaded with either no I/O or only terminal I/O active, or

3. The virtual machine execution is suspended because the user is in CP mode executing CP commands.

To leave CP mode and return his virtual machine to the "eligible list" for Q2, the user can issue one of the CP commands that transfer control to the virtual machine operating system for execution (for example, BEGIN, IPL, EXTERNAL, and RESTART).

In CP, interactive users (Q1), if any, are considered for dispatching before noninteractive users (Q2). This means that CMS users entering commands which do not involve disk or tape I/O operations should get fast responses from the VM/370 system even with a large number of active users.

When coming from console function mode, the user is placed in the eligible list for Q1. If he needs still more time after using his Q1 time slice, he is placed on Q2 if there is room; otherwise, he is placed in the eligible list for Q2.

An installation may choose to override the CP scheduling and dispatching scheme and force allocation of the CPU resource to a specified user, regardless of its priority or operating characteristics. The favored execution facility allows an installation to:

1. Specify that one particular virtual machine is to receive up to a specified percentage of CPU time.

2. Specify that any number of virtual machines are to remain in the queues at all times. Assignment of the favored execution option is discussed under "VM/370 Performance Options."

Performance for Time-Shared Multiprogrammed 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.

After determining this, you can perform external observations of turn-around time on benchmarks and specify a point beyond which the addition of more users would be unacceptable. However, when that point is reached, more sophisticated internal measurement is required to determine the most scarce resource and how the bottleneck can be relieved by additional hardware.

Several possible conditions can be identified resulting from different bottlenecks. They are:

- Real storage is the bottleneck; levels of multiprogramming are low compared with the number of contending users. Hence, each user is dispatched so infrequently that running time or response time may become intolerable.

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

- Real storage space may be adequate for the CPU, 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, CPU utilization falls and throughput deteriorates.

- Storage, CPU, 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 CPU utilization.

Estimates of typical working set sizes are needed to determine how well an application may run in a multiprogramming environment on a given virtual storage system. A measure of the application's CPU 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 the extra overhead, the above points may not be too critical. However, if throughput and turnaround time are important, then the converse is true, and these items need close evaluation before allocating resources to a virtual machine operation.

High levels of multiprogramming and overcommitment of real storage space leads to high paging rates. High paging rates can indicate a healthy condition; but, be concerned about page stealing and get evidence that this rate is maintained at an acceptable level. A system with a high rate of page stealing is probably thrashing.

Emphasizing Interactive Response Times

Most of the conditions for good performance, established for the time-shared batch systems, apply equally well to mixed mode systems. However, two major factors make any determination more difficult to make. First, get evidence to show that, in all circumstances, priority is given to maintaining good interactive response, and that non-trivial tasks take place truly in the background. Second, background tasks, no matter how large, inefficient, or demanding should not be allowed to

dominate the overall utilization of the time-sharing system. In other words, in mixed mode operation, get evidence that 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.

TIMERS IN A VIRTUAL MACHINE

This section describes the results obtained in using timers in a virtual machine created by CP.

THE INTERVAL TIMER

Virtual location 80 (X'50'), the interval timer, contains different values than would be expected when operating in a real machine. On a real machine, the interval timer is updated 60 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 only virtual CPU time, and not wait time. It is updated by CP whenever a virtual machine passes control to CP, and this one updating reflects the entire time the virtual machine had control. Note that during the time a virtual machine has control, the virtual interval timer does not change; the virtual CPU time used is added to the virtual interval timer when CP regains control. For some privileged instructions, CP may be able to simulate the instruction and still return control to the virtual machine before the end of that virtual machine's time slice. In such cases, the virtual interval timer is updated but only for those privileged instructions which require normal or fast reflect entry into the dispatcher. For those privileged instructions which do not require entry into the dispatcher, the virtual interval timer is not updated until CP gets control at the end of the time slice.

If the virtual machine assist feature 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 by virtual machine assist to execute privileged instructions is charged to the virtual interval timer.

VM/370 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 CPU 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.

CPU TIMER

A virtual machine must have the ECMODE directory option to use the System/370 CPU timer.

The CPU timer is supported in a virtual machine in much the same way as is the interval timer. That is, the CPU timer in a virtual machine records only virtual CPU time, and it is updated when the virtual machine passes control back to CP.

If the REALTIMER option is specified, the CPU timer reflects all virtual CPU time and virtual wait time except CP time used for services, such as privileged instruction execution, for that virtual machine.

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 CPU Timer (SPT) and Store CPU 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 CPU 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 has one real TOD clock, and all virtual machines can interrogate that real TOD clock. The Store Clock (STCK) instruction is non-privileged; 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/370 is 00:00:00 GMT January 1, 1900.

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 which 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/370 timing facility. It provides 24 or 32 bytes of time and date information in the format shown in Figure 10.

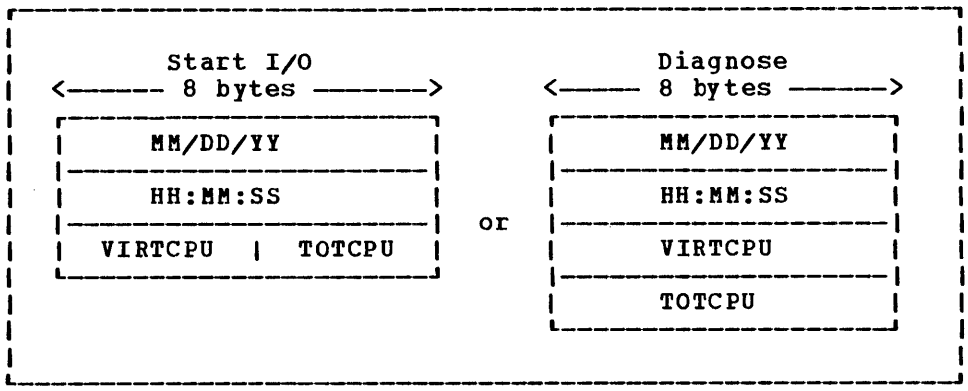


Figure 10. Formats of Pseudo Timer Information

The first eight-byte field is the date, in EBCDIC, in the form Month/Day-of-Month/Year. The next eight-byte field is the Time of Day in Hours:Minutes:Seconds. The VIRTCPU and TOTCPU fields contain virtual CPU and total CPU time used. The units in which the CPU 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 CPU and the total CPU 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 under "DIAGNOSE Instruction in a Virtual Machine." If this method is used, the virtual and total CPU times are expressed as doublewords in microseconds.

Note: The following recommendations were made by VM/370 users that run DOS/VS and/or OS/VS virtual machines and have not been submitted to any formal IBM test. You should evaluate their usefulness in your own environment prior to implementation.

DOS/VS SYSTEM GENERATION RECOMMENDATIONS

Your primary objectives should be to reduce the number of START I/O (SIO) instructions issued by DOS/VS, including its own paging I/O operations, and to avoid double CCW translation.

It is usually better to have several virtual machines rather than to have many partitions active in one virtual machine. That is, if you have a communications system, a batch system, and a test system, you should create a separate virtual machine for each one. If, however, you only run VM/370 part of the day and want to minimize operational differences, one multi-partition production DOS/VS machine may be preferable.

To reduce the number of privileged instructions that must be handled by CP and the virtual machine assist feature, generate a tailored DOS/VS supervisor for each of the virtual machines and leave out any unneeded options. Except for 2314 disks, do not specify seek separation in the FOPT macro. CP issues its own standalone seek. Specify the Block Multiplexer option in the PIOCS macro and in the OPTION statement of the VM/370 directory entry if the computer has block multiplexer channels.

There are two methods of sharing a DOS/VS system residence volume. One method is to have a read/write DOS/VS system residence (SYSRES) minidisk for each DOS/VS virtual machine, and then share a read-only copy of the private core image, relocatable, and source statement libraries. Another method is to share a read-only copy of the DOS/VS SYSRES minidisk, and have separate read/write private libraries for each virtual machine. Note that in the latter case, only one standard label cylinder is available and all virtual machines must co-ordinate their use of that standard label cylinder. Which of these methods you should use depends on your own situation and and personal preference.

In the FOPT macro, specify a sufficient number of SLD (Second Level Directory) entries in order to reduce repetitious reading of the directory.

Use the System Directory List (SDL) in the SVA (shared virtual area) for all job control, disk and tape open and close transients, and for the Attention routine.

If you have sufficient DASD space, let both POWER/VS and CP spool. Generate the POWER/VS system with only the options that suit your needs, but make the I/O buffer sizes as large as possible, up to 2008 bytes. If one job step in a DOS/VS job stream aborts, it is easy to flush the remainder of the job stream using POWER/VS facilities. Using CP spooling alone, you have to cancel each job step manually.

When generating a DOS/VS supervisor, whenever DOS/VS recommends using a 2K boundary or a multiple of 2K, it is better if you use 4K, since that is the size of CP's pages. For the same reason, you should not

default your SEND macro, since DOS/VS rounds the supervisor size to the next 2K boundary. You should calculate the size of the supervisor yourself and specify a 4K boundary in the SEND macro. This forces the BG program to be loaded starting at the next 4K page boundary.

It is usually best to make the DOS/VS partition sizes, and therefore the whole DOS/VS virtual machine, large enough so that all jobs run V=R. Let CP do the paging. Although rotational position sensing (RPS) cannot be used with V=R partitions, it is usually better to avoid double CCW translation and double paging instead of using RPS.

Note: You must specify ",REAL" on the DOS/VS // EXEC job control card.

Set RSIZE equal to the supervisor size plus the sum of all V=R partitions, plus the SVA, plus 32K.

Generate job accounting so DIAGNOSE instructions can be manually added to close printer and punch spool files automatically.

If a program in the DOS/VS core image library may be executed in either a DOS/VS virtual machine or under CMS/DOS, link edit the program with ACTION REL linkage editor control statement so that the DOS relocating loader will be used.

If you are generating DOS/VS under VM/370, you must restore the DOS/VS PID tape onto a full disk pack; a minidisk cannot be used. Create an operational DOS system residence pack on another full pack and then use CORGZ to copy the DOS/VS SYSRES to a minidisk.

Generate several tailored DOS/VS supervisors, if desired, and store them on the same DOS SYSRES with different supervisor names (such as $$A$SUPn, where n is 1, 2, 3, 4, etc.).

Although permitted on a DOS/VS system running stand-alone on the computer, do not use the same DOS file-id for more than one disk if minidisks are used; CP may select the wrong minidisk.

In order to initialize a DOS/VS minidisk for use under VM/370, the VM/370 IBCDASDI service program must be used. If the whole disk is used by DOS/VS, any DOS or OS initialize disk program may be used.

If, for some reason, your VM/370 directory CONSOLE statement specifies an address of 009, you should generate your DOS/VS supervisor with two console addresses, 009 and 01F, so that the DOS/VS supervisor will also be able to run on the real CPU using the real console address of 01F. However, since CMS can use a console address of 01F, you may wish to specify 01F in the CONSOLE statement for ease of use with both DOS and CMS virtual machines. If you need to use more than one DOS/VS console under VM/370, or wish to use a 3270 (or 3158) display device in Display Operator Control mode, refer to "Defining Consoles" in the "Special Considerations" section.

If you need to have a particular virtual machine such as a communiations oriented system perform as efficiently as possible under VM/370, use the CP performance option SET FAVORED with no percentage specified, and also reserve enough page frames to at least equal the number of high activity pages in the virtual machine.

DOS/VS ACCOUNTING

DOS/VS Accounting gives inaccurate and inconsistent elapsed CPU times when operating under VM/370 with Virtual Machine Assist (VMA). This is

because the interval timer (located at virtual storage location X'50') which is used by the DOS/VS Accounting routine is only updated when CP gets control. Therefore, when DOS/VS accesses the interval timer data, a variable amount of time may have elapsed since the last update by CP of the interval timer, and thus, an inaccurate CPU time is recorded.

To attempt to minimize this inaccuracy at the cost of some additional CP overhead, you may wish to consider adding the following dummy DIAGNOSTIC instruction

```
83000000
```

at the following locations in the DOS/VS Supervisor source statements:

- In the SVC 24 routine, before the L R3,SYSTIMER statement.
- In the SVC 52 routine, before the L R3,SYSTIMER statement.
- In the STCLOCK routine, before the STCK CLOCK statement.
- In the timer interrupt handler routine, before the LM R2,R3,SYSTIMER statement.
- In the Job Accounting Initialization routine (JATIMER), before the statement that references SYSTIMER.

DOS/VS IN A V=R VIRTUAL MACHINE

To avoid compromising CP's real page zero, the DOS/VS supervisor must be modified and reassembled to prevent DOS/VS from reading sense information into page zero. The recommended changes to the DOS/VS supervisor are:

1. In the FOPT macro, remove the label 'SSKADR' from the SSKADR DC statement.
2. Following the SGTCON macro, add the following 5-byte DC statement:

```
SSKADR DC CL5
```

3. Assemble and link edit the DOS/VS supervisor again.

This supervisor will now work correctly in a V=R machine as well as on a virtual (or real) machine.

OS/VS SYSTEM GENERATION RECOMMENDATIONS

Generate OS/VS so that all commonly used transient routines are resident in storage. Run all jobs, if possible, as V=R jobs. VS2 Release 1 (sometimes referred to as Single Virtual Storage (SVS)), in fact, can use two address spaces:

- One address space for V=V jobs
- Another address space for the SVS system and any V=R jobs.

If a large enough virtual machine is defined so that all jobs can run V=R, SVS will not have to swap between address spaces and will perform better under VM/370.

OS/VS1 PERFORMANCE

If VS1 Release 3 or above is run in a virtual machine under VM/370, you can improve your VS1 batch performance if accounting repeatability is not too important in your installation. Starting with VS1 Release 3, if you specify:

```
CTIMERS=INCLUDE
```

on the CENPROCS macro during VS1 system generation, VS1 does its job step accounting using the CPU Timer instead of the interval timer.

Because the SET and STORE CPU TIMER instructions are privileged instructions that are not handled by the Virtual Machine Assist feature, a considerable amount of extra overhead is incurred by CP to handle these CPU Timer-related instructions. Although the VS1 system generation publication states that you must specify CTIMERS=INCLUDE for System/370 Models 155 II and above, some VM/370 customers have generated VS1 systems specifying (or defaulting to) CTIMERS=EXCLUDE without apparent ill effect.

Note: If you already have a VS1 system generated with CTIMERS=INCLUDE, to change to CTIMERS=EXCLUDE, you must perform a complete VS1 system generation; it is not possible to just generate a new VS1 nucleus.

VS1 STORAGE LIMITS

The maximum real and virtual storage size supported by VS1 is 4000K (that is, four megabytes). VS1 (Release 4 or below) will not function correctly if it is run:

- On a real CPU having more than four megabytes of real main storage.
- Under VM/370 with a virtual machine storage size greater than four megabytes.

Refer to "VM/VS Handshaking" in "Section 9. Special Considerations" for a description of handshaking between VS1 and VM/370.

OS/VS1 IN A V=R VIRTUAL MACHINE

If the version of VS1 you are using is earlier than VS1 Release 4, and you wish to avoid data transfer operations into real page zero, the system programmer must do one of the following:

- If the VS1 nucleus already exists, replace the existing ORDER statement with the following linkedit control statement in the VS1 linkedit deck:

  ```
  ORDER IEAAIH00,IEAIOS00(P)
  ```

  All INSERT control statements that were produced during the original VS1 system generation process should be utilized.

- If you are going to generate a new VS1 nucleus, you may accomplish the IOS alignment by modifying the ORDER control statement in the VS1 Stage 2 job stream prior to Stage 2 execution.

MISCELLANEOUS RECOMMENDATIONS

In general, if VM/370 will be running all the time and if you are going to run many one- or two-partition DOS/VS virtual machines, DOS/VS should be generated with as few options as possible. This is also true when several virtual machines share a system residence volume.

On the other hand, if VM/370 will be running only part of the time, and DOS/VS will be running standalone the rest of the time, you will want to make the operation of DOS/VS under VM/370 as transparent as possible to the computer and to other DOS/VS users. In this case, you want the same multi-partition DOS/VS system to run under VM/370 and standalone. You should also use POWER/VS and dedicate the appropriate unit record devices to the virtual machine to maintain compatibility.

Very often, options that improve performance on a real machine have no effect (or possibly an adverse effect) in a virtual machine. For example, seek separation, which improves performance on the real machine, is redundant in a virtual machine: CP itself issues a standalone seek for all disk I/O.

Sharing the system residence volume avoids the necessity of having to keep multiple copies of the operating system online. The shared system residence volume should be read-only. An OS system can be shared among users if all data sets with write access are removed from the system residence volume. A DOS system can be shared among virtual machines if each virtual machine has a unique standard label cylinder and its own read/write private core image library where it can catalog programs. Some change to DOS is necessary to support shared system residences. Refer to the VM/370: System Programmer's Guide for more details.

Section 7. Configuring Your Virtual Machine

Section 7 covers the configuration of your virtual machine. A general discussion of directory entries is followed by detailed explantions of each directory entry control statement. This is followed by a number of representative directory entries.

Section 7 contains the following topics:

Directory Entries in General

Defining Virtual Storage, CPU and Console

- USER Control Statement
- OPTION Control Statement
- CONSOLE Control Statement

Defining Direct Access Storage Devices

- MDISK Control Statement
- LINK Control Statement
- DEDICATE Control Statement

Defining Unit Record Devices

- SPOOL Control Statement

Defining Other Devices

- SPECIAL Control Statement

Other Directory Control Statements

- ACCOUNT Control Statement
- IPL Control Statement

Control Statements in General

Representative Directory Entries

- The System Operator's Virtual Machine
- A Virtual Machine to Receive System Dumps
- Production Virtual Machines
- Other System Virtual Machines

Directory Entries in General

The VM/370 system maintains a file of directory entries, usually on the system residence disk, one entry for each virtual machine that will be allowed to have access to the system. Keeping this file updated is the responsibility of the system programmer and the system operator. As additions and/or changes are submitted, an updated file is created via the CMS Editor facility or on punched cards.

A VM/370 Directory service program is then invoked, either by the CMS DIRECT command, or by reading it in from the card reader. The directory entries are checked for syntax errors and if valid they will (1) replace the previous set of directory entries on the directory volume and (2) will become the currently active directory file.

Each directory entry is made up of a number of directory control statements. These statements provide VM/370 with the configuration of your virtual machine. In general your configuration can be thought of as consisting of the following

- Virtual storage, console, and CPU
- Direct access storage devices
- Unit record devices
- Other devices

Figure 11 shows the relationship of a directory entry to both the VM/370 system's real devices and the virtual machine's virtual devices.

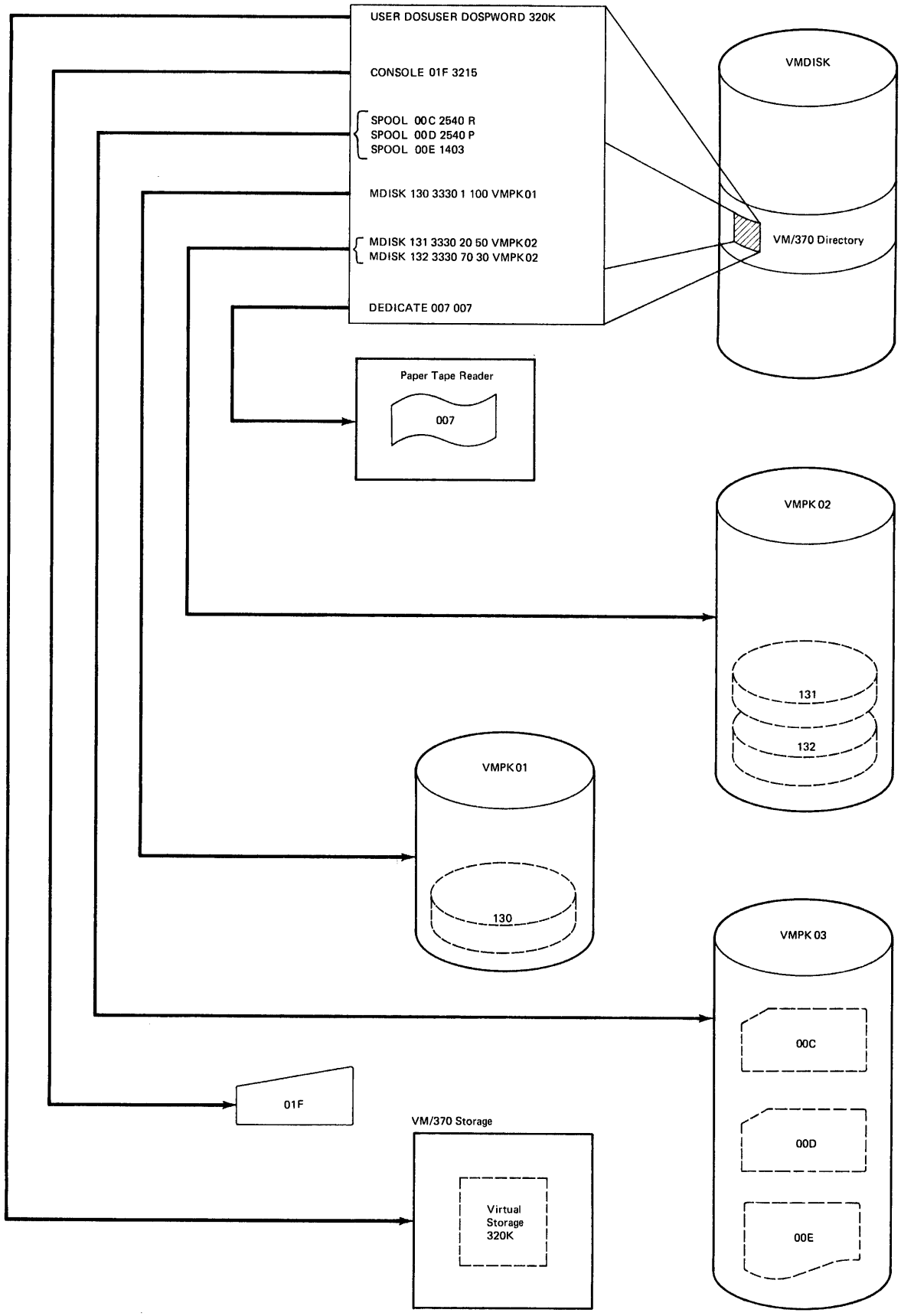


Figure 11. Relationship of VM/370 Directory Entries

One of the most valuable features of VM/370 is the ability to configure a central processing unit that is tailored to a particular function you wish it to perform.

USER CONTROL STATEMENT

Use the USER control statement to enter your virtual machine's identification, password, storage size, CP command class, priority and logical line editing symbols. The USER control statement:

```
USER DOSSYS DSV1234 320k 768k G
```

defines a production DOS/VS virtual machine with a userid of DOSSYS, a password of DSV1234 and a virtual storage of 320K. The maximum virtual storage size specified allows the DOS/VS user to redefine his virtual storage up to 768K, dynamically via the CP DEFINE command. For example, if a particular production run requires 512K of storage, the user can issue:

```
#cp define storage as 512K
```

VM/370 will reset the virtual system, change the virtual machine's size, and clear the virtual storage to binary zeroes. You must now reload your operating system in order to continue.

The specified command class, G, could have been omitted and obtained by default. The scheduling priority defaults to 50 and the line editing characters default to the system defined values. You can change the line editing characters, dynamically via the CP TERMINAL command. For example, the command:

```
#cp terminal chardel <
```

changes the character delete symbol to <.

The userid, password, maximum storage size, and priority affect the VM/370 system's security and performance and therefore can not be changed by the general user. This type of change can only be performed by a system programmer or system operator.

OPTION CONTROL STATEMENT

The OPTION control statement allows you to specify certain options and features for you virtual machine. The operands of the OPTION statement are all keywords and therefore can be specified in any order.

REALTIMER OPTION

Enter REALTIMER if your virtual timer is to be updated during virtual wait time as well as during virtual CPU time. This option is required

if your operating system is to handle timer driven interrupts. If this option is not specified in your directory entry, you can obtain this timing facility by issuing the CP command:

```
#cp set timer real
```

You can also turn the option off with the command:

```
#cp set timer off
```

ECMODE OPTION

Specify ECMODE if your virtual machine will be using operating systems that will be running in extended control mode, using the dynamic address translation (DAT) facility, using extended control registers other than zero, or addressing I/O channels 6 through 15. If this option is not specified in your directory, you can obtain the facility by issuing the CP command:

```
#cp set ecmode on
```

ISAM OPTION

The ISAM option provides for the special handling, by VM/370, of self-modifying channel programs that ISAM generates for some of its operations. If this option is not specified in your directory entry, you can obtain the ISAM facility via the CP command:

```
#cp set isam on
```

You must specify the ISAM option if you are:

- Using ISAM in an OS/PCP, OS/MFT, or OS/MVT virtual machine.
- Using ISAM in a V=R region of an OS/VS virtual machine

You need not specify the ISAM option if you are:

- Using ISAM in a DOS or DOS/VS virtual machine.
- Using ISAM in a V=V region of an OS/VS virtual machine.
- Running in the VM/370 virtual=real area and the NOTRANS option has been set on.

VIRT=REAL OPTION

If you specify the VIRT=REAL option, your virtual machine is eligible to occupy VM/370's low storage so that with the exception of page 0, all other virtual storage addresses correspond to the real storage addresses. VM/370's page 0 occupies the first 4096 bytes of real storage and your virtual page 0 is relocated to a position immediately following the area set aside by VM/370 for V=R operation (see Figure 12).

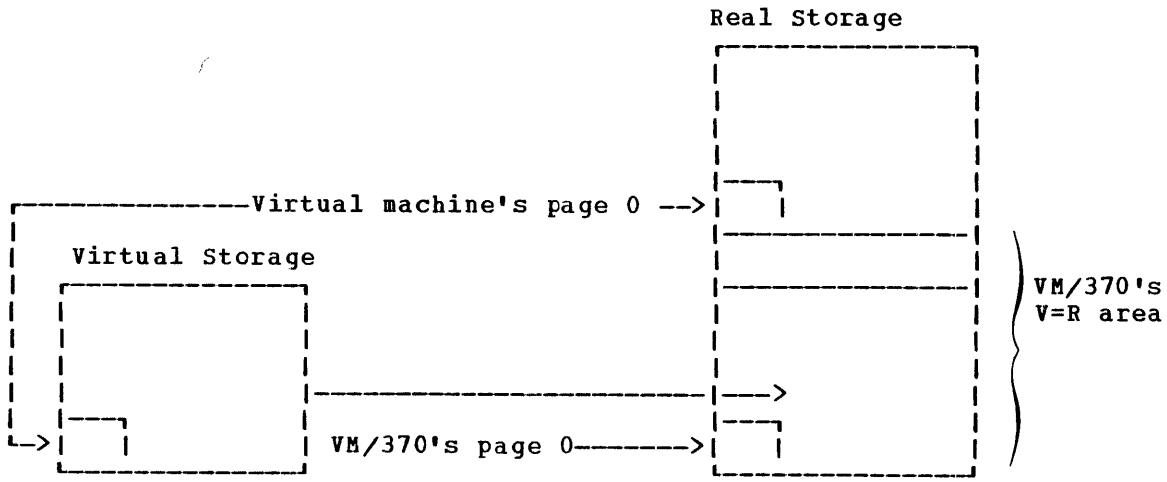


Figure 12. Virtual=Real Machine

This option can be specified for many virtual machines, however, only one virtual machine can occupy the V=R area at any one time. If you have specified this option and the V=R area is occupied when you log on, your virtual machine will be created in virtual=virtual mode and you will be sent a message informing you of the action.

ACCT OPTION

The ACCT option allows you to charge another user for virtual machine resources. For example, if your machine is performing a batch type operation, you can generate job accounting cards for each job processed. See the VM/370: System Programmer's Guide for a discussion of Diagnose code X'4C', generating accounting cards for a virtual user.

SVCOFF OPTION

The SVCOFF option specifies that SVC interruptions will be handled by CP rather than the Virtual Machine Assist feature. You can, however, override this option by issuing the CP command:

```
#cp set assist svc
```

If your operating system uses the SVC 76 interface for error recording by VM/370, the SVC 76 interrupts will be handled by CP whether SVCOFF is in effect or not.

BMX OPTION

The default mode of operation for virtual machine I/O is via selector channels. By specifying BMX, the I/O occurs as block multiplexer channel operations with the following exceptions:

- Virtual I/O operations on channel 0 are always processed as byte multiplexer channel operations.

- Channels, with an attached channel-to-channel adapter, are restricted to selector channel operation.

The channel mode of operations for all but the above exceptions can be changed via the CP command:

```
#cp define channels as sel
        -- or --
#cp define channels as bmx
```

However, when this command is executed, the virtual machine is reset and you must reload your operating system.

CONSOLE CONTROL STATEMENT

The CONSOLE control statement defines the device that will be the console of your virtual machine. Devices that are supported as virtual machine consoles are listed in the VM/370: Introduction. An example of a CONSOLE control statement is as follows:

```
CONSOLE 01F 3215 A
```

The virtual device address is 01F and the device type is 3215. The device address is determined by the channel and device address that the console was assigned during the system generation of your operating system. You can specify either 1052, 3210, or 3215 for device type, regardless of the actual real device used. Only one console device can be defined for a virtual machine. In the event a different console is required, you can use the DEFINE command to either change the console address as in:

```
#cp define 01f as 009
```

or to add an alternate console as in:

```
#cp define console as 009
```

The last operand in the CONSOLE control statement example is a spooling class that applies to printer output should the console data be spooled. The default console spool class is T. For information on how to specify multiple consoles for use by a virtual machine, refer to "Defining Consoles" in "Section 9. Special Considerations."

DASD can be utilized by virtual machines in several ways.

A real disk volume can be shared by several users, each owning a number of contiguous cylinders. This subsetting of a real disk volume is called physical pack sharing and the subset of cylinders is called a virtual disk or minidisk.

The data on a minidisk or an entire disk volume can be shared by two or more users. One user is the owner of the virtual disk and by means of a LINK password, other users are allowed controlled concurrent access. This is called logical data sharing.

A real disk volume can also be dedicated to a specific virtual machine for its own private use. In this instance your virtual operating system is required to perform all necessary interrupt handling, error recovery and error recording.

MDISK CONTROL STATEMENT

You can use the MDISK control statement to define a portion of a direct access volume (or an entire volume) as one of your permanent virtual disks with a specified virtual address. For example:

```
mdisk 250 3330 50 25 oswrk1 wr all
```

defines 25 cylinders starting at cylinder 50 of a 3330 DASD volume with a serial number of OSWRK1. Your virtual machine has read/write access to this minidisk at virtual address 250 and any other user can link to it in read status without a password. The absence of a write password denotes that linkage in write mode is not available.

You can also use the MDISK control statement to define a temporary disk area that although automatically available to you when you log on, is released to the system when you log off. An example of this type of definition would be:

```
mdisk 280 2319 t-disk 15
```

Each time you log on to the VM/370 system, 15 cylinders of temporary disk space on a 2319 volume will be made available to your virtual machine at virtual address 280.

When using temporary disks you must remember to initialize or format the disk area each time you log on; VM/370 allocates whatever space is available at the time and you have no way of knowing its prior usage or content. See "Section 2. General Considerations" for a discussion on initialization and formatting of virtual disks.

Due to the transient nature of a temporary disk and to preserve security, you should always clear the area before you log off.

Temporary disk space can also be defined dynamically via the CP DEFINE command. The above example of a T-DISK defined on a directory control statement could be duplicated with the following command, issued during the terminal session:

```
#cp define t2319 as 280 cyl 15
```

See "Appendix B: VM/370 Restrictions" for restrictions that apply to the definition of minidisks.

LINK CONTROL STATEMENT

The LINK control statement is used to gain access to someone else's virtual disk. The control statement requires that you know the userid of the virtual machine that owns the disk and the virtual address by which it is known to the owner. For example, the control statement:

```
LINK SMITH 330 310 R
```

provides you with a read-only link to a virtual disk that user SMITH owns at virtual address 330. Your virtual machine can then access the disk at virtual address 310.

The action of the LINK control statement can be simulated dynamically via the CP LINK command. The above example of linking, when performed from a virtual machine environment would look like:

```
#cp link to smith 330 as 310 rr
```

DEDICATE CONTROL STATEMENT

The DEDICATE control statement can be used to provide your virtual machine with a virtual disk that is mapped in a one-to-one correspondence to a real DASD. You can define a dedicated DASD in either of two ways. The control statement:

```
DEDICATE 250 350
```

specifies that whenever you log on, addressing device 250 will give you sole access to the volume mounted on the disk device at real address 350. On the other hand, the control statement:

```
DEDICATE 250 DOSRES
```

provides that whenever you log on, addressing device 250 will give you sole access to whatever real device has the volume labeled DOSRES mounted on it.

The second method has the clear advantage that if a malfunction should occur on the device on which the DOSRES volume is mounted, the volume can be transferred to another device and the virtual machine can continue to run. In the first example, the operator would not only have to mount the volume on another device but would also have to issue the CP commands:

```
detach 350 from yourid
attach 351 to yourid as 250
```

to provide the new virtual-to-real address correspondence.

If you should require a dedicated DASD only infrequently, you can always request a dynamic ATTACH from the system operator. For example, the message:

```
#cp msg operator pls attach 350 as 250
```

would result in the operator issuing:

```
attach 350 to yourid as 250
```

and you would receive the confirmation:

```
DASD 250 ATTACHED
```

Note: When dedicating a 2305 device, both the real and virtual device addresses must specify the first exposure on the 2305 (device address 0 or 8). Processing, however, takes place for all 8 exposures, 0 through 7 or 8 through F.

With very few exceptions, it would be quite inefficient to provide each virtual machine with its own dedicated unit record devices. VM/370 provides an intermediate auxiliary storage area for the input and output of all the virtual machines. This storage area resides on a direct access device and is called the spooling area. VM/370's control program manages the data flow between the virtual machines, the spooling area, and the real unit record devices.

SPOOL CONTROL STATEMENT

To have your virtual machine's input and output handled by VM/370 you include one or more of the following control statements in your directory entry:

```
SPOOL 00C 2540 READER
SPOOL 00D 2540 PUNCH
SPOOL 00E 1403
```

If your operating system supports two readers, you could add the second reader to the above configuration as follows:

```
SPOOL 012 3505
```

Additional spooling devices can also be added dynamically. For example, the command:

```
#cp define 3525 as 013
```

adds another punch, and

```
#cp define 3211 as 002
```

adds another printer.

Remember that devices added via the CP DEFINE command are part of your virtual machine for the current terminal session only.

In most cases, spooling represents the most efficient way of handling the unit record input and output of many virtual machines. However, special cases may justify the dedication of a real unit record device to a single virtual machine.

One special case would be if your virtual machine's operating system does its own spooling; for example, POWER/VS under DOS/VS or JES under OS/VS. You can eliminate double spooling of printer output by including a DEDICATE statement such as:

```
DEDICATE 00E 002
```

in your directory. This will pass all virtual printer 00E output directly to the real printer at 002.

Another case where you may want a unit record device dedicated to your virtual machine would be if your virtual machine produced a sufficient volume of output to keep the device busy.

You can have a unit record device dynamically dedicated to your virtual machine by the system operator. Send the system operator the message:

```
#cp msg operator pls attach punch at 00d
```

and, if a punch is free at 00d, he will issue the command:

```
attach 00d to yourid as 00d
```

You will receive a confirmation message:

```
PUN 00D ATTACHED
```

When you have no further use for the device, issue the command:

```
#cp detach 00d
```

Other input and output devices such as magnetic tapes, channel-to-channel adapters, pseudo timers, communication lines, and devices that while available to the System/370 are not supported by VM/370 can be included in your directory entry.

SPECIAL CONTROL STATEMENT

Use the SPECIAL control statement to add I/O devices that do not necessarily require corresponding real devices.

The SPECIAL control statement only provides your virtual machine with a virtual address for a specific type of device. In order to utilize the device, you must issue an appropriate CP command.

For example, you can specify a virtual channel-to-channel adapter as:

```
SPECIAL 070 CTCA
```

Some other user, userb, can also specify a similar device:

```
SPECIAL 080 CTCA
```

To communicate with one another, one of the users must issue the CP COUPLE command. If you decide to do the coupling, you would issue:

```
#cp couple 070 to userb 080
```

and receive the confirmation:

```
CTCA 070 COUPLE TO USERB 080
```

USERB would be sent the message:

```
CTCA 080 COUPLE BY YOURID 070
```

You can use the SPECIAL control statement to specify a virtual transmission control unit for a multiple-access operating system. If your system requires three communication lines from a 2703 you can specify:

```
SPECIAL 061 2703 IBM
SPECIAL 062 2703 IBM
SPECIAL 063 2703 TELE
```

Before a terminal can communicate with your multi-access system, it must issue the DIAL command:

```
dial yourid
```

to connect to any available line part; or issue:

```
dial yourid 062
```

to connect to a particular line.

Note that of the three lines that were specified, one was a teletypewriter line and two were IBM terminal lines. When the DIAL command is issued with no specific address, VM/370 will connect the terminal to any available, compatible (IBM or TELE) line belonging to the specified userid. If no lines are available, or if all lines are busy, an error message is issued and the connection is not made.

A device such as a magnetic tape drive can be used by only one virtual machine at a time; it is therefore configured into your directory entry with a DEDICATE statement. The statement:

```
DEDICATE 181 281
```

will allow your operating system to access the device at real address 281 via a virtual address of 181.

You can also use the DEDICATE statement to place a device that is not supported by VM/370 into your virtual machine configuration. The device must be physically connected to the System/370, it must be supported by your virtual machine's operating system, and it must not violate any of the VM/370 restrictions contained in Appendix A. You could, for example, include the statement:

```
DEDICATE 007 012
```

where real address 012 could represent a 2671 Paper Tape Reader that is part of the System/370 on which VM/370 is running. If your operating system was generated with a 2671 defined at address 007, VM/370 will handle the device and CCW address translation associated with reading from the device. Your operating system will be responsible for error recovery and error recording procedures.

There are two other directory control statements. One provides accounting information to VM/370 and the other provides for the automatic loading of an operating system whenever you log on to the system.

ACCOUNT CONTROL STATEMENT

When you log off the system (and also upon command from the system operator, VM/370 will punch accounting information containing your virtual machine's usage of real system resources. This usage is charged to a userid and an account number.

The ACCOUNT control statement contains a unique account number for your virtual machine and an optional distribution code as in the following example:

```
ACCOUNT 13578642 BIN346
```

The accounting cards that are punched for your virtual machine will have your userid in columns 1 through 8 and the account number 13578642 in columns 9 through 16. The separator sheets for spooled printer output and the separator cards for spooled punch output will contain your userid and the distributions code, BIN346.

If the ACCOUNT statement is used, it must follow the USER control statement and precede any device statement. If the ACCOUNT statement is omitted, both the account number and the distribution code default to the userid.

IPL CONTROL STATEMENT

If your virtual machine runs one operating system most of the time, you can have that system automatically loaded each time you log on. Use the IPL statement to identify your operating system to VM/370. If you specify:

```
IPL 350
```

the virtual address 350 represents the address of the device that contains the system to be loaded. Or, if your virtual machine's operating system has been "saved" via the CP SAVESYS command, you can specify:

```
IPL DOSVS
```

where DOSVS is the name under which the system was saved.

If the system operator of your installation is to have the capability of automatically logging your virtual machine on via the class A or B AUTOLOG command, your virtual machine directory entry must contain an IPL control statement.

Your virtual machine, as defined in the VM/370 directory, can be dynamically modified after you have logged on. In the discussion of each directory control statement, the operands that can be overridden with CP commands will be specified. Note, however, that when you override your directory entry, the modified configuration is in effect for the current terminal session only. The next time you log on, your original directory configuration will be in effect.

In the preceding discussion of the individual directory control statements, with the exception of the OPTION statement, all operands are positional. You must include all possible operands that precede the last one you specify and operands that would normally fall to the right of the last one specified will take default values. See "Part 2: Defining Your Virtual Machine" of the VM/370: Planning and System Generation Guide for detailed directory statement formats.

When using the SPOOL, DEDICATE, and SPECIAL control statements to define virtual devices, you must be careful to specify virtual addresses that will not result in conflict or contention in the virtual control unit interface. Virtual addresses that indicate the same channel and control unit will share a common virtual control unit block. If, for example, you specify:

```
SPOOL 102 3211
SPECIAL 103 3270
```

the control unit 0 on channel 1 will control both a nonshareable device (the 3211 printer) and a shareable device (the 3270 display unit). Processing of channel programs involving these two devices can result in a hung or busy condition.

When using the DEDICATE control statement to attach a 2305 to your virtual machine, both the real and virtual addresses must refer to the first exposure on the unit. The first exposure (or base device address) of the 2305 is 0 or 8; the resulting address will be of the form xx0 or xx8. When the statement is processed, however, all eight exposures, 0-7 or 8-F, are included.

The following sample VM/370 directory entries provide an installation with some of the entries necessary for operation and updating. The indenting shown in the sample directory entries is for clarity only, and is not required by the Directory program. LINK control statements are used whenever possible to minimize the number of changes to the VM/370 directory whenever a minidisk extent is moved. A brief explanation of some of the virtual machine userids follows.

THE SYSTEM OPERATOR'S VIRTUAL MACHINE (OPERATOR)

The userid for the following directory entry must be the same as the userid that was specified on the SYSOPER operand of the SYSOPR macro when the VM/370 system was generated. The operator is given all command privilege classes except class F. Actually, if other virtual machines are defined with command privilege classes appropriate for updating VM/370, the operator's virtual machine only needs class A command privileges. The 191 minidisk that is defined contains CMS files and EXEC procedures, along with the service programs used to update VM/370.

```
USER OPERATOR PASSWORD 256K 1M ABCDEG
     ACCOUNT ACCTNO BIN1
          CONSOLE 009 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          LINK VMSYS 190 190 RR
          MDISK 191 3330 1 10 UDISKA WR RPASS WPASS
```

A VIRTUAL MACHINE TO RECEIVE SYSTEM DUMPS (OPERATNS)

The userid for the following directory entry is the userid that was specified on the SYSDUMP operand of the SYSOPR macro when the VM/370 system was generated. All abnormal termination dumps are sent to this virtual machine. This user is normally given command privilege classes A, B, C, and E. If the directory entry contains all of the disks normally attached to the system, described as full-volume minidisks, you can rewrite the VM/370 directory by using the DIRECT command. The operations group can also examine any disk while it is attached to the system, if you define these disks as full-volume minidisks.

```
USER OPERATNS PASSWORD 320K 1M ABCEG
     ACCOUNT ACCTNO BIN2
          CONSOLE 009 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403 A
          LINK VMSYS 190 190 RR
          MDISK 191 3330 101 10 UDISKA WR RPASS WPASS
          MDISK 350 3330 0 404 SYSRES WR RPASS WPASS
          MDISK 351 3330 0 404 SYSWRK RR RPASS WPASS
          MDISK 250 3330 0 404 UDISK1 RR RPASS WPASS
          MDISK 251 3330 0 404 UDISK2 RR RPASS WPASS
          MDISK 232 2314 0 203 BATCH1 RR RPASS WPASS
          MDISK 233 2314 0 203 BATCH2 RR RPASS WPASS
          MDISK 234 2314 0 203 TSOSYS RR RPASS WPASS
          MDISK 235 2314 0 203 TSOWRK RR RPASS WPASS
```

Some installations may want to combine the functions of OPERATOR and OPERATNS into one virtual machine. This can be accomplished in the above examples by adding the last 8 control statements of the directory entry for OPERATNS to the directory entry for OPERATOR. The OPERATOR virtual machine can then perform all the system functions required to operate the VM/370 system.

PRODUCTION VIRTUAL MACHINES

The following directory entries represent some batch type virtual machines that can be used to run production jobs under various operating systems. The operands that are specified on the OPTION control statement reflect the requirements of the particular system being used. Disk space can either be dedicated or shared with other systems.

A DOS VIRTUAL MACHINE

```
USER DOSUSER PASSWORD 256K
     ACCOUNT ACCTNO BIN3
     OPTION ACCT
          CONSOLE 01F 3215
          SPOOL 00B 2501
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          MDISK 350 3330 101 30 OSDOS1 W
          MDISK 351 3330 1 20 UDISK1 W
```

A DOS/VS VIRTUAL MACHINE

```
USER DOSVUSER PASSWORD 512K
     ACCOUNT ACCTNO BIN4
     OPTION ECMODE
          CONSOLE 01F 3215
          SPOOL 012 3505
          SPOOL 013 3525
          SPOOL 002 3211
          LINK VMSYS 190 190 RR
          MDISK 191 3330 11 10 UDISKA WR RPASS WPASS
          MDISK 350 3330 100 50 VOSDOS W
          MDISK 351 3330  21 30 UDISK1 W
```

AN OS/MFT VIRTUAL MACHINE

```
USER OSUSER PASSWORD 512K
     ACCOUNT ACCTNO BIN5
     OPTION REALTIMER ISAM
          CONSOLE 01F 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          DEDICATE 230 OSRES
          DEDICATE 231 OSWRK
          DEDICATE 185 285
          DEDICATE 186 286
```

AN OS/VS VIRTUAL MACHINE

```
USER OSVUSER1 PASSWORD 512K
     ACCOUNT ACCTNO BIN6
     OPTION REALTIMER VIRT=REAL ECMODE
          CONSOLE 01F 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          SPOOL 012 3505
          SPOOL 002 3211
          LINK VMSYS 190 190 RR
          MDISK 191 3330  21  10 UDISKA WR RPASS WPASS
          MDISK 350 3330   0 100 VOSDOS MW
          MDISK 351 3330  51  50 UDISK1 W
```

ANOTHER OS/VS VIRTUAL MACHINE

```
USER OSVUSER2 PASSWORD 512K
     ACCOUNT ACCTNO BIN7
     OPTION REALTIMER ECMODE ISAM
          CONSOLE 01F 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 002 3211
          LINK VMSYS 190 190 RR
          MDISK 191 3330  31  10 UDISKA WR RPASS WPASS
          LINK OSVUSER1 350 350 MW
          MDISK 351 3330   0  50 UDISK3 W
```

A MULTI-ACCESS VIRTUAL MACHINE

The following directory entry represents a multi-access TSO system configured to handle up to 4 concurrent remote terminals and one local 3270. It has been given the VIRT=REAL option in order to improve response time.

```
USER TSOSYS PASSWORD 384K
     ACCOUNT ACCTNO BIN8
     OPTION REALTIMER VIRT=REAL
          CONSOLE 01F 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          DEDICATE 290 TSOSYS
          DEDICATE 291 TSOWRK
          SPECIAL 070 3270
          SPECIAL 080 2702 IBM
          SPECIAL 081 2702 IBM
          SPECIAL 082 2702 IBM
          SPECIAL 083 2702 TELE
```

OTHER SYSTEM VIRTUAL MACHINES

In addition to the virtual machines discussed to this point there are those virtual machines whose function is to:

- Support and update the VM/370 system.
- Test new releases of the system before placing them into production status.
- Provide the hardware service representative with the ability to run diagnostics, and extract recorded error data.
- Provide other users with a remote file spooling capability.

A VIRTUAL MACHINE FOR UPDATING AND SUPPORTING VM/370 (VMSYS)

The following directory entry defines a virtual machine (VMSYS) that can support and update the VM/370 system. The 194 minidisk contains alterations or fixes to CP after they have been thoroughly tested and considered stable. The 354 minidisk contains the distributed source files. The VMSYS virtual machine's privilege classes include class E and class G command privileges, so that it can issue the SAVESYS command to save CMS and other sytems. The 190 minidisk contains the CMS nucleus and all of the CMS modules and EXEC procedures available to all CMS users. Any virtual machine that wants to use the CMS system, links to this disk (190). The 353 minidisk contains the distributed CMS source code in an unaltered form. The 193 minidisk holds CMS PTFs and updates.

```
USER VMSYS PASSWORD 512K 16M BCEG
     ACCOUNT ACCTNO BIN9
     OPTION ECMODE REALTIMER
          CONSOLE 01F 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          SPOOL 002 3211
          MDISK 190 3330 035 110 CPV2L0 MR RPASS
          MDISK 191 3330 019 010 CPV2L0 MR RPASS
          MDISK 194 3330 145 058 CPV2L0 MR RPASS
          MDISK 199 3330 034 001 CPV2L0 WR RPASS
          MDISK 193 3330 001 050 USERD1 MR RPASS
          MDISK 294 3330 051 050 USERD1 MR RPASS
          MDISK 350 3330 101 003 USERD1 MW RPASS
          MDISK 353 3330 001 110 USERD2 MR RPASS
          MDISK 354 3330 001 110 USERD3 MR RPASS
```

A HARDWARE SERVICE VIRTUAL MACHINE (FESYS)

The following directory entry defines a virtual machine (FESYS) that is normally used by the hardware service representative. Usually, this is the only virtual machine that has command privilege class F. When a virtual machine with class F command privileges logs on, outboard recording is terminated. With class F command privileges, you can set the recording mode for a device, set the mode for recording soft errors, and execute the EREP program. The minidisk is used for diagnostics, CMS EXEC procedures, the CPEREP program, or any other programs used by the hardware support representative.

```
USER FESYS PASSWORD 256K 1M FG
     ACCOUNT ACCTNO BIN10
          CONSOLE 009 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          LINK VMSYS 190 190 RR
          MDISK 191 3330 151 10 UDISKA WR RPASS WPASS
```

A VM/370 VIRTUAL MACHINE (TESTSYS)

The directory entry for this virtual machine has the ECMODE and REALTIMER options, allowing its use to check a new CP nucleus before moving that nucleus to the floor system. TESTSYS contains two minidisks, 350 and 351. These disks are exact copies of the real system residence and scratch volumes. They are formatted and allocated so that the real system can spool and page on these disks. If you need additional disks, link to those disks before you load the virtual VM/370 system.

```
USER TESTSYS PASSWORD 512K
     ACCOUNT ACCTNO BIN11
          OPTION ECMODE REALTIMER
          CONSOLE 01F 3215
          SPOOL C 2540 R
          SPOOL D 2540 P
          SPOOL E 1403
          LINK VMSYS 190 190 RR
          MDISK 191 3330 161 10 UDISKA WR RPASS WPASS
          MDISK 350 3330   1 15 SYSWRK WR RPASS WPASS
          MDISK 351 3330  16 20 SYSWRK WR RPASS WPASS
```

THE REMOTE SPOOLING VIRTUAL MACHINE (RSCS)

The virtual machine defined by the following directory entry executes the Remote Spooling Communications Subsystem component of VM/370. The ECMODE option is required by the RSCS supervisor to permit the use of the clock comparator. Files to be transmitted are spooled to its virtual reader. Virtual punches and printers are dynamically defined and detached by RSCS, as required. The 190 virtual disk contains the RSCS nucleus, RSCS modules, and the line driver programs. The binary synchronous lines, dedicated to RSCS, are allocated and deallocated by RSCS as required to transmit files to and from remote stations.

```
USER RSCS PASSWORD 512K
     ACCOUNT ACCTNO BIN12
     OPTION ECMODE
          CONSOLE 01F 3215
          SPOOL C 2540 R
          MDISK 190 3330 020 005 CMSVL1 R
          DEDICATE 0B1 078
          DEDICATE 0B2 079
          DEDICATE 0B3 07A
```

Section 8. Generating Your Operating System under VM/370

Section 8 contains a general discussion on the major phases of system generation and suggests ways in which some steps in the procedure can take advantage of VM/370 facilities such as the CMS Editor and the CMS file facility. Probably, the most significant advantage to generating your system under VM/370 is the ability to do so without disturbing the normal every day production activity. The system programmer or whoever else is responsible for the operating system can log on to his own virtual machine and go through the generation steps at his own pace while the daily work is being processed. When the system has been brought up and fully tested, you can place it in operation, replacing the previous version, with a minimum of interruption.

Section 8 will also show examples of portions of the system generation procedure for DOS/VS and OS/VS1. Wherever feasible the examples will involve the use of VM/370 facilities.

Section 8 contains the following topics:

Generation Procedures Under VM/370

- Preparatory Phase
- Preliminary Phase
- Generation Phase
- Testing Phase
- How VM/370 Can Help

Generating DOS/VS Under VM/370

- Build System Generation Job Streams
- Copy the Distribution Tape to Disk
- Ready the Interim System
- Assemble and Load New Supervisor
- Add I/O Modules to System
- Final HOusekeeping

Generating OS/VS Under VM/370

- Preparing for OS/VS System Generation
- Stage I Processing
- Stage II Processing
- Final Housekeeping

The system generation of most operating systems can be thought of as occurring in four phases.

PREPARATORY PHASE

The preparatory phase involves the tailoring of the system generation macros to conform to your installation's requirements and preparing the job stream with which these macros will be assembled. Any other system generation job streams not involving intermediate results can also be set up at this time.

PRELIMINARY PHASE

The preliminary phase involves the initialization of the DASD volume to be used during system generation, the unloading of the distributed system to an initialized DASD volume, and the starting of the interim distributed system.

GENERATION PHASE

The generation phase includes the actual building of your tailored system. The system generation macros, prepared previously, are assembled into your new system nucleus. Utility, input and output, and system control routines are copied from the distribution libraries and placed in the appropriate libraries of the new system. Special input and output routines are assembled and also added to the system libraries. User-written routines can be copied over from the previous system.

TESTING PHASE

The testing phase involves the running of the IBM-supplied Installation Verification Procedure (IVP). The IVP checks that the newly created system is operational. It also verifies that the new System Control Program (SCP) supports your machine configuration. The jobs in the IVP test only the System Control Program. Program Products, programs with service classification "C" or similar programs added after the SCP has been generated should be tested via their own testing procedures or sample programs.

HOW VM/370 CAN HELP

VM/370 can be of considerable help in all four phases of system generation. The biggest advantage, available in all four phases, is the

ability to sandwich a normally lengthy generation procedure, that formerly had required a dedicated machine, into the regular day to day production schedule of an installation.

Job streams that are used during system generation can be created and updated using VM/370's CMS Editor; they can be stored as CMS files and passed over to the interim operating system via CMS EXEC procedures whenever required by the system generation process. For a detailed and fundamental discussion of the CMS Editor and EXEC facilities, see the VM/370: CMS User's Guide.

This subsection presents the major steps in a DOS/VS system generation procedure performed in a virtual machine running under VM/370. This virtual machine is assumed to have a VM/370 directory entry as shown in Figure 13. The link to CMSSYS provides you with the facilities of CMS which will be used whenever feasible. Alternate procedures will be discussed wherever applicable.

The virtual machine console is assumed to be a 3270 display terminal.

```
USER DOSVSYS PASSWORD 512K
 ACCOUNT ACCTNO BIN4
 OPTION ECMODE
  CONSOLE 009 3210
  SPOOL 00C 2540 R
  SPOOL 00D 2540 P
  SPOOL 00E 1403
  LINK CMSSYS 190 190 RR
  MDISK 191 3330 50 5 UDISK3 MR
  MDISK 250 3330 0 404 DOSSYS MR
```

Figure 13. Virtual Machine for DOS/VS System Generation.

In addition to the devices specified in the directory entry for user DOSVSYS, you will temporarily need a dedicated magnetic tape unit at x'180' on which to mount the distribution tape. The attachment of the magnetic tape drive is discussed in the section where it is first used.

BUILD SYSTEM GENERATION JOB STREAMS

During the entire system generation process you will be supplying the virtual machine with job streams tc perform the individual steps. On a standalone system, these job streams are usually kept in card form. Under VM/370, you can use the CMS Editor to create CMS files containing these job streams. You can then use the spooling functions of VM/370 to place these job streams into the card reader of your virtual machine whenever they are needed. This is especially helpful if your virtual console is at some distance from the real reader.

The first phase in a system generation for a tape and disk configuration is to unload the IBM-supplied base system from a distribution tape to an initialized DASD volume.

Create two short CMS files that contain the necessary card reader input data for the two utility programs that occupy the first two files on the distribution tape. The assumption is made that you have logged on to VM/370 as user DOSVSYS and that you are in the CMS environment, having issued the command:

```
ipl cms
```

The following sequence of commands and data entry is entered:

```
edit initload djcl                          (establish file ident.)
input                                       (start data entry)
// job intdsk                               ---------------
// assgn syslog,x'01f',c1                         ^
// date 12/31/75                                  |
// log                                            |
// assgn syslst,x'00e',l1                         |
// assgn sysopt,x'250',d4                         |
// exec                                         job
// uid iq                                     stream
// vtoc strtadr=(0403000),extent=(19)             |
vol1dossys                                        |
// end                                            |
// job disrst                                     |
// assgn sys005,x'250',d4                         |
// assgn sys006,x'180',t2                         |
// assgn syslst,x'00e',l1                         v
// exec                                     ---------------
(null line)                                 (terminates data entry)
file                                        (writes file on disk)
```

The job stream to initialize the disk and unload the tape is now a CMS file identified by a filename of INITLOAD and a filetype of DJCL.

Similarly, you can create CMS files containing job streams that will:

- Unload the IBM-supplied system to disk without initializing the disk
- Add I/O device specifications to the IBM-supplied supervisor
- Define standard labels
- Perform librarian functions
- Assemble and catalog a new supervisor
- Assemble and catalog additional I/O modules
- Back up the new system on tape or disk

COPY THE DISTRIBUTION SYSTEM TO DISK

Let us assume that there are two job streams available to us as CMS files. They are:

INITLOAD DJCL	initialize disk at X'250' and load from tape on x'180'
SKIPINIT DJCL	skip first file on tape and restore tape on x'180' to disk at x'250'

When you logged on to the VM/370 system with the userid DOSVSYS, your virtual machine had every I/O device it needed except for a magnetic tape unit. Since this device is only required during the tape-to-disk restore operation, it would be a waste of I/O resources to tie it up for the entire terminal session. Send the VM/370 system operator a message such as:

```
msg op pls mount dos/vs distribution tape as 180
```

The system operator will mount the appropriate tape reel on some idle tape drive, say 183, and issue the command:

```
attach 183 to dosvsys as 180
```

You will receive an acknowledgement:

```
TAPE 180 ATTACHED
```

You are now ready to start your system generation. If you are not in the CMS environment, issue the command:

```
ipl cms
```

Spool the punch to your virtual reader:

```
cp spool punch to *
```

If the disk is to be initialized, punch out the CMS file:

```
punch initload djcl (noheader)
```

The NOHEADER option eliminates the header record normally punched out to identify a card deck.

After a PUNCH command, you will receive an acknowledgement that the file was spooled to your reader.

If the disk had already been initialized, you would have punched out the file:

```
LOADONLY DJCL
```

Now that the job stream is in your virtual reader, you can perform an IPL from the tape. Issue the CP command:

```
cp ipl 180
```

and when the system goes into a wait state, press the PA1 Key or equivalent to place you in CP console mode. Issue the CP command READY 00c to ready your virtual reader and then BEGIN to return control to the system tape.

Your virtual machine is now under the control of the utility programs on tape and you should reply to all queries as you would if you were on a real machine.

When the distributed DOS/VS system has been restored to disk, you will receive a RESTORE complete and an END OF JOB message; your virtual machine will be placed in a disabled wait state and your console is put in to CP mode. You can now perform an IPL from x'250' and use the distributed system to perform the rest of the system generation procedures.

READY THE INTERIM SYSTEM

The IBM-supplied supervisor does not contain any I/O device specifications. Until you generate a new supervisor with your particular I/O configuration, you must include the appropriate ADD and ASSGN commands each time you perform an IPL of the DOS/VS system.

The following job stream files can be created for use in this and other phases of system generation:

```
IPLADD   DJCL  add and assign I/O devices
STDLABEL DJCL  assign standard labels
DSERV    DJCL  display the directories
PCHLNKED DJCL  punch out Z.LINKEDIT book
PCHDELET DJCL  punch out Z.DELETExx book
```

To start the system, assign standard labels, display the directories, and punch out the LINKEDIT and DELETExx books, enter the following sequence of commands:

```
cp spool punch to *
punch ipladd djcl (noheader)
punch stdlabel djcl (noheader)
punch dserv djcl (noheader)
punch pchlnked djcl (noheader)
punch pchdelet djcl (noheader)
cp ipl 250 clear
```

The last command loads the DOS/VS supervisor from unit 250. When the system goes into a wait state, ready your reader as follows:

```
(press PA1 Key or equivalent)
```

then enter:

```
ready 00c
begin
```

As each job ends, press the ENTER or RETURN key, depending on the type of terminal. The next job will then start.

When you receive the end-of-job message:

```
EOJ PCHDELET
```

do not press the ENTER or RETURN key. This would cause DOS/VS to read in the card deck that it just punched out; remember that the punch is still spooled to the reader. Rather, enter the command:

```
#cp ipl cms
```

to get you into the CMS environment. If you now enter:

```
read zlib books
```

the books that you punched out in the jobs PCHLNKED and PNCHDELET are now read from the virtual reader into a CMS file labeled ZLIB BOOKS. You can use this file to develop other job streams link-edit relocatable modules to your care image library and to trim your system of unwanted members of the core image, relocatable and source libraries.

ASSEMBLE AND LOAD NEW SUPERVISOR

The job stream to assemble your installation tailored supervisor is another candidate for a CMS file. In addition to the storage advantage, there is the ease with which you can assemble different versions of your basic supervisor.

For example, if you are planning to use the Supervisor Select function, enter your basic supervisor macro assembly job stream in a file called ASMBLSU1 DJCL. To generate another job stream to assemble a different version you would load CMS and enter:

```
edit asmblsu1 djcl
```

to access the basic supervisor memo file for a CMS EDIT session. You would then use the CHANGE and INPUT subcommands to alter the supervisor macros and the job name. Enter or change the SUPVR macro to include ID=2; this will identify your new version of the supervisor as $$A$SUP2.

When all the changes have been made, you can file the new version under its own name by issuing:

```
file asmblsu2 djcl
```

You now have two job streams on file, the original and the new version. In like manner, you can generate job streams to assemble any number of versions.

In order to assemble these two supervisors you must punch the appropriate job stream files to the virtual card reader. While you are still in the CMS environment, issue the following sequence of commands:

```
cp spool punch to *
punch ipladd djcl (noheader)
punch stdlabel djcl (noheader)
punch asmblsu1 djcl (noheader)
punch asmblsu2 djcl (noheader)
cp ipl 250 clear
```

At this point you may want to take advantage of the CMS Editor to check your program listings for errors without printing out a hard copy. You can do this by spooling your printer output to your virtual reader, returning to the CMS environment, creating a CMS file from the reader data, and using the Edit function to scan the listing.

Include the CP command:

```
cp spool printer to *
```

in the previous sequence of commands, before you load the DOS/VS system. This will direct printer output to your virtual reader. Your punch is already spooled to the reader.

If you are processing more than one job and want the output of each job to be kept separate, you must close the printer and/or punch at the end of each job. When you receive the EOJ ASMBLSU1 message and the virtual machine stops on an ATTN 00C, you can close both the punch and printer files and also assign them spool filenames and filetypes by issuing the commands:

```
#cp close punch name asmblsu1 deck
#cp close printer name asmblsu1 list
```

You can then enter a null line to start the next job. While spool files will be automatically closed when you issue an IPL command, you may want to use the CLOSE command on the last job in order to give the files a name.

Enter:

```
#cp ipl cms
```

to return to the CMS environment. Use the CMS READCARD command to generate CMS files from your reader files. Each READCARD command converts the first file in your reader to a CMS file with the filename and filetype you specify in the READCARD command line. That file is then deleted from your reader queue. If you are not sure of which file is first in your reader, query the reader with:

```
cp query reader all
```

The response will itemize all the files in your reader, in sequence, and include such information as file identification number, printer or punch file, and filename.

If the response to your QUERY command was:

```
... 1502 ... PUN ... ASMBLSU1  DECK
... 1503 ... PRT ... ASMBLSU1  LIST
... 1541 ... PUN ... ASMBLSU2  DECK
... 1542 ... PRT ... ASMBLSU2  LIST
```

and you then issued:

```
readcard supvsr1 text
```

CMS will read file number 1502 and create a CMS file with a filename of SUPVSR1 and a filetype of TEXT.

If, however, you want to first check the listing for errors, you can issue:

```
cp order reader 1503
```

which will reorder your reader queue to bring the listing file first. Then issue:

```
readcard supvsr listing
edit supvsr listing
```

to create a CMS file containing the assembly listing and to access it for the Edit function. Using the Edit subcommands, you can now scan the listing for errors. If there are no errors, you can create a CMS file, SUPVSR TEXT, from file 1502, which is now the first file in the reader queue. If there are errors, you can issue:

```
erase supvsr listing
```

to discard the assembly listing, and

```
cp purge reader 1502
```

to remove the corresponding assembly text deck.

You can now use the CMS Editor to correct the supervisor macros and pass the jobstream over to DOS/VS for another assembly.

When you have a supervisor text deck reader to catalog, you can insert it into a LINKEDIT DJCL job stream that you created from the CMS file, ZLIB BOOKS previously punched out. To insert the supervisor text deck issue the following:

```
(from the CMS environment)
edit linkedit djcl     (access the job stream)
next                   (position current line pointer)
delete 2               (remove catals and bkend entries)
locate /include        (position current line pointer)
getfile supvsr1 text   (insert supervisor text deck)
file                   (replace updated jobstream)
```

The updated LINKEDIT job stream can now be passed over to DOS/VS for execution. The portion of the LINKEDIT file following the new supervisor deck provides for the optional linkage editing of the IBM-supplied library and service routines. You can execute this phase interactively as on a real machine, or again, you can use the CMS Editor to tailor the LINKEDIT job stream, removing the individual jobs not required and removing the PAUSE statements from the jobs that are to be run.

The assembling and cataloging of any additional IOCS modules can also be performed with the help of the CMS Editor. In the CMS environment, create a job stream file containing all the IOCS macros to be assembled and ensure that each macro statement includes the SEPASMB=YES entry. The following series of commands can be entered:

```
cp spool punch to *
cp purge reader all
punch iocsmods djcl
cp spool punch hold
cp spool printer to *
cp ipl 250 clear
```

After sending the job stream to the virtual reader, invoke the HOLD option for the punch and spool the printer to the reader. When the assemblies have been completed, reload CMS and examine the listings to check for any errors.

```
#cp ipl cms
cp spool printer off
cp spool punch nohold
readcard iocsmods listing
edit iocsmods listing
     .
     .   (use EDIT subcommands to scan listing)
     .
     .
     .
quit
```

If there are errors, use the Editor to correct the macro entries and resubmit the job stream. If there are no errors, you can enter:

```
print iocsmods listing
```

to get a hard copy of the assembly listings.

Build a job stream to catalog the IOCS text decks to the system relocatable library:

```
cp change reader all nohold
readcard iocsmods text
edit catalrio djcl
input // job catalrlb
input // exec maint
getfile iocsmods text
input /*
input /&
file
```

FINAL HOUSEKEEPING

Now that the basic system generation is complete, a number of housekeeping procedures can be set up as job streams. Many of these procedures are executed regularly. Consider job streams to perform the following:

- Set new standard labels
- Reallocate library sizes
- Condense libraries
- Set automatic condense limits
- Copy the system residence pack for backup (also private libraries)
- Catalog often-used job control and linkage editor control statements to the procedure library.

The matter of cataloging job control statements and the like to the procedure library brings out another use of the CMS Editor. You can initially create the job streams and pass them over to DOS/VS for testing in the same way as you did for the system generation procedures. When they have been checked out, you can use one job stream to catalog the other procedures on to your system's procedure library

This subsection discusses the major steps in an OS/VS system generation performed in a virtual machine running under VM/370. While the procedures outlined below are taken from an OS/VS1 system generation, they are similar enough to OS/VS2 procedures to allow their use as examples. The primary difference between the generation of OS/VS1 and OS/VS2 is the VM/VS Handshaking feature for OS/VS1. This feature requires that the VM option in the SCHEDULR macro be specified.

In the previous subsection, the "flip flop" technique was used in the generation of DOS/VS. This involves using one virtual machine and multiple alternating systems. While this was convenient, only one system could be executing at any one time. OS/VS generation runs are much longer and methods of improving the overall through-put should be explored.

The generation of OS/VS under VM/370 will be performed assuming the existence of two virtual machines. Figure 14 shows the directory entries for these two machines. In addition to the devices specified in the directory entry for user OSVSSYS, you will temporarily need a dedicated magnetic tape unit at virtual address X'181' on which to mount the starter system and distribution library tapes. The attachment of the magnetic tape drive is discussed where first used. The OSCMS virtual machine is used to prepare job streams for the OSVSSYS virtual machine. It is also used to scan output of the OS/VS machine. Some steps in the system generation can be performed entirely in the CMS machine.

The job streams that will be used in the system generation runs will be created, stored, and updated via the CMS Editor. The VM/370 spool system and CP spool file commands will be used to load the appropriate combination of of job streams into the virtual card reader of the OS/VS machine at the various stages of generation.

PREPARING FOR OS/VS SYSTEM GENERATION

Before the starter system and distribution system libraries can be used for system generation, certain procedures must be performed. These procedures include:

- Initialization of the volumes (DLIBA1 and DLIBA2) that are to contain the starter system and distribution libraries.
- Restoring the starter system from tape to a DASD volume (DLIBA1).
- Using the starter system to copy the distribution libraries from the tape volumes (DLIBT1 and DLIBT2) to a DASD volume (DLIBA2).
- Miscellaneous other jobs such as creating standalone card decks for independent utilities, listing the VTOC (Volume Table of Contents) of each current system volume for a possible re-allocation run, and initializing the volume which will contain the new OS/VS system.

```
USER OSVSSYS PASSWORD 768K
  ACCOUNT ACCTNO BIN16
  OPTION ECMODE REALTIMER
    CONSOLE 01F 3210
    SPOOL 00C 2540 R
    SPOOL 00D 2540 P
    SPOOL 00E 1403
    MDISK 350 3330 0 404 DLIBA1 MR RPASS WPASS
    MDISK 351 3330 0 404 DLIBA2 MR RPASS WPASS
    MDISK 250 3330 0 404 OSVSYS MR RPASS WPASS

USER OSCMS PASSWORD 320K
  ACCOUNT ACCTNO BIN16
    CONSOLE 01F 3210
    SPOOL 00C 2540 P
    SPOOL 00D 2540 P
    SPOOL 00E 1403
    LINK CMSSYS 190 190 RR
    MDISK 191 3330 50 10 UDISK1 WR RPASS
```

Figure 14. Virtual Machines for OS/VS System Generation

ONE OR TWO TERMINALS

For the following examples, it is assumed that the two virtual machines whose directory entries were shown in Figure 14 each have their own 3270 terminals. If two terminals are available, this is the most efficient method. Each system (CMS and OS/VS) runs independently with full terminal access.

You can, however, run the same procedures using a single terminal. If, for example, you want to access the OSVSSYS virtual machine, you place the virtual machine you are on (OSCMS) in disconnect (DISCONN) mode and log on to OSVSSYS. The disconnected virtual machine, OSCMS, continues to run. When you are ready to swap virtual machines again, you disconnect from the OSVSSYS machine and log back on to OSCMS. The act of logging back on to a disconnected virtual machine stops its execution and you must enter the BEGIN command to resume operation.

If you use the disconnect method of running two virtual machines from one terminal, you must be aware of at least two precautions:

- When a virtual machine is running disconnected, all console output is lost unless you initiate console spooling:

  ```
  #cp spool console start
  ```

  Spooling of the console output will continue until you either issue:

  ```
  #cp spool console stop
  ```

  or you log off. Disconnecting the virtual machine will not stop console spooling. Therefore, the spooled console log for a terminal session punctuated with several disconnects will consist of one uninterrupted printer file.

- When a virtual machine is running disconnected, a fifteen-minute time-out period is initiated if an attempt is made to read from the terminal or if the machine goes into a disabled wait state. At the

end of that time, the virtual machine is automatically logged off. However, if you log on before the 15 minutes is up, the virtual machine is not logged off.

INITIALIZING THE STARTER SYSTEM VOLUME

The utility control statements that you supply in the form of card input on a standalone system can be generated as CMS files in a VM/370 environment. The CMS Editor and EXEC facilities, along with the VM/370 spool file system, allow you to create, update and store entire job streams or individual job steps. Data as well as job control language statements can be included in these CMS files.

As an example, to create the reader input required by the IBCDASDI utility program for the initialization of the DASD volume to contain the starter system, log on to your OSCMS virtual machine, load CMS, and enter the following:

```
edit initstrt ojcl                          (open and identify new file)
input                                       (enter input mode)
initvol1 job
   msg todev=3210,toaddr=01f
   dadef todev=3330,toaddr=350,volid=scratch,passes=0
   vld newvolid=dliba1,ownerid=sysprogmr
   vtocd strtadr=14,extent=5
   end
(null line)                                 (to return to edit mode)
file
```

The control statements for the IBCDASDI utility program now reside in a CMS file, identified as INITSTRT OJCL and belonging to the OSCMS virtual machine.

To transfer a copy of this file to the virtual reader of the OSVSSYS virtual machine, enter the following:

```
cp spool punch to osvssys
punch initstrt ojcl (noheader)
```

The punched file is now queued in the virtual reader of user OSVSSYS, whether OSVSSYS is logged on or not. If OSVSSYS is logged on and does not have messages disabled (SET MSG OFF), a message at the terminal will notify him of the addition to his reader queue. If OSVSSYS is not logged on, notification of files pending in the virtual reader is displayed as part of the LOGON messages received by the user when he eventually logs on.

To perform the initialization, use a second terminal and log on to userid OSVSSYS. Send the following messages to the system operator:

```
msg op pls mount scratch 3330 pack as 350
msg op pls mount osvs1 starter tape as 181
```

The system operator will mount a 3330 pack on some available spindle, say 150, and enter:

```
attach 150 to osvssys as 350
```

Similarly, mounting the OS/VS1 starter tape on an available tape drive, say 471, he enters:

```
attach 471 to osvssys as 181
```

You will receive the following two messages, confirming the operator's action:

```
DASD 350 ATTACHED
TAPE 181 ATTACHED
```

From the OSVSSYS terminal you can now enter the CP commands:

```
ready 181
ipl 181
```

to load the IBCDASDI program, which is the first file on the starter tape. When IBCDASDI is loaded, the program goes into a wait state. To check for this, display the PSW by issuing:

```
#cp display psw
```

To define the control statement input device (the virtual reader) to the IBCDASDI program, press the ENTER key (or its equivalent), and the message:

```
IBC105A DEFINE INPUT DEVICE
```

will be displayed. Reply, in upper case:

```
INPUT=2540,00C
```

When the volume initialization is complete, you will receive the message:

```
IBC163A END OF JOB
```

at your terminal and the system enters the wait state.

RESTORING THE STARTER SYSTEM TO DISK

In a similar manner, at the OSCMS terminal, you can create a CMS file containing the control statements for the IBCDMPRS program and punch it to the virtual reader of the OSVSSYS virtual machine. The starter tape is now positioned at the end of the first file and ready to load the second file, the IBCDMPRS program. Use the OSVSSYS terminal to enter the command:

```
ipl 181
```

Follow the same procedure as in the volume initialization step to define the control statement input device to the program. When you receive the END OF JOB message, the starter system will have been restored to the DASD volume and can now be made operative.

LOADING THE STARTER SYSTEM

In order to start your interim starter system, enter the following from the OSVSSYS terminal:

```
ipl 350 clear
```

When you receive the message:

```
IEA760A SPECIFY VIRTUAL STORAGE SIZE
```

enter a null line if the size of your virtual machine is less then 384K. For the size machine used in this discussion (768K) you would enter:

```
R 00,'2048'
```

Continue to reply to the prompting messages as directed in the appropriate system generation manual.

When you receive the message:

```
IEE048I INITIALIZATION COMPLETED
```

you can ready the starter system to process input. Enter the following commands:

```
MN JOBNAMES,T
START RDR,00C
START WTR,00E
START INIT.P0,,,A
```

RESTORING THE DISTRIBUTION LIBRARIES TO DISK

Ask the system operator to mount a scratch pack as your virtual 351:

```
#cp msg op pls mount scratch 3330 pack as 351
```

At the OSCMS terminal, you can create the IEHDASDR control statement file for initializing the volume to contain the distribution libraries. You can then punch the file to the OSVSSYS machine for execution. Before executing the IEHDASDR utility program, vary the DASD volume offline.

When you receive the completion message:

```
IEH806I ANALYSIS OF DDNAME=SYSIN IS COMPLETE.
        VOLUME SERIAL NO.=DLIBA2
```

vary the initialized volume back online.

The IEBCOPY utility program is used to unload the distribution libraries from tape to disk. The IEBCOPY control statements are in the form of load procedures and are contained in the first file of each distribution library tape.

Have the system operator mount the distribution library tapes and follow the procedures in the appropriate system generation manual to unload the distribution libraries to either one or two DASD volumes, depending on their capacity.

OTHER MISCELLANEOUS PREPARATORY STEPS

If you decide to punch out the independent utilities and IPL text, you may want to consider storing the card images in your OSCMS virtual machine.

Before you run the IEBPTPCH program, issue the CP command:

```
#cp spool punch to oscms
```

to send the punched output to the virtual reader of the OSCMS machine. When the IEBPTPCH program has completed, enter the CP command:

```
#cp close punch
```

to release the file to the OSCMS virtual machine. At the OSCMS console you can now enter:

```
readcard utility output
```

to create a CMS file containing all the card images punched. Using the CMS Editor you can create separate files for each utility, add appropriate control statements, and file them for future use.

On the other hand, you should also consider that VM/370 itself has utility and service programs which will perform functions practically identical to the OS/VS independent utilities.

The VM/370 version of IBCDASDI is available to you on the system disk under the filename of IPL and a filetype of IBCDASDI. An added feature of the VM/370 version is its ability to initialize virtual disks (minidisks). For a description and operating procedures, see the VM/370: Planning and System Generation Guide.

The function of IBCDMPRS can be obtained by running VM/370's DDR service program in standalone mode. For a description and operating procedures, see the VM/370: Operator's Guide.

The functions of ICAPRTBL are available under VM/370 via the CP commands LOADBUF and LOADVFCB.

The class D command LOADBUF can load the Universal Character Set Buffer (UCSB) or Forms Control Buffer (FCB) for the real 3211. See the VM/370: Operator's Guide for usage information. Also, the class G command LOADVFCB can be used to load a Forms Control Buffer (FCB) for a virtual 3211. See the VM/370: CP Command Reference for General Users.

You can perform the initialization of the volume that will be your new system residence pack from the OSCMS machine while you start the Stage 1 processing on the OSVSSYS machine.

Use the CMS Editor to create an IBCDASDI control statement file and identify it as IBCDASDI CTLS. Use the EDIT subcommand GETFILE to insert the IPL TEXT file previously punched from the distribution library volume. You can then perform the initialization by entering:

```
cp spool punch to * cont
punch ipl ibcdasdi (noheader)
punch ibcdasdi ctls (noheader)
cp spool punch nocont
cp close punch
cp ipl 00c
```

and replying to the prompting messages.

STAGE I PROCESSING

Stage I processing consists of creating the Stage I input job stream (job control statements and system generation macros), executing a

single assembly, and checking the assembly listing for any errors. If errors are detected, the Stage I procedure is repeated.

PREPARING STAGE I INPUT

Here again, the facilities of the CMS Editor can be utilized to create and storage the Stage I input. The job stream can then be punched to the OSVSSYS machine's virtual reader to be assembled under the OS/VS starter system. You also have the capability of editing the job stream, making changes to it, and filing the modified copy under a different name.

The Stage I assembly can actually be executed in either the OS/VS machine or in the CMS machine, whichever is more convenient.

To perform the assembly in the CMS virtual machine, there are two requirements to be considered:

1. Your CMS virtual machine, OSCMS, must have access to the supervisor macros (SYS1.AGENLIB) that reside on volume DLIBA2, owned by the OSVSSYS virtual machine. This can be accomplished by entering the following commands from the OSCMS terminal:

   ```
   cp link osvssys 351 351 rr
   acc 351 b
   filedef cmslib disk osgenlib maclib b dsn sys1 agenlib
   global maclib osgenlib
   ```

   The above commands can be placed into a CMS EXEC and invoked by a single command. For details on setting up EXEC procedures, see "Part 3. The EXEC Language" in the VM/370: CMS User's Guide.

2. The CMS Assembler requires, for its input, a CMS file containing only source statements and having a filetype of ASSEMBLE. The job stream previously discussed contained both job control statements and source macro statements. The following procedure will allow you to assemble Stage I in either virtual machine.

   Create one CMS file containing the job control statements that precede the system generation macros and file it with some identification such as STG1IN OJCL. Create another CMS file containing the system generation macros followed by an END statement and ending with */ card. File this with some filename such as STG1IN and a required filetype of ASSEMBLE. The filetype identifies it as an Assembler source statement file.

STAGE I EXECUTION IN THE CMS VIRTUAL MACHINE

With both of the above requirements met, you can assemble your Stage I input in your CMS virtual machine by issuing the CMS command:

```
assemble stg1in
```

The output of the assembly will be two CMS files:

```
STG1IN TEXT     (punch output)
STG1IN LISTING  (printer output)
```

The punched output is the job stream for Stage II input. The printed output documents the expansion of each specified macro instruction including the punch statements that comprise the input to Stage II.

When assembling under CMS, assembly errors are displayed on the terminal and you need not wait for the listing to be printed. Your source file can be corrected via the CMS Editor and the assembly re-tried. When you finally have a clean assembly, the two output files will reflect the final assembly and can then be printed, punched, or used to create job streams.

STAGE I EXECUTION IN THE OS/VS VIRTUAL MACHINE

You can punch the required job stream for Stage I execution to the OS/VS machine's virtual card reader with the following commands:

```
cp spool punch to osvssys cont
punch stg1in ojcl (noheader)
punch stg1in assemble (noheader)
cp spool punch nocont
cp close punch
```

The above use of the CONT option concatenates multiple CMS files into one punch output file.

If, at your OS/VS machine, you enter the CP command:

```
#cp spool punch to oscms
```

before you execute Stage I; and subsequently enter:

```
#cp close punch
```

the Stage I output will be transferred to the OSCMS virtual card reader. At the OSCMS terminal the command:

```
readcard stg1in text
```

will create a CMS file that corresponds to the punched output of the Stage I assembly performed in the OSCMS machine (STG1IN TEXT). The assembly listing, in the absence of any printer spooling changes, will be directed to the real printer for output.

STAGE II PROCESSING

The output of Stage I is a job stream made up of a number of individual jobs to set up system data sets, assemble nucleus modules, and copy modules from the distribution libraries to the new system residence volume.

PREPARING STAGE II INPUT

If the three utility data sets that are used by the assembler during Stage II have not been allocated and cataloged, you can create the appropriate job stream as a CMS file with some identification such as STG2IN DSALLOC.

To transfer the Stage II input to the OS/VS machine, issue the commands:

```
cp spool punch to osvssys cont
punch stg2in dsalloc (noheader)
punch stg1in text (noheader)
cp spool punch nocont
cp close punch
```

STAGE II EXECUTION

Process the Stage II input job stream on the OSVSSYS virtual machine. To enhance the performance of your virtual machine, you can take advantage of the various VM/370 performance options such as running in the virtual=real area, and favored execution with a guaranteed percentage of CPU time.

While the OSCMS virtual machine is not directly involved in the Stage II processing, you may want to use it to subdivide the Stage I output into individual job streams. This may make it easier for you in the event that restarting of Stage II becomes necessary.

FINAL HOUSEKEEPING

The new system residence volume now contains the new System Control Program. Program Products are added in separate runs; the procedures are contained in the appropriate Program Product manuals. The OSCMS virtual machine can be used to set up or modify job streams used to test programs that do not come under the scope of the Installation Verification Procedure supplied with the System Control Program.

Section 9. Special Considerations

Section 9 discusses those aspects of running an operating system under VM/370 that apply only in certain unique situations or under certain operating conditions. This section should be considered as an extension to "Section 2. General Considerations" and an adjunct to "Section 6. System Planning Considerations."

Section 9 contains the following topics:

Defining Multiple Consoles

VM/VS Handshaking

Using The Diagnose Interface To CP

Multiple-Access Virtual Machines

The ASP Virtual Machine

Channel Switching

DASD Reserve/Release

CP allows you to specify only one console as such in the VM/370 directory via the CONSOLE statement. That console is the one you are logically connected to when you use a VM/370-supported terminal or console to log onto VM/370 and IPL the operating system that is to run in your virtual machine. That console is supported only in 3215/1052 mode, so if the operating system is to use that console as its primary console, the operating system must have been generated with the console specified as a 3215, 3210, or 1052, regardless of whether the actual device is something else (such as a 3270, 3158 or 3066 display console, 2741 or 3767 terminal).

HOW TO SPECIFY MORE THAN ONE CONSOLE FOR A VIRTUAL MACHINE

If your operating system can support multiple consoles (where different classes of system messages can be routed to different consoles), or alternate consoles (where you can switch to a back-up console when the primary console becomes inoperative), you must tell CP about the existence of these additional consoles. There are several ways that this can be accomplished, either via directory statements or by issuing CP commands. The way you specify the second console also depends on whether you always want to use a specific device at a specific I/O address, or whether you want more flexibility in selecting which device or terminal is to be used.

USING A SPECIFIC DEVICE AS ANOTHER CONSOLE BY THE VIRTUAL MACHINE

Assume in this example that a DOS/VS virtual machine (DOSVUSER) is to run on a System/370 Model 158 with its display console at address 01F and a local 3270 terminal at address 1B8.

If DOS/VS wants to use the 3158 display console in Display Operator Console (DOC) mode, then that console cannot be defined in the VM/370 CONSOLE directory statement since CP only supports simulated 3215/3210/1052 devices as the virtual machine console. DOS/VS must be generated with at least two consoles specified, let us say 01F as its primary console, and some other address, such as 009, for the second console. The second console must be specified as a 3210, 3215, or 1052 to DOS/VS even though a 3270 terminal is actually used, since it will appear to DOS/VS to be a 3210/3215/1052 device.

Therefore, the VM/370 directory CONSOLE statement would be:

```
CONSOLE 009 3210
```

DOS/VS's primary console can be specified either by a DEDICATE statement in the VM/370 directory or via the CP ATTACH command after you log on. The format of the DEDICATE statement, if used, would be similar to the following:

```
DEDICATE 01F 01F
```

Instead of having the DEDICATE statement, you could send a message to the VM/370 operator after you log on to VM/370, requesting that he issue the following ATTACH command:

```
attach 01f to userid as 01f
```

In either case, DOS/VS would then be able to use the 3158 console in Display Operator Console mode.

HAVING A FLEXIBLE CHOICE OF DEVICES TO USE AS A SECOND CONSOLE

If you want to be able to use any locally attached 3270 display terminal as the DOS/VS primary console in Display Operator Console mode, you can either have a SPECIAL statement in your VM/370 directory, or, after you log on to VM/370, you can issue the CP command DEFINE.

If the SPECIAL statement is used, it would appear as follows:

```
SPECIAL 01F 3270
```

If the SPECIAL statement was not used, assuming that a local 3270 line had been enabled by the VM/370 operator, you could issue the following DEFINE command:

```
define graf 01f 3270
```

Regardless of whether the SPECIAL statement or the DEFINE command was used, after logging on to VM/370 via the device specified in the CONSOLE statement and loading the DOS/VS system into the virtual machine, you must issue the CP command DIAL at the local 3270 that you wish to use in display mode to logically connect that 3270 and its real I/O address of 1B8 to the operating system. Note: A remote 3270 cannot be used in this manner since the DIAL command does not support remote 3270 terminals.

If the second console was a remote terminal such as a 2741 or 3767 connected via a 2702 or a 3704/3705 in 2702 emulation mode, the SPECIAL statement would look like the following:

```
SPECIAL 01F 2702 IBM
```

or the DEFINE command would be:

```
define line as 01f ibm
```

VM/VS Handshaking is a communication path between the VM/370 Control Program and a virtual machine operating system (OS/VS1) that makes each system control program aware of any capabilities or requirements of the other. VM/VS Handshaking consists of:

- Closing CP spool files when VS1 job output from its DSO, terminator, and output writer is completed
- Processing VS1 pseudo page faults
- Providing an optional nonpaging mode for VS1 when it is run under the control of VM/370
- Providing miscellaneous enhancements for VS1 when it is run under the control of VM/370

The handshaking feature improves the operational characteristics of VS1 with VM/370 and yet allows the same VS1 operating system to run without change in either (1) a real machine or (2) a virtual machine under the control of VM/370. When the VM/VS Handshaking feature is active, the operation of VS1 with VM/370 more closely resembles the standalone operation of VS1. There is less need for virtual machine operator intervention because VS1 closes its CP spool files so they can be processed by VM/370 when the job output from the VS1 DSO, terminator, and output writer is complete. Also, one VS1 task can be dispatched while another is waiting for a page to be brought into real storage if the pseudo page fault handling portion of handshaking is active. With nonpaging mode, duplicate paging can be eliminated.

ACTIVATING VM/VS HANDSHAKING

Although handshaking is a system generation feature for VS1, it is active only when VS1 is run under the control of VM/370; it is disabled when that same VS1 operating system is run on a real machine. Storage for the VS/1 virtual machine should be no larger than 4M. The VM/VS Handshaking feature is not active unless:

1. VS1 is generated with the VM/370 option;
2. VS1 is run under the control of a version of VM/370 that supports the feature (VM/370 supports handshaking beginning with Release 2 PLC 13);
3. The virtual machine size is at least one megabyte, but not greater than four megabytes; and
4. If non-paging mode is to be used, the virtual storage size (a VS1 parameter) must be equal to the virtual machine size.

The pseudo page fault portion of the handshaking feature is not active unless it is set on. It can be set on, and later set off, with the CP SET PAGEX command line. If you use the CP IPL command again, PAGEX is turned off.

When a VS1 virtual machine with the handshaking feature is loaded, its initialization routines determine whether the handshaking feature should be enabled or not. First, VS1 checks to see if it is running under the control of VM/370 by issuing an STIDP (Store Processor ID) instruction. STIDP returns a version code; a version code of X'FF' indicates VS1 is running under VM/370.

If VS1 finds a version code of X'FF', it then issues a DIAGNOSE code X'00' instruction to store the VM/370 extended-identification code. If an extended-identification code is returned to VS1, VS1 knows that VM/370 supports handshaking; if nothing is returned to VS1, VM/370 does not support handshaking. At this point in the VS1 initialization process, VM/VS Handshaking support is available. If VS1 is running in the nonpaging mode and if the virtual machine operator issues the CP SET PAGEX ON command, full VM/VS Handshaking support is available.

Provided that the above conditions are satisfied when VS1 is loaded under control of VM/370, VS1 will issue a special message after message IEA760A specify virtual storage size:

IEA788I NON-PAGING MODE OF VS UNDER VM/370

Special considerations which are unique to the non-paging mode should be examined prior to implementation of this mode:

- An "EOB" or "U" response to VS1 message IEA760A will automatically set the virtual storage size equal to the virtual machine size, provided the latter is one megabyte or larger.
- A VS1 Release 4 or earlier system which requires more than four megabytes of storage cannot function in either a virtual or real machine. A VS1 Release 5 system, however, will use only four megabytes and issues a message to that effect.
- A minimum VS1 nucleus will tend to be more suitable for the non-paging mode because more space will be available for problem program partitions.
- A large VS1 nucleus will tend to be more restricted by lack of partition space in the non-paging mode.

In the non-paging mode, virtual storage is mapped in normal fashion such that the JES buffer pool, VTAM workspace, RTAM area, JES routines, resident modules, and the pageable supervisor are at the top of virtual storage. Minimizing these functions in terms of virtual storage requirements will maximize problem program partition space.

- The VS1 paging dataset is not utilized in the non-paging mode. Page parameters and/or page packs are not required for the VS1 system.
- It is possible to execute V=R jobs in the non-paging mode as the V=R line is a valid IPL parameter and the V=R logic within VS1 is applicable. Any benefits from such operation would appear to be questionable, as the entire VS1 system functions in V=R mode in the non-paging mode.
- "FASTNIP" will automatically force non-paging mode if the virtual machine size is 1 megabyte or greater.

It is not required by the handshaking function to invoke the non-paging mode, and there may be many occasions to run a VS1 system in a virtual machine size of one megabyte or larger (4 megabyte maximum) without utilizing this mode.

To avoid initiation of the non-paging mode, specify the virtual storage size explicitly to VS1 in response to message IEA760A: for example, R00, '6144'.

The following brief examples of VS1 IPL in a VM/370 environment (with handshaking active) may better illustrate initiation of non-paging mode.

Example 1:

VM Size = 768K
Sysgen Virt Stg Size = 6 M
IEA706A Response = "EOB"

Non-paging mode not initiated; VM size less than 1 M.

Example 2:

VM Size = 1 M
Sysgen Virt Stg Size = 6 M
IEA760A Response = "U"

Non-paging mode initiated, Virt Stg Size forced to 1 M by VS1. VS1 system probably will not complete initialization because of insufficient virtual storage.

Example 3:

VM Size = 4 M
Sysgen Virt Stg Size = 4 M
Response to IEA760A = R00,'4096'

Non-paging mode initiated. Explicit response happens to equal VM size.

Example 4:
VM Size = 3 M
Response to IEAA760A = R00,'6144'

Non-paging mode not initiated. Explicit response sets virtual storage size to 6 M. VS1 will require 6 M of paging space, however.

CLOSING CP SPOOL FILES

When the handshaking feature is active, VS1 closes the CP spool files when the job output from the VS1 DSO, terminator, and output writer is complete. Once the spool files are closed, they are processed by VM/370 and sent to the real printer or punch. With the VM/VS Handshaking feature, virtual machine operator intervention is not required to close CP spool files.

During its job output termination processing, VS1 issues DIAGNOSE code X'08' instructions to pass the CP CLOSE command to VM/370 for each CP spool file.

PSEUDO PAGE FAULTS

A page fault is a program interrupt that occurs when a page that is marked "not in storage" is referred to by an instruction within an

active page. The virtual machine operating system referring to the page is placed in a wait state while the page is brought into real storage. Without the handshaking feature, the entire VS1 virtual machine is placed in page wait by VM/370 until the needed page is available.

However, with the handshaking feature, a multiprogramming (or multitasking) VS1 virtual machine can dispatch one task while waiting for a page request to be answered for another task. VM/370 passes a pseudo page fault (program interrupt X'14') to VS1. When VS1 recognizes the pseudo page fault, it places only the task waiting for the page in page wait and can dispatch any other VS1 task. Thus, when VS1 uses pseudo page faults, its execution under the control of VM/370 more closely resembles its execution on a real machine.

When a page fault occurs for a VS1 virtual machine, VM/370 checks that the pseudo page fault portion of handshaking is active and that the VS1 virtual machine is in EC mode and enabled for I/O interrupts. Then, VM/370 reflects the page faults to VS1 by:

- Storing the virtual machine address, that caused the page fault, at location X'90', the translation exception address
- Reflecting a program interrupt (interrupt code X'14') to VS1
- Removing the VS1 virtual machine from page and execution wait

When VS1 recognizes program interrupt code X'14', it places the associated task in wait state. VS1 can then dispatch other tasks.

When the requested page is available in real storage, VM/370 reflects the same program interrupt to VS1, except that the high order bit in the translation exception address field is set on to indicate completion. VS1 removes the task from page wait; the task is then eligible to be dispatched.

VS1 NONPAGING MODE

When VS1 is run under the control of VM/370, it executes in nonpaging mode if:

- Its virtual address space is equal to the size of the VM/370 virtual machine
- Its virtual machine size is a least one megabyte
- The VM/VS Handshaking feature is available

When VS1 executes in nonpaging mode, it uses fewer privileged instructions and avoids duplicate paging. The VS1 Nucleus Initialization Program (NIP) fixes all VS1 pages to avoid the duplicate paging. Note, that the working set size may be larger for a VS1 virtual machine in nonpaging mode than for one not in nonpaging mode.

MISCELLANEOUS ENHANCEMENTS

When OS/VS1 is run in the VM/370 environment without the handshaking feature, some duplication results. VS1 must perform certain functions when it is run on a real machine; it continues to perform all those functions in a VM/370 virtual machine even though VM/370 also provides

services. However, with the handshaking feature, VS1 avoids many of the instructions and procedures that are redundant or less efficient in the VM/370 environment. For example, VS1 avoids:

- ISK (Insert Storage Key) instructions and instead uses a key table
- Seek separation for 2314 direct access devices
- The ENABLE/DISABLE sequence in the VS1 I/O Supervisor (IOS)
- TCH (Test Channel) instructions preceding SIO (Start I/O) instructions

HOW VIRTUAL MACHINES CAN COMMUNICATE WITH CP VIA THE DIAGNOSE INTERFACE

Since the DIAGNOSE instruction is restricted from its normal use in a virtual machine environment, a special Diagnose Interface has been established so that virtual machines can communicate easily and efficiently with CP.

By inserting DIAGNOSE instructions where appropriate in the operating system's code, the following functions can be requested by a virtual machine:

	Diagnose Code
• Examine the CPU's extended identification code.	0
• Examine the contents of real main storage.	4
• Invoke a virtual console function (that is, a CP command) from the virtual machine operating system.	8
• Obtain, from CP's pseudo timer: - Today's date (mm/dd/yy) - The time-of-day (hh:mm:ss) - Virtual and total CPU time used by the virtual machine	C
• Release pages of virtual storage.	10
• Manipulate your input spool file in one of the following ways: - Read the next reader, punch, or printer spool data - Select a new file for processing - Repeat the active file n times - Restart the current file at the beginning - Backspace one record - Retrieve the subsequent file descriptor	14
• Perform a standard DASD I/O operation.	18
• Clear the VM/370 I/O error recording area on disk.	1C
• Perform a general I/O operation for tape or disk.	20
• Interrogate CP's device type and features control blocks.	24
• Notify CP that a dynamically modified channel program is to be executed.	28
• Locate the start DASD address of the VM/370 I/O error recording area (LOGREC).	2C

	Diagnose Code
• Read one page of LOGREC data.	30
• Read the VM/370 system dump spool file.	34
• Read the VM/370 system symbol table.	38
• Dynamically update the VM/370 directory.	3C
• Generate accounting cards for the user.	4C
• Save the 3704/3705 Communications Controller program in page image format.	50
• Specify that the 3270's PA2 key is to be used for either: - Simulating an external interrupt to a virtual machine such as VS APL. - Clearing the output display of a 3270 screen.	54
• Display up to 1760 characters of data on a 3270 screen in a very rapid fashion.	58
• Prepare to display error messages according to the user's setting of the EMSG function.	5C
• Determine the virtual machine storage size.	60
• Find, load, or purge a named segment.	64

For more information on how to use these Diagnose Interface functions, refer to the section entitled "Diagnose Instruction in a Virtual Machine" in the VM/370: System Programmer's Guide.

MULTIPLE-ACCESS VIRTUAL MACHINES

Multiple-access programs are those which execute in a virtual machine and directly control terminals. These terminals need not be of the type supported by CP as virtual operator consoles, but may be of any type supported by the virtual machine. The lines used are either dedicated to the virtual machine via the directory entry or assigned to the virtual machine dynamically.

Figure 15 shows two multiple-access systems (controlled by virtual machines VM1 and VM2). Each controls real 3277s by using part of the resources of the real 3272, but it appears to both virtual machines as if they each have sole control of the real 3272. (The virtual system consoles of VM1 and VM2 are not shown.)

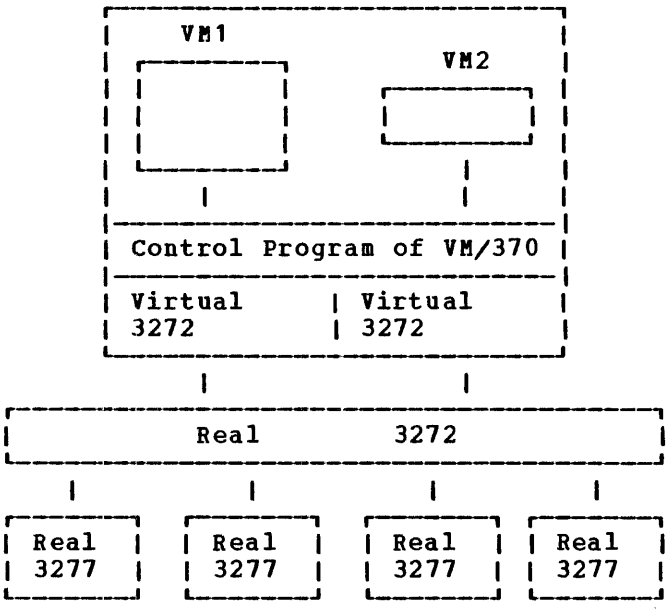


Figure 15. Virtual Devices: Local 3270 Terminals

Except when running a 3704 or 3705 in NCP mode, a subset of the lines of a real transmission control unit (TCU) can be defined as virtual lines for a virtual machine, as shown in Figure 16.

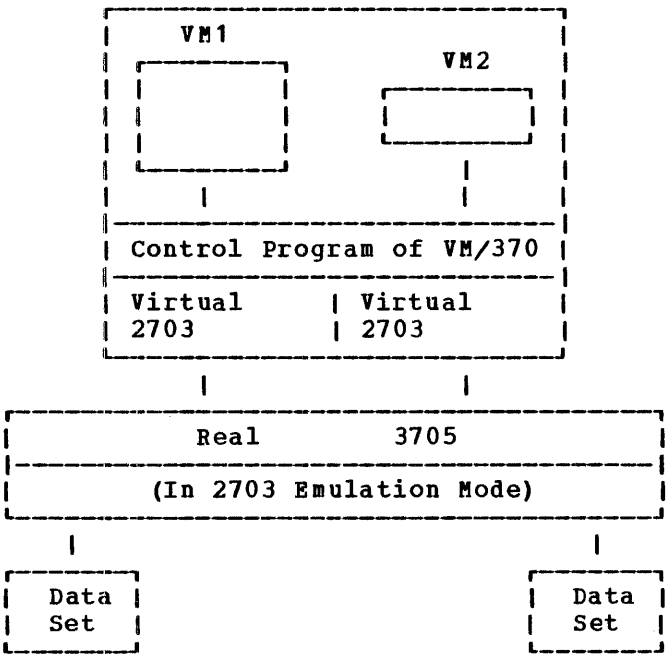


Figure 16. Virtual Devices: Remote Terminals

In Figure 16, two lines on the real 3705 are defined as virtual lines for two virtual machines named VM1 and VM2. The remaining lines may support virtual operator consoles.

The virtual machine operating system may be one like VM/370 itself, or TSO, that supports a number of remote terminals, as shown in Figure 17.

When the terminals supported by the virtual machine's operating system are of the same type as those supported by VM/370 as virtual system consoles, a real line can be assigned to a virtual line.

To do this, virtual lines are defined in the virtual machine's VM/370 directory entry via the SPECIAL control statement, or are added to the logged-on virtual machine via the CP DEFINE command.

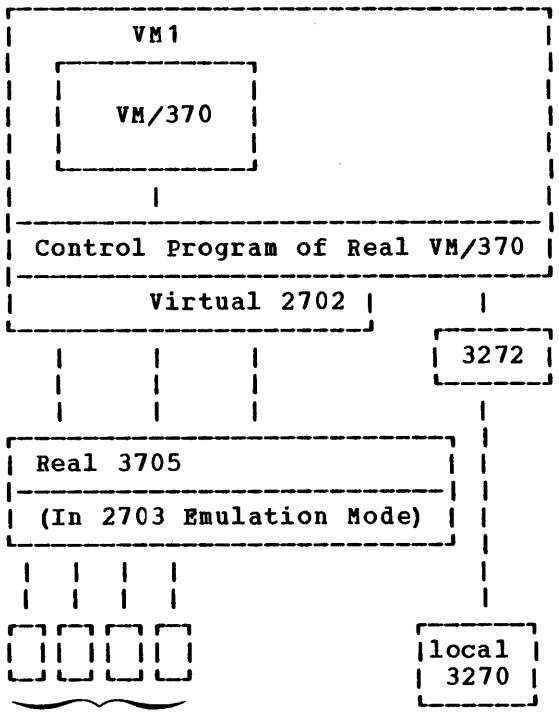


Figure 17. A Virtual VM/370 Multiple-Access System

Figure 18 illustrates a VM/370 directory entry for a multiple-access virtual machine to run VM/370 under VM/370.

```
USER VM370 PASSWORD 1M
  OPTION REALTIMER ECMODE
  CONSOLE 01F    3215
   SPOOL   00C   2540     READER
   SPOOL   00D   2540     PUNCH  B
   SPOOL   00E   1403     A
   DEDICATE 190 SYSRES
   DEDICATE 191 SYSWRK
   SPECIAL 080   2702     IBM
   SPECIAL 081   2702     IBM
   SPECIAL 082   2702     IBM
   SPECIAL 083   2702     IBM
   SPECIAL 070   3270
```

Figure 18. VM/370 Directory Entry for a VM/370 Virtual Machine

You can use the CP DIAL command to connect a terminal supported by both CP and a multiple-access system. Such terminals can be on either nonswitched or switched lines. To connect to the virtual machine in Figure 18, you would enter the command as follows:

```
dial vm370
```

The CP system matches the terminal type with an equivalent virtual line that is available and enabled (in this example, 070, 080, 081, 082, or 083). Once a connection is made, the terminal is controlled by the virtual machine to which it is logically connected (in this example, the VM/370 virtual machine). It remains connected until you log off using standard logoff procedures, or until the virtual machine is forcibly logged off. You are then free to log onto CP or to DIAL another multiple-access system.

Dial-up terminals supported by a multiple-access system may be of a different type than those supported by VM/370 as virtual system consoles. Such terminals must be on switched lines, and the CP DIAL command cannot be used. You must dial the multiple-access system's telephone numbers directly.

A communications system can be tested using multiple virtual machines in place of multiple real machines, as shown in Figure 19. The virtual 2701 units could each be defined as a one-line 2701; on the real machine there exists a single two-line 2701.

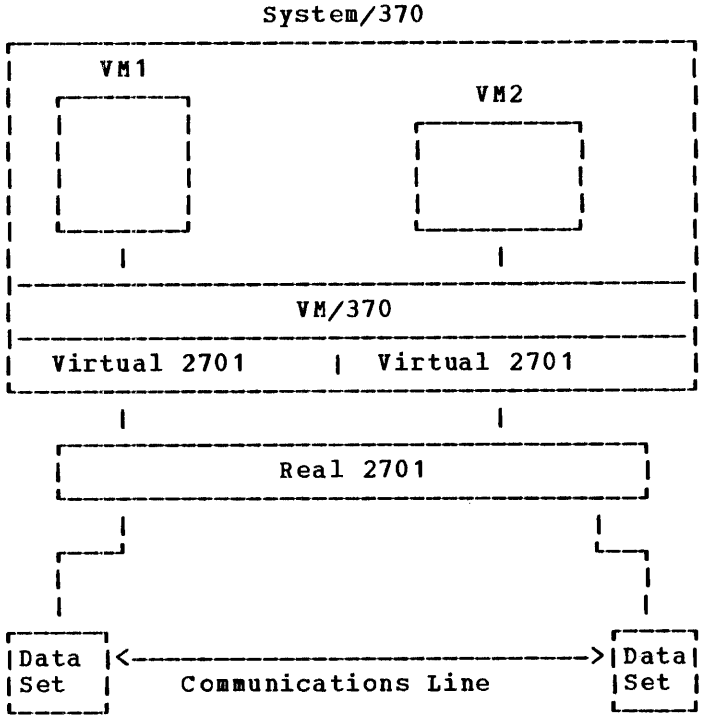


Figure 19. A Communications Test System

The communications test system in Figure 19 is based on the assumption that the real transmission control unit is equipped with the appropriate data sets and line capability.

Figure 20 illustrates a virtual transmission control unit running remote 3270-type units.

When the terminals supported by the multiple-access system are not of the type supported by VM/370 as virtual operator consoles, the real line appearances must be:

- Defined in the VM/370 directory entry for the virtual machine via the DEDICATE control statement, for example:

  DEDICATE vaddr raddr

  where vaddr is the virtual address and raddr is the real address of the appropriate line appearance on the real transmission control unit

  -- or --

- Attached to the virtual machine by an operator with privilege class B, for example:

  attach raddr to vm1 as vaddr

  where raddr is the real address of the appropriate line appearance on the real transmission control unit, and vaddr is the address of the line appearance as generated in the virtual machine operating system.

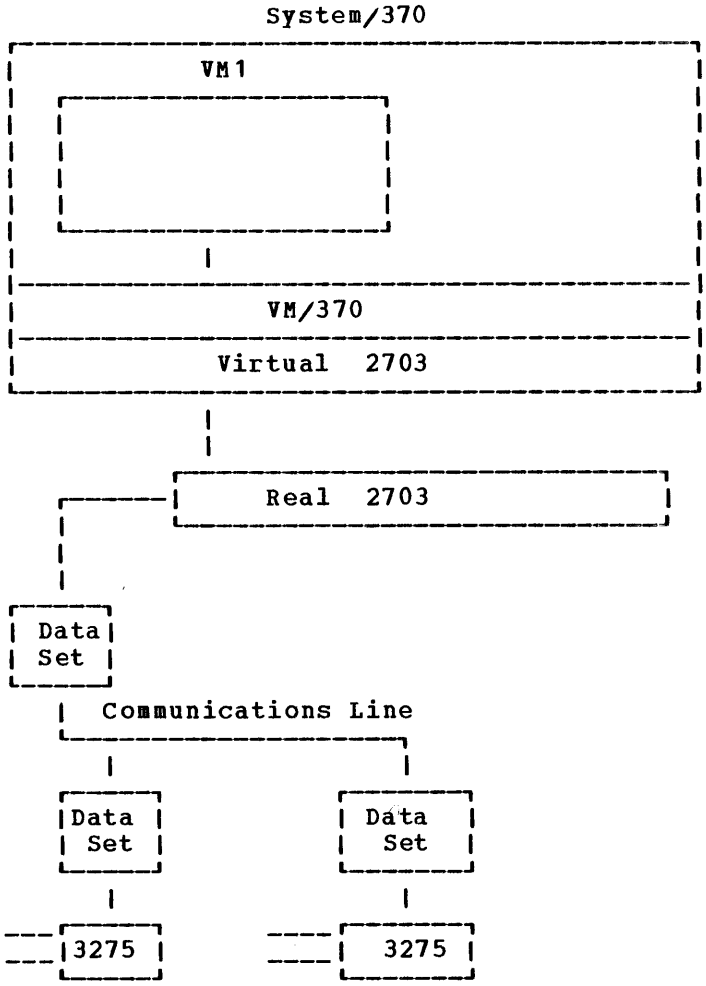


Figure 20. A Virtual 2703 TCU Controlling Remote 3270 Terminals

When virtual machine activity is initiated 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 be paged out before the time the virtual machine begins processing. The paging activity required so that the virtual machine can respond to the teleprocessing request may cause an increase in the time required to respond to the request.

Use the locked pages or reserved page frames options to improve performance.

If the program must be run in the dynamic paging area, then locking specific pages of the virtual machine into real storage may ease the problem. However, other than page zero and the page containing the TP interrupt handler, it is not always easy or possible to identify which specific pages are always required.

A more flexible approach than locked pages is the reserved page frames option. When a temporarily inactive virtual machine having this option is reactivated, these page frames are immediately available. If the program code or data required to satisfy the request was in real storage at the time the virtual machine became inactive, no paging activity is required for the virtual machine to respond.

If you use the OS Attached Support Processor (ASP) you may find virtual machines useful in two ways.

First, a virtual ASP system can be run by two virtual machines (VM1 and VM2) together with a virtual channel-to-channel adapter (CTCA). This is illustrated in Figure 21.

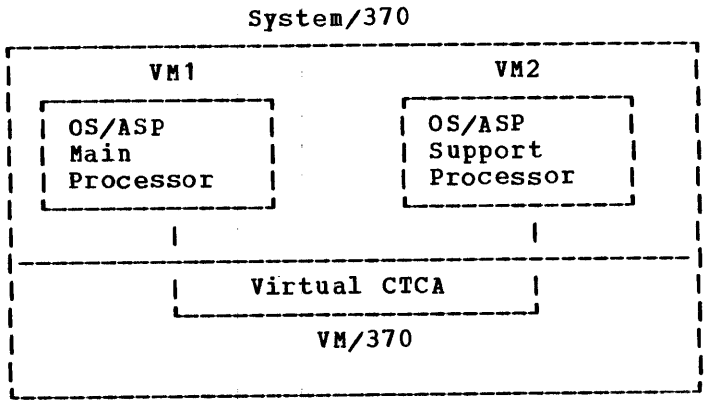


Figure 21. Two ASP Virtual Machines

The virtual ASP system (VM1 and VM2) shown in Figure 21 may be used to install and test a new ASP release, or for ASP system testing in a virtual environment concurrent with normal production. This eliminates the need to dedicate the real ASP computer for new system testing.

The VM/370 directory entry for each of the virtual ASP processors must contain a record defining a virtual channel-to-channel adapter in the form:

```
SPECIAL  280 CTCA
```

where 280 is the address of the channel-to-channel adapter as generated in the operating system.

When both virtual ASP machines are logged on VM/370, the CP COUPLE command must be issued by one of the virtual machine operators:

```
couple 280 to vm2  380
```

where 280 is the address of VM1's virtual channel-to-channel adapter, VM2 is the userid of the second virtual machine, and 380 is the address of VM2's virtual channel-to-channel adapter. After the channels are coupled, the operator of each virtual machine can then load his operating system and start running ASP.

Second, an additional virtual machine (VM3), with a real channel-to-channel adapter to a real System/360 or System/370, can be running as either the main or support processor in a production ASP system. This is illustrated in Figure 22.

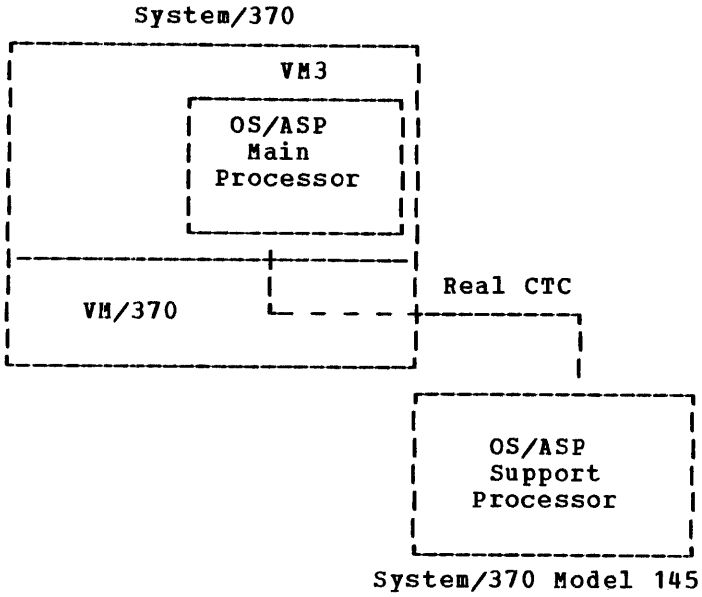


Figure 22. One Real and One Virtual ASP Machine

The real channel-to-channel adapter must be defined in the VM/370 system generation procedure via the RDEVICE macro with a device type of CTCA (DEVTYPE=CTCA). The virtual machine (VM3) must have this device assigned to it before the IPL. This can be done via the DEDICATE statement in the virtual machine's VM/370 directory entry as follows:

```
DEDICATE 280 380
```

where 280 is the address of the channel-to-channel adapter as generated in the OS system, and 380 is the address of the real channel-to-channel adapter as specified in the VM/370 system generation procedure.

If there is no DEDICATE statement for the channel-to-channel adapter in the virtual machine's directory entry, a resource operator with privilege class B must attach the real channel-to-channel adapter to the virtual machine.

Note: See the description of the COUPLE, DEFINE, and DETACH commands in the VM/370: CP Command Reference for General Users for further information on the virtual channel-to-channel adapter.

VM/370 does not itself include any channel switching facilities. However, channel switching can be used between VM/370 and another operating system if the other operating system supports channel switching. The two- or four-channel switch can be used between:

- Two CPUs, one running VM/370 and the other running an operating system that supports channel switching.
- VM/370 and an operating system running in a virtual machine under the control of VM/370 on one CPU, if the virtual machine operating system supports channel switching.

CHANNEL SWITCHING BETWEEN TWO CPUS

You can use the two- or four-channel switch for devices attached to two CPUs. For example, one CPU could be running VM/370 and the other could be running OS. The configuration is shown in Figure 23.

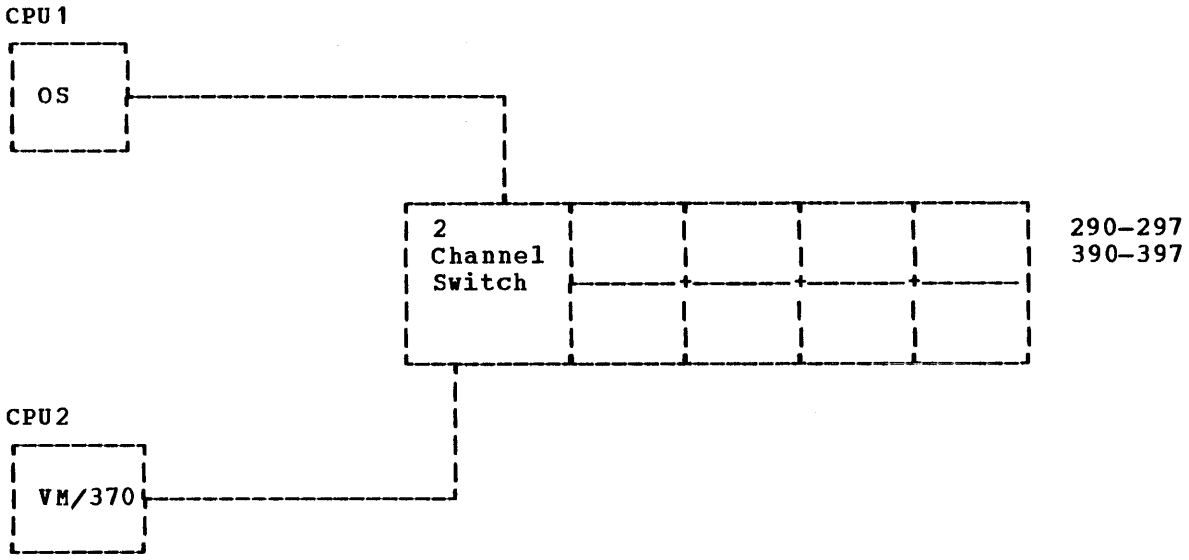


Figure 23. Channel Switching between Two CPUs

VM/370 requires the following RDEVICE and RCTLUNIT macros to support this configuration:

```
RDEVICE  ADDRESS=(290,8),DEVTYPE=3330,FEATURE=2CHANSW
RDEVICE  ADDRESS=(390,8),DEVTYPE=3330,FEATURE=2CHANSW
RCTLUNIT ADDRESS=290,CUTYPE=3830
RCTLUNIT ADDRESS=390,CUTYPE=3830
```

These macros make it possible for you to run VM/370 on CPU1 or CPU2. If you are always going to run VM/370 on CPU2, you can eliminate one path (eliminate one RDEVICE and RCTLUINT set of macros).

If any I/O devices controlled by VM/370 for its own exclusive use are attached to a control unit with a two- or four-channel switch, the CPU controlling the other channel interface must vary the CP-owned devices offline. For example, if all eight disks in the preceding configuration are mounted and two of those disk are CP-owned volumes (such as CP system residence and CP paging and spooling volumes), the OS system running on CPU1 must vary the CP-owned volumes offline. This procedure protects volumes that CP needs.

CHANNEL SWITCHING ON ONE CPU

You can also use the two- or four-channel switch for devices attached to one CPU that is running VM/370. For example, one CPU could be running VM/370 with OS running in a VM/370 virtual machine. The configuration is shown in Figure 24.

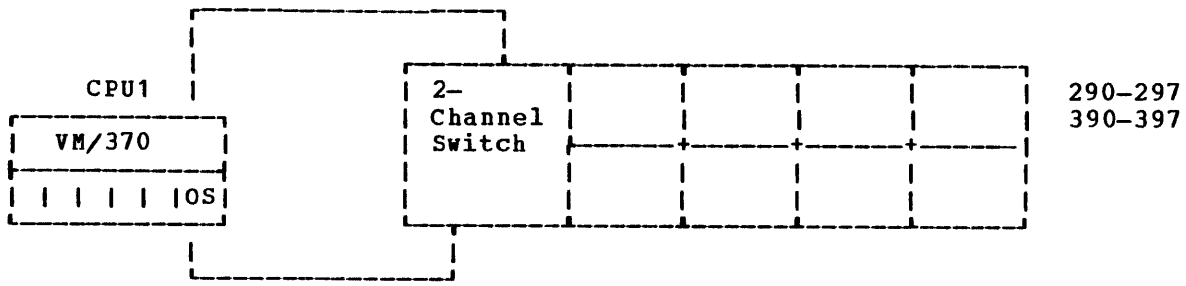


Figure 24. Channel Switching on One CPU

VM/370 requires the following RDEVICE and RCTLUNIT macros to support this configuration:

```
RDEVICE  ADDRESS=(290,8),DEVTYPE=2314,FEATURE=2CHANSW
RDEVICE  ADDRESS=(390,8),DEVTYPE=2314,FEATURE=2CHANSW
RCTLUNIT ADDRESS=290,CUTYPE=IFA
RCTLUNIT ADDRESS=390,CUTYPE=IFA
```

For this example, you should have all the devices associated with one path offline when you load VM/370. Otherwise, the following message is displayed:

```
DMKCPI954E DASD raddr VOLID volid NOT MOUNTED,
           DUPLICATE OF DASD raddr
```

In this example, the 2314 DASD devices can only be used by the OS system running in a virtual machine if they are dedicated to that virtual machine via the ATTACH command or the DEDICATE control statement in the VM/370 directory. The device addresses generated for the virtual machine operating system do not need to be the same as those defined for the real machine. Also, any operating system that is running under the control of VM/370, and using channel switching with VM/370, cannot access minidisks which are on devices with channel switching. Because reserve/release is supported only for minidisks that are full packs,

minidisks that are not defined as full packs cannot be accessed via channel switching on one CPU.

However, minidisks can be used if one of the channel interfaces is using the device as a dedicated device and the other channel interface is using reserve/release support. In this example, such support implies a full pack minidisk. Suppose that the channel 2 devices are online and channel 3 device offline when VM/370 is loaded. Also suppose that 291 contains a full pack minidisk. When VM/370 is loaded, device 291 is considered a CP system device. VM/370 virtual machines can access the full pack minidisk at 291, and it has reserve/release protection. Then, the OS virtual machine controlling the channel 3 interface can have the same pack dedicated to it as 391. The OS virtual machine can only access the full pack minidisk as 391; other VM/370 virtual machines can only access it as 291.

As another example, consider channel switching for tapes. If the real configuration includes a 2816 Switching Unit or a Two- or Four-Channel Switch feature, it can be made to operate under control of a virtual machine operating system. For example, if 580 and 680 are the alternate device addresses for a particular tape drive, then:

- Generate the virtual machine operating system for the appropriate hardware (in this case a 2816 Switching Unit on channels 5 and 6).
- Generate CP as though 580 and 680 are different devices (with different control units and channels) and generate them with the Two-Channel Switch feature (FEATURE=2CHANSW).
- Issue the CP ATTACH command for both device addresses (580 and 680) whenever the real device is to be attached to the virtual machine.

The device addresses generated for the virtual machine operating system do not need to be the same as those on the real machine.

The devices must be used by the virtual machine as dedicated devices (attached, or defined with a DEDICATE statement in the VM/370 directory).

VM/370 does not include any reserve/release support for virtual machines sharing DASD volumes. If the real configuration includes a Two- or Four-Channel Switch feature, reserve/release support can be made to operate under control of the virtual machine operating system. In this way, a virtual machine can share a DASD volume (either virtual or real).

For example, if 230 and 330 are alternate device addresses for a particular DASD facility to be shared by USERA and USERB (two virtual machines running on the same real computing system), then:

- Generate the virtual machine operating system for USERA to support the appropriate hardware (device at 230, Two-Channel Switch) and software (RESERVE/RELEASE).
- Generate the virtual machine operating system for USERB to support the appropriate hardware (device at 330, Two-Channel Switch) and software (RESERVE/RELEASE).
- Generate CP as though 230 and 330 were different devices (with different control units and channels); and generate them with the Two-Channel Switch feature (FEATURE=2CHANSW).
- Issue the CP ATTACH command, attaching device 230 to USERA and device 330 to USERB.

If the system generated for USERB is to run in a real machine rather than a virtual machine:

- Generate the CP system with device 230 but not 330.
- Issue the CP ATTACH command, attaching device 230 to USERA.

In both cases:

- The device addressess generated for systems to run in a virtual machine do not need to be the same as on the real machine.
- The devices used by virtual machines must be dedicated (attached or defined with a DEDICATE record in the VM/370 directory). Minidisks cannot be shared in this manner; only full packs can be shared in this manner.

Note: Neither the CP sysres nor any other CP-owned disk should be shared between two CPUs, even though it is theoretically possible.

Appendixes

The following reference information is included:

VM/370 ASSEMBLER

The VM/370 Assembler is distributed as a part of the VM/370 system and is required for installation and further support of the system. All necessary installation and support macros are provided in CMS libraries.

The Conversational Monitor System (CMS) and the Remote Spooling Communications Subsystem (RSCS) are components of VM/370 and are distributed with it. Certain other facilities mentioned in this publication are not part of VM/370, but can be separately ordered from IBM. These include: IBM System/360 and System/370 operating systems, IBM language processors and other program products, IBM Installed User Programs, and IBM Field Developed Programs. For more information, contact your IBM representative.

PROGRAM PRODUCTS

Figure 25 lists the IBM Program Products that are executable under CMS.

To find the amount of storage required to install a program product, refer to the appropriate program product publication.

INSTALLED USER PROGRAMS

A text-processing program is available: IBM Installed User Program (IUP) SCRIPT/370 (IBM Program No. 5796-PAF). SCRIPT/370 creates formatted output from one or more CMS files, each of which contains text and/or SCRIPT control words. The SCRIPT files are created and modified at a terminal using the CMS Editor.

SCRIPT/370 manuscript facilities include right margin justification, line centering, inserting top and bottom titles, and the ability to invoke additional SCRIPT input files from the file being processed. Other facilities to assist in the preparation of large documents include symbolic capabilities that can generate a table of contents, and number pages and figures.

IBM Program Product	IBM Program Number
DOS PL/I Optimizing Compiler	5736-PL1
DOS PL/I Resident Library	5736-LM4
DOS PL/I Transient Library	5736-LM5
DOS/VS COBOL Compiler & Library	5746-CB1
DOS/VS COBOL Object Library	5746-LM4
OS Code & Go FORTRAN	5734-FO1
OS FORTRAN IV (G1)	5734-FO2
OS FORTRAN IV Library (Mod I)	5734-LM1
OS FORTRAN IV (H) Extended	5734-FO3
OS FORTRAN Library (Mod II)	5734-LM3
FORTRAN Interactive Debug	5734-FO5
OS/VS COBOL Compiler & Library	5740-CB1
OS/VS COBOL Library Only	5740-LM1
OS Full American National Standard COBOL Version 4 Compiler and Library	5734-CB2
OS Full American National Standard COBOL Version 4 Library	5734-LM2
OS COBOL Interactive Debug	5734-CB4
OS PL/I Optimizing Compiler	5734-PL1
OS PL/I Resident Library	5734-LM4
OS PL/I Transient Library	5734-LM5
OS PL/I Optimizing Compiler and Libraries	5734-PL3
OS PL/I Checkout Compiler	5734-PL2
VS BASIC Processor	5748-XX1
VS APL	5748-AP1

Figure 25. IBM Program Products

A time-sharing facility is available as an IUP: the McGill University System for Interactive Computing (MUSIC), IBM Program No. 5796-AJC. MUSIC is a conversational time-sharing operating system that can execute in a virtual machine under VM/370.

MUSIC supports several programming languages, for example: OS FORTRAN G1, VS BASIC, OS ANS COBOL, APL and OS Assembler F. MUSIC also has a batch facility, Context Editor, accounting, sort programs, and command language. For additional information about MUSIC, see the MUSIC: General Information Manual, Order No. G320-1238.

A data reduction program is available as an IUP: the statistics-generating package for VM/370 (VM/SGP), IBM Program No. 5796-PDD. VM/SGP reduces the data that is collected by the VM/370 measurement facility, thus providing useful information for installation management, systems programmers, consultants, and users.

INTEGRATED EMULATORS

Emulator-dependent programs (except for DOS emulation under OS or OS/VS) that execute on a particular System/370 equipped with the appropriate compatibility features can execute on that System/370 in DOS or OS virtual machines under VM/370.

Figure 26 shows, by System/370 model number, which integrated emulators can execute under VM/370 and the compatibility feature numbers (#xxxx) that are required.

No changes are required to the emulators, to DOS or OS, or to VM/370 to allow emulator-dependent programs to execute in virtual machines.

On the System/370 Model 158 only, the Virtual Machine Assist feature cannot operate concurrently with the 7070/7074 compatibility feature (Feature #7117).

System/370 Model	S/360 Model 20	1401 1440 1460 1410 7010	1401 1440 1460 1410 7010	7070 7074	7080	709 7090 7094 7094II
135	#7520	#4457				
145		#4457	#4458			
155 II, 158			#3950	#7117		
165 II				#7117	#7118	#7119
168				#7127	#7128	#7129

Figure 26. Integrated Emulators that Execute under VM/370

Appendix B: CP Restrictions

A virtual machine created by VM/370 is capable of running an IBM System/360 or System/370 operating system as long as certain CP restrictions are not violated. Virtual machine restrictions and certain execution characteristics are stated in this appendix.

DYNAMICALLY MODIFIED CHANNEL PROGRAMS

In general, virtual machines may not execute channel programs that are dynamically modified (that is, channel programs that are changed between the time the START I/O (SIO) is issued and the time the input/output ends, either by the channel program itself or by the CPU). However, some dynamically modified channel programs are given special consideration by CP: specifically, those generated by the Indexed Sequential Access Method (ISAM) running under OS/PCP, OS/MFT, and OS/MVT; those generated by ISAM running in an OS/VS virtual=real partition; and those generated by the OS/VS Telecommunications Access Method (TCAM) Level 5, with the VM/370 option.

The self-modifying channel programs that ISAM generates for some of its operations receive special handling if the virtual machine using ISAM has that option specified in its VM/370 directory entry. There is no such restriction for DOS ISAM, or for ISAM if it is running in an OS/VS virtual=virtual partition. If ISAM is to run in an OS/VS virtual=real partition, you must specify the ISAM option in the VM/370 directory entry for the OS/VS virtual machine.

Virtual machines using OS/VS TCAM (Level 5, generated or invoked with the VM/370 option) issue a DIAGNOSE instruction when the channel program is modified. This instruction causes CP to reflect the change in the virtual CCW string to the real CCW string being executed by the channel. CP is then able to execute the dynamically modified channel program properly.

The restriction against dynamically modified channel programs does not apply if the virtual machine has the virtual=real performance option and the NOTRANS option has been set on.

MINIDISK RESTRICTIONS

The following restrictions exist for minidisks:

1. In the case of read home address with the skip bit off, VM/370 modifies the home address data in user storage at the completion of the channel program because the addresses must be converted for minidisks; therefore, the data buffer area may not be dynamically modified during the input/output operation.

2. On a minidisk, if a CCW string uses multitrack search on input/output operations, subsequent operations to that disk must have preceding seeks or continue to use multitrack operations. There is no restriction for dedicated disks.

3. OS/PCP, MFT, and MVT ISAM or OS/VS ISAM running virtual=real may be used with a minidisk only if the minidisk is located at the beginning of the physical disk (that is, at cylinder zero). There is no such restriction for DOS ISAM or OS/VS ISAM running virtual=virtual.

4. VM/370 does not return an end-of-cylinder condition to a virtual machine that has a virtual 2311 mapped to the top half (that is, tracks 0 through 9) of 2314 or 2319 cylinders.

5. If the user's channel program for a minidisk does not perform a seek operation, then to prevent accidental accessing, VM/370 inserts a positioning seek operation into the user's channel program. Thus, certain channel programs may generate a condition code (CC) of zero on a SIO instead of an expected CC of one, which is reflected to the virtual machine. The final status is reflected to the virtual machine as an interrupt.

6. A DASD channel program directed to a 3330, 3340, or 3350 device may give results on dedicated drives which differ from results on minidisks having non-zero relocation factors if the channel program includes multiple-track operations and depends on a search ID high or a search ID equal or high to terminate the program. This is because the record 0 count fields on the 3330, 3340, and 3350 must contain the real cylinder number of the track on which they reside. Therefore, a search ID high, for example, based on a low virtual cylinder number may terminate prematurely if a real record 0 is encountered.

   Note: Minidisks with non-zero relocation factors on 3330, 3340, and 3350 devices are not usable under OS and OS/VS systems. This is because the locate catalog management function employs a search ID equal or high CCW to find the end of the VTOC.

7. The IBCDASDI program cannot assign alternate tracks for a 3330, 3340, or 3350 minidisk.

8. If the DASD channel programs directed to 3330/3340/3350 devices include a write record R(0), results differ depending on whether the 3330/3340/3350 is dedicated (this includes a minidisk defined as the entire device) or nondedicated. For a dedicated 3330/3340/3350, a write R(0) is allowed, but the user must be aware that the track descriptor record may not be valid from one 3330/3340/3350 to another. For a nondedicated 3330/3340/3350, a write record R(0) is replaced by a read record R(0) and the skip flag is set on. This could result in a command reject condition due to an invalid command sequence.

9. When performing DASD I/O, if the record field of a search ID argument is zero when a virtual Start I/O is issued, but the search ID argument is dynamically read by the channel program before the search ID CCW is executed, then the real search ID uses the relocated search argument instead of the argument that was read dynamically. To avoid this problem, the record field of a search ID argument should not be set to binary zero if the search argument is to be dynamically read or if a search ID on record 0 is not intended.

TIMING DEPENDENCIES

Timing dependencies in input/output devices or programming do not function consistently under VM/370:

1. The following telecommunication access methods (or the designated option) violate the restriction on timing dependency by using program-controlled interrupt techniques and/or the restriction on dynamically modified channel programs:

   - OS Basic Telecommunications Access Method (BTAM) with the dynamic buffering option.
   - OS Queued Telecommunications Access Method (QTAM).
   - DOS Queued Telecommunications Access Method (QTAM).
   - OS Telecommunications Access Method (TCAM).
   - OS/VS Telecommunications Access Method (TCAM) Level 4 or earlier, and Level 5 if TCAM is not generated or invoked with the VM/370 option.

   These access methods may run in a virtual=real machine with CCW translation suppressed by the SET NOTRANS ON command. Even if SET NOTRANS ON is issued, CCW translation will take place if one of the following conditions is in effect:

   - The channel program is directed at an a nondedicated device (such as a spooled unit record device, a virtual CTCA, a minidisk, or a console).
   - The channel program starts with a SENSE operation code.
   - The channel program is for a dialed terminal.
   - START I/O tracing is in effect.
   - The CAW is in page zero or beyond the end of the virtual=real area.

   (OS BTAM can be generated without dynamic buffering, in which case no virtual machine execution violations occur. However, the BTAM reset poll macro will not execute under VM/370 if issued from third level storage. For example, a reset poll macro has a NOP effect if executed from a virtual=virtual storage under VS1 which is running under VM/370.)

2. Programming that makes use of the PCI channel interrupt for channel program modification or processor signalling must be written so that processing can continue normally if the PCI is not recognized until I/O completion or if the modifications performed are not executed by the channel.

3. Devices that expect a response to an interrupt within a fixed period of time may not function correctly because of execution delays caused by normal VM/370 system processing. An example of such a device is the IBM 1419 Magnetic Character Reader.

4. The operation of a virtual block multiplexer channel is timing dependent. For this reason, the channel appears available to the virtual machine operating system, and channel available interrupts are not observed. However, operations on virtual block-multiplexing devices should use the available features like Rotational Position Sensing to enhance utilization of the real channels.

CPU MODEL-DEPENDENT FUNCTIONS

On the System/370 Model 158 only, the Virtual Machine Assist feature cannot operate concurrently with the 7070/7074 compatibility feature (Feature #7117).

Programs written for CPU model-dependent functions may not execute properly in the virtual machine under VM/370. The following points should be noted:

1. Programs written to examine the machine logout area do not have meaningful data since VM/370 does not reflect the machine logout data to a virtual machine.

2. Programs written to obtain CPU identification (via the Store CPU ID instruction, STIDP) receive the real machine value. When the STIDP instruction is issued by a virtual machine, the version code contains the value 255 in hexadecimal ("FF") to represent a virtual machine.

3. Programs written to obtain channel identification (via the Store Channel ID instruction, STIDC) receive information from the virtual channel block. Only the virtual channel type is reflected; the other fields contain zeroes.

4. No simulation of other CPU models is attempted by VM/370.

VIRTUAL MACHINE CHARACTERISTICS

Other characteristics that exist for a virtual machine under VM/370 are as follows:

1. If the virtual=real option is selected for a virtual machine, input/output operations specifying data transfer into or out of the virtual machine's page zero, or into or out of storage locations whose addresses are greater than the storage allocated by the virtual=real option, must not occur. The storage-protect-key mechanism of the IBM System/370 CPU and channels operates in these situations but is unable to provide predictable protection to other virtual machines. In addition, violation of this restriction may compromise the integrity of the system. The results are unpredictable.

2. VM/370 has no multiple path support and, hence, does not take advantage of the two-channel switch. However, a two-channel switch can be used between the IBM System/370 running a virtual machine under VM/370 and another CPU.

3. The DIAGNOSE instruction cannot be issued by the virtual machine for its normal function. VM/370 uses this instruction to allow the virtual machine to communicate system services requests. The Diagnose interface requires the operand storage addresses passed to it to be real to the virtual machine issuing the DIAGNOSE instruction. For more information about the DIAGNOSE instruction in a virtual machine, see the VM/370: System Programmer's Guide.

4. A control unit normally never appears busy to a virtual machine. An exception exists when a forward space file or backward space file command is executed for a tape drive. Subsequent I/O operations to the same virtual control unit result in a control

unit busy condition until the forward space file or backward space file command completes. If the real tape control unit is shared by more than one virtual machine, a control unit busy condition is reflected only to the virtual machine executing the forward space file or backward space file command. When a virtual machine attempts an I/O operation to a device for which its real control unit is busy, the virtual machine is placed in I/O wait (nondispatchable) until the real control unit is available. If the virtual machine executed a SIOF instruction (rather than SIO) and was enabled for block-multiplexing, it is not placed in I/O wait for the above condition.

5. The CP IPL command cannot simulate self-modifying IPL sequences off dedicated unit record devices or certain self-modifying IPL sequences off tape devices.

6. The VM/370 spooling facilities do not support punch-feed-read, stacker selection, or column binary operations. Detection of carriage control channels is supported for a virtual 3211 only.

7. VM/370 does not support count control on the virtual 1052 operator's console.

8. Programs that use the integrated emulators function only if the real computing system has the appropriate compatibility feature. VM/370 does not attempt simulation. The DOS emulator running under OS or OS/VS is not supported under VM/370.

9. The READ DIRECT and WRITE DIRECT instructions are not supported for a virtual machine.

10. The System/370 SET CLOCK instruction cannot be simulated and, hence, is ignored if issued by a virtual machine. The System/370 STORE CLOCK instruction is a nonprivileged instruction and cannot be trapped by VM/370; it provides the true TOD clock value from the real CPU.

11. The 1050/1052 Model 2 Data Communication System is supported only as a keyboard operator's console. Card reading, paper tape I/O, and other modes of operation are not recognized as unique, and hence may not work properly. This restriction applies only when the 1050 system is used as a virtual machine operator's console. It does not apply when the 1050 system is attached to a virtual machine via a virtual 2701, 2702, or 2703 line.

12. The pseudo-timer (usually device address OFF, device type TIMER) does not return an interrupt from a Start I/O; therefore, do not use EXCP to read this device.

13. A virtual machine device IPL with the NOCLEAR option overlays one page of virtual machine storage. The IPL simulator uses one page of the virtual machine to initiate the IPL function. The starting address of the overlayed page is either the result of the following formula:

$$\frac{\text{virtual machine size}}{2} = \text{starting address of the overlayed page}$$

or the hexadecimal value 20,000, whichever is smaller.

14. To maintain system integrity, data transfer sequences to and from a virtual system console are limited to a maximum of 2032 bytes. Channel programs containing data transfer sequences that violate

this restriction are terminated with an interrupt whose CSW status indicates incorrect length and a channel program check.

Note: A data transfer sequence is defined as one or more read or write CCWs connected via chain data. The introduction of command chaining defines the start of a new data transfer sequence.

15. When an I/O error occurs on a device, the System/370 hardware maintains a contingent connection for that device until a SENSE channel command is executed and sense data is recorded. That is, no other I/O activity can occur on the device during this time. Under VM/370, the contingent connection is maintained until the SENSE command is executed, but I/O activity from other virtual machines can begin on the device while the sense data is being reflected to the virtual machine. Therefore, the user should be aware that on a shared disk, the access mechanism may have moved during this time.

16. The mode setting for 7-track tape devices is maintained by the control unit. Therefore, when a virtual machine issues the SET MODE channel command to a 7-track tape device, it changes the mode setting of all 7-track tape devices attached to that control unit.

    This has no effect on virtual machines (such as OS or DOS) that issue SET MODE each time a CCW string is to be executed. However, it can cause a problem if a virtual machine fails to issue a SET MODE with each CCW string executed. Another virtual machine may change the mode setting for another device on the same control unit, thereby changing the mode setting of all 7-track tape devices attached to that control unit.

17. OS/VS2 is supported in uniprocessor mode only.

18. A shared system or one that uses discontiquous saved segments cannot be loaded (via IPL) into a virtual machine running in the virtual=real area.

19. The DUMMY feature for VSAM data sets is not supported and should not be used at program execution time. Specifying this option on the DLBL command will cause an execution-time OPEN error. See VM/370: System Messages for additional information.

Index

READER'S COMMENT FORM

Title: **IBM Virtual Machine Facility/370: Operating Systems in a Virtual Machine**

Order No. **GC20-1821-0**

Please check or fill in the items; adding explanations/comments in the space provided.

Which of the following terms best describes your job?

☐ Customer Engineer	☐ Manager	☐ Programmer	☐ Systems Analyst
☐ Engineer	☐ Mathematician	☐ Sales Representative	☐ Systems Engineer
☐ Instructor	☐ Operator	☐ Student/Trainee	☐ Other (explain below)

How did you use this publication?

☐ Introductory text ☐ Reference manual ☐ Student/ ☐ Instructor text

☐ Other (explain) _____

Did you find the material easy to read and understand? ☐ Yes ☐ No (explain below)

Did you find the material organized for convenient use? ☐ Yes ☐ No (explain below)

Specific criticisms (explain below)

Clarifications on pages _____

Additions on pages _____

Deletions on pages _____

Errors on pages _____

Explanations and other comments:

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

Trim Along This Line

GC20-1821-0

YOUR COMMENTS PLEASE . . .

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance and/or additional publications or to suggest programming changes will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.* Your comments will be carefully reviewed by the person or persons responsible for writing and publishing this material. All comments or suggestions become the property of IBM.

FOLD FOLD

FIRST CLASS
PERMIT NO. 172
BURLINGTON, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
VM/370 PUBLICATIONS
24 NEW ENGLAND EXECUTIVE PARK
BURLINGTON, MASS. 01803

FOLD FOLD

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

Trim Along This Line

IBM Virtual Machine Facility/370: Op. Sys. in a Virtual Machine

Printed in U. S. A.

GC20-1821-0

SUGGESTION
INPUT
FORM

Virtual Machine Facility/370:
Operating System in a Virtual Machine **Order No. GC20-1821-0**

You can use this form to submit conversion hints for possible publication in this book. It is understood that IBMand its affiliated companies shall have the nonexclusive right, in their discretion, to use, copy, and distribute all submitted information or material, in any form, for any and all purposes, without any obligation to the submitter, and that the submitter has the unqualified right to submit such information or material upon such basis. When submitting conversion hints, indicate the system and release level from which you are converting and the system and release level to which you are converting.

Your views about this publication might help improve its usefulness; this form will be sent to the author's department for appropriate action. *Using this form to request system assistance or additional publications will delay response.* For more direct handling of such requests, please contact your IBMrepresentative or the IBM Branch Office serving your locality.

What is your occupation? _____

Order No. of latest Technical Newsletter (if any) for this publication: _____

Name of your IBM representative _____

Please indicate your name, customer name, address, and telephone number in the space below.

State problem (attach examples, etc., when available).

Suggested solution (may be described here, in letter form appended to this form, and/or in a computer printout if you prefer).

Trim Along This Line

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

YOUR COMMENTS PLEASE . . .

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance and/or additional publications or to suggest programming changes will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.* Your comments will be carefully reviewed by the person or persons responsible for writing and publishing this material. All comments or suggestions become the property of IBM.

FOLD FOLD

FIRST CLASS
PERMIT NO. 172
BURLINGTON, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
VM/370 PUBLICATIONS
24 NEW ENGLAND EXECUTIVE PARK
BURLINGTON, MASS. 01803

FOLD FOLD

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

Trim Along This Line

IBM Virtual Machine Facility/370: Op. Sys. in a Virtual Machine
Printed in U. S. A.
GC20-1821-0