IBM

SC19-6201-06

Virtual Machine/System Product

Planning Guide and Reference

Release 6

IBM

Virtual Machine/System Product
Planning Guide and Reference
Release 6

SC19-6201-06

Seventh Edition (August 1988)

This edition, SC19-6201-06 is a major revision of SC19-6201-05, and applies to Release 6 Virtual Machine/System Product (VM/SP), program number 5664-167, unless otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Summary of Changes

For a detailed list of changes, see "Summary of Changes" on page 437.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Ordering Publications

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

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

© Copyright International Business Machines Corporation 1980, 1981, 1982, 1983, 1985, 1986, 1987, 1988. All rights reserved.

Preface

This book is a reference manual for all Virtual Machine/System Product (VM/SP) Release 6 users. It describes the enhanced functions and capabilities that improve VM/SP's performance and make it a more versatile product for a wide range of applications.

This publication includes information about:

- Planning for system generation
- Defining your VM/SP system
- Generating a 3704/3705/3725 control program that runs with VM/SP
- Updating VM/SP.

Who This Book Is For

The intended audience is system programmers and anyone responsible for the planning, installation, and updating of a VM/SP system. The reader is expected to have a general understanding of data processing and teleprocessing techniques. This book assumes you have thought about:

- What VM/SP functions your site requires
- What connections you need to other sites and the implications for coordination
- What your hardware and physical requirements are and the implications for coordination.

How This Book Is Organized

This planning guide is divided into two parts and four appendices.

"Part I. Planning for System Generation" describes the various components, options, and features of VM/SP and tells you what you must do to install them.

"Part II. Defining Your VM/SP System" tells you how to create the files that define your system. These are the Real I/O Configuration (DMKRIO), CP System Control (DMKSYS), VM/SP Directory (VMUSERS DIRECT), System Name Table (DMKSNT), Forms Control Buffer Load (DMKFCB), and CMS Nucleus Generation Profile (DMSNGP) files.

The appendices include information about:

- Configuration aids for VM/SP
- VM/SP restrictions
- Coding the NAMESYS macro
- Sample SNTMAP output.

This book also has a glossary and index.

Contents

Part I. Planning Your VM System

Part I contains important planning information for your VM/SP system. It describes the various components, options, and features of VM/SP and tells you what you must do to install them.

The following topics are discussed in Part I:

- Introduction to Planning and VM/SP
- Planning Device Configurations for VM/SP
- Estimating VM/SP Storage Requirements
- Planning for Other VM/SP Areas
- Extending VM/SP
- Planning for Hardware Devices.

Chapter 1. Introduction to Planning and VM/SP

Contents of Chapter 1

What Planning Is

Planning is an important part of the VM/SP installation process. It helps prevent problems and lets you anticipate system requirements. The better a VM/SP system installation process is planned, the less time it takes to do the job. If properly done, the actual installation takes far less time than the planning process.

What Planning Involves

Planning tasks can be organized into the following categories:

Installation. VM/SP works for and with other products that have certain requirements about the way you define VM/SP. Planning for those requirements is essential to installing VM/SP.

Customization. VM/SP must be customized to work in your environment to handle your particular needs. You must understand what VM/SP needs to perform the basic functions you desire, what options it offers, and what implications those options may have in your environment.

Operation. By understanding VM/SP operation, you can determine how it should be managed for operating within your installation. You should decide who will be handling the operations and whether any special training will be required.

Administration. Knowing what VM/SP functions will be available to your users will help you determine what administrative tasks you will be performing. For instance, will users be responsible for operational tasks on remote devices? Will users need to identify themselves on remote systems to which they submit jobs?

Diagnosis. Problems encountered are not always with the VM/SP product. They may be with communications lines or network connections with other systems. Your installation's plan for handling problem situations and follow-up diagnosis can help speed recovery and save time.

Another area of planning that should not be overlooked is *migration*. If you are migrating from another release of VM/SP, there will be certain requirements. You will have to plan when it will be done, how long will it take, and who should do it.

Planning Considerations for Installation

Among the things that must be considered when planning for VM/SP installation are:

- System control files (System Definition Files)
- Storage requirements
- Configurations
- Device requirements
- Minidisk allocation
- Options that affect performance
- System libraries
- Programming language support
- Access method support

- Virtual machine operating systems other than CMS
- National Language Support.

Before installing VM/SP there are many things that must be thought out. Also, there are things that can be done in advance that make the installation proceed more smoothly. This chapter briefly discusses a few of the more important aspects of planning for VM/SP installation.

What VM/SP Is

The Virtual Machine/System Product (VM/SP) is a program product that manages a real system. All of its resources:

- Main (IPL) processor
- Attached (non-IPL) processor
- Storage
- I/O devices.

are provided for many users at the same time. Each user of VM/SP perceives a real, dedicated system. All properties of the system appear solely devoted to the user's machine from the user's perspective. But VM/SP only simulates those properties for each user. The user's sole control of the system exists only as one of many functions of VM/SP. The appearance of this exclusive control over the system is created by VM/SP. Only the essence or echo of a real, dedicated machine is displayed giving rise to the term *virtual machine*. Figure 1 on page 6 shows an example of a VM/SP system.

VM/SP has the following components:

1. The Control Program (CP), which controls the resources of the real processor to provide for many virtual machines.
2. The Conversational Monitor System (CMS), which provides a wide range of terminal user dialog and time-sharing services. Using CMS, you can create and manage files, and compile, test, and run application programs.
3. The Interactive Problem Control System (IPCS), which provides VM/SP installations with an interactive online facility for reporting and diagnosing software failures, and for managing problem information and status.
4. The Group Control System (GCS) (an optional component), which supports a virtual machine group operating environment. Members of the group share common storage space, a common virtual machine supervisor, and the ability to communicate with each other. GCS is required if you plan to install RSCS (Remote Spooling Communications Subsystem) Version 2 or SNA (Systems Network Architecture) products.
5. The Transparent Services Access Facility (TSAF) (an optional component), lets an end user application connect to a resource (such as a data base). This is done without knowing the actual user ID and node ID where that resource resides.
6. Procedures Language/VM is the component that contains the VM/SP System Product Interpreter, which processes the Restructured Extended Executor (REXX) high-level programming language.

7. The AVS (APPC/VM VTAM Support) is an optional component that works with the Virtual Telecommunications Access Method (VTAM) product. This component lets programs in a TSAF collection communicate with programs in a Systems Network Architecture (SNA) network.

The processors that VM/SP supports are listed under "Processors" on page 16. The real processor must:

- Have the Dynamic Address Translation (DAT) feature (a hardware service that translates virtual storage addresses to real storage addresses) and the System Timing facility
- Operate in extended control mode (a mode in which all the features of a processor function, including DAT).

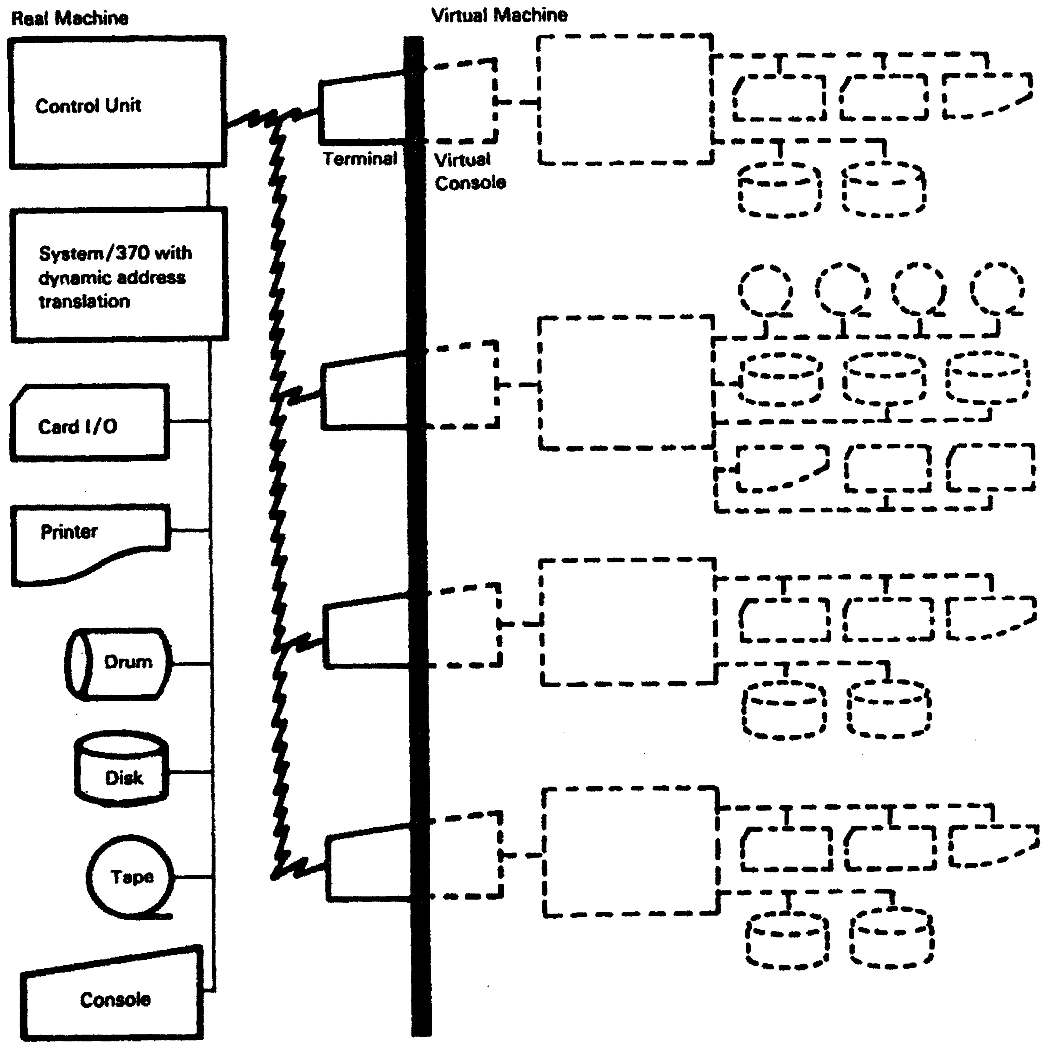


Figure 1. VM/SP System

Virtual Machine Operating Systems

As stated earlier, CP handles the work of many virtual machines. It also manages the work flow within each virtual machine. This allows each virtual machine the freedom to run a different operating system or different releases of the same operating system.

The operating systems that can run in virtual machines are:

Batch or Single User Interactive	Multiple-Access
OS/VS1	VM/SP
VS1/BPE	VM/Systems Extensions
OS/VS2 SVS	VM/Basic System Extensions
OS/VS2 MVS	Time Sharing of OS/VSE with VSE/ICCF (5746-1 TSI)
RSCS V1	IX/370
CMS	VM/SP HPO
GCS	MVS/SP™
	VSE/SP V1
	VSE/SP V2
	VSE/SP V3
	VM/370
	AIX/370
	Conversational
	CMS, GCS, RSCS

CP provides some of these with virtual device support and virtual storage. However, they are limited by the restrictions listed in Appendix B, "VM/SP Restrictions" on page 411.

Planning Storage Requirements

Planning for VM/SP storage requirements falls into two general categories:

- Real storage planning
- Direct access storage device (DASD) planning.

Many factors affect how much storage of each kind any given system might require.

Some of the factors that affect *real storage* requirements are:

- Nucleus size
- Number of real system devices, control units, and channels
- Types of system devices
- Number of virtual machines defined in the system directory.

Some of the factors that affect *DASD* requirements are:

MVS/SP is a trademark of the International Business Machines Corporation.

- Nucleus size
- Maximum number and size of logged on virtual machines
  - Paging space
  - Spooling space
  - CMS minidisks.
- Operating systems running in the virtual machines
- Other facilities
- Applications
  - Error recording
  - System restart
  - System directory
  - Access method support
  - Interactive Problem Control System (IPCS)
  - Saved systems
  - Applications.

Note: See Chapter 3, "Estimating VM/SP Storage Requirements" on page 41 for more information on storage requirements.

Introduction to VM/SP System Generation

System generation is a procedure that creates a VM/SP system tailored to your needs. The first step in the procedure restores a sample working copy of a basic VM/SP system, called the *starter system.* Then, use the starter system to create a VM/SP system configured to your own hardware. You also describe your DASD volumes and define how they are to be used.

The following versions of the VM/SP starter system can be ordered:

- 3350
- 3375
- 3380
- 9313/9332
- 9335/3370.

Each starter system must be restored to a similar disk. For example, a 3380 starter system is restored to a 3380 disk. Once restored, though, all starter systems can build any supported system residence volume type.

Before beginning the system generation procedure, review the following:

- Know which devices to include in your VM/SP system
- Decide how many virtual machines to define
- Know what system National Language(s) you want available on your VM/SP system
- Select the volumes to be owned and used by CP for system residence, paging, spooling, CMS minidisks, Saved Segments, and other applicable second level systems
- Know what the licensed program requirements are
- Compute the amount of real storage available to VM/SP and the amount of DASD required

- Define the user identity of the real system operator
- Know how to code the macro statements that define your system (See Part II of this book.)
- Examine the step-by-step outline of the procedure. (See the *VM/SP Installation Guide*.)

Then, tailor the following System Definition files:

- Real I/O configuration file (DMKRIO) listing your I/O devices. (To attach a Mass Storage System (MSS) to VM/SP, coordinate the Real I/O configuration file with the Mass Storage Control (MSC) tables.)
- VM/SP directory file (VMUSERS DIRECT) describing the virtual machines
- CP system control file (DMKSYS) describing CP-owned volumes, the real storage size, and so on
- System name table (DMKSNT) describing the name and location of saved systems and physical saved segments, 3800 printer image libraries, 3704/3705 control programs, and CP message repositories
- CMS nucleus generation profile (DMSNGP ASSEMBLE) defining parameters for the CMS nucleus
- Your own forms control buffer (module DMKFCB) if you wish. (This module is supplied with the product tape.)

Once you have defined your VM/SP system with the System Definition files, you can begin to generate your system. You should, however, read the rest of Part I to be sure nothing else is needed for your specific situation.

Preparing the Files That Describe Your VM/SP System

A major task in getting ready to install VM/SP is the preparation of files that define your VM/SP system. These files are called *system definition* files. Before you start to generate a system on a real machine, you must tailor three System Definition files. These three files describe the VM/SP system you plan to generate. Also, you may choose at this time to build two optional files. (See Figure 2 on page 10.)

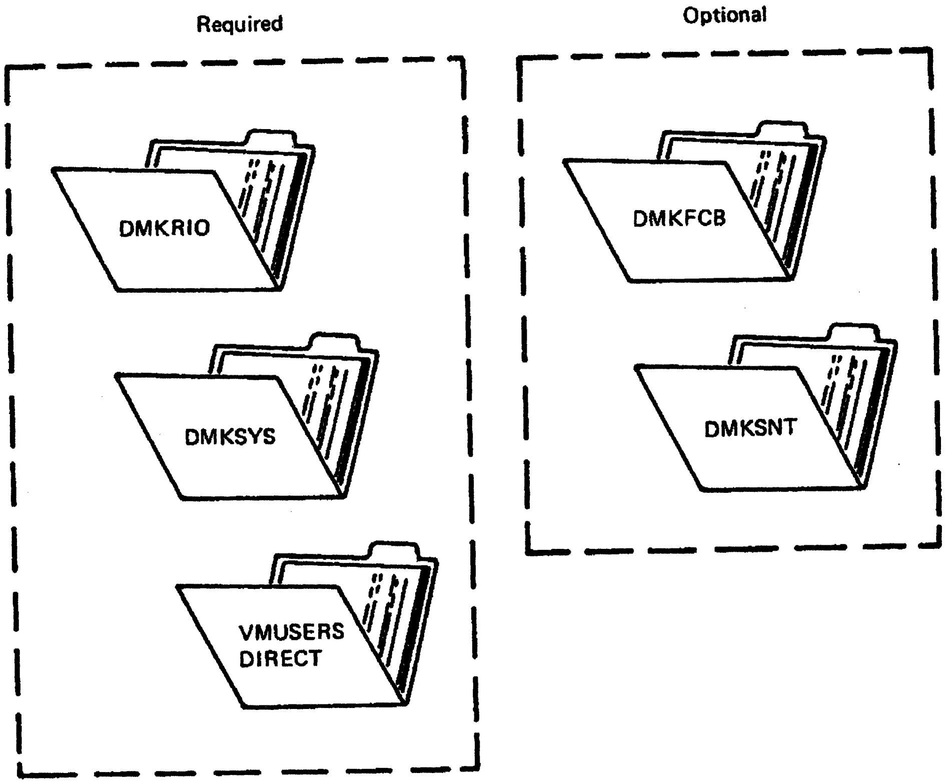


Figure 2. Required and Optional Files for System Generation

To tailor these files, you can use:

- Files created with the System Product Editor
- Existing VM/SP files altered with the System Product Editor
- Other VM/SP or VM/370 system files on the product tape
- Card input
- Other system files on tape in a card-to-card image.

Use a CMS command such as READCARD, VMFPLC2 LOAD, SPLOAD, MOVEFILE, or TAPPDS to bring files into the system.

Depending on the installation method used, these files can be ready to install at system generation time. For example, you can prepare control statements to create the system directory in advance. You can plan for changing the copies of system definition files supplied with VM/SP so that they require little time during installation.

CP System Integrity

CP system integrity provides access to the operating system for only authorized users. The system is designed, utilized, and maintained to avoid the compromise of security controls. In other words, VM/SP CP system integrity prevents illegal access to the system. Without the customer's knowledge and permission, no guest operating system mechanism or virtual machine program not authorized by CP should be able to:

- Circumvent or disable the Control Program main or secondary storage protection
- Access a Control Program (CP) password protected resource

- Use restricted passwords to access the system
- Obtain control in real supervisor state
- Obtain privilege class authority or directory functions greater than those it was assigned
- Circumvent the system integrity of any guest operating system such as MVS or VM/SP. (Operation of any VM/SP CP facility implies system integrity for the guest.)

The following special terms help describe the soundness of your system:

Main Storage Protection: CP isolates one virtual machine from another via hardware DAT.

Secondary Storage Protection: CP isolates minidisk and virtual disk extents via channel program translation.

Password Protected Resource: CP protects resources via logon passwords and minidisk passwords.

Directory Capabilities: Options are selected to control functions restricted for specific assignment(s). Functions not usually granted to general users and ones that permit bypassing system integrity controls are just two examples.

Guest Operating System: Another system control program is used while operating under VM/SP CP.

Note: VM/SP system integrity applies to class G users only.

MVS Guest Machine Environment

VM/SP system integrity applies to the following environments for MVS guest machines only:

- V=R with the NOTRANS option
- V=R with the Shadow-Table-Bypass SET command option
- Preferred Machine Assist
- Single Processor Mode.

However, a user or program on an MVS guest machine allowed to bypass system integrity controls can also bypass those built into VM/SP. Under these conditions, the customer is solely liable for any breach of security. Safeguards have to be taken to ensure that:

- Required MVS system integrity controls are installed
- Authorized programs and users are properly controlled.

Note: VM/SP CP system integrity infers no protection of data between multiple users of a single CMS batch system.

Customer Responsibilities for CP

The customer is the only one who can answer for the security of the customer's data. For system integrity to be meaningful, proper use of security controls is essential.

Some areas where effective controls should be used are:

- Password protection
- Assignment of suitable privilege classes
- Assignment of directory options

- Set up and authorization of guest virtual machines.

Specific actions and restrictions may vary, depending on system resources and conditions. The customer must take proper steps to select, apply, and implement these actions and restrictions. Moreover, they must be thorough enough for adequate security, ensuring complete control by the customer.

Note: IBM will accept APARs that describe exposures to the system integrity of VM/SP. For instance, problems caused by a program running in a virtual machine not authorized by a mechanism under the customer's control can expose the system's integrity. A customer who discovers a system integrity problem or exposure should report it to the Customer Support Center (ISG Level 1).

GCS System Integrity

An operating system has system integrity when it is designed, implemented, and maintained to protect itself against unauthorized access, to the extent that security controls specified for that system cannot be compromised. GCS system integrity prevents any unauthorized program running under the control of the GCS supervisor from:

- Bypassing store or fetch protection
- Bypassing GCS authorization controls, and/or
- Obtaining control in virtual supervisor state or with a PSW protection key other than the one assigned to the unauthorized program.

GCS system integrity is enforced by running unauthorized programs in problem state, and by checking all parameters, parameter lists, and addresses passed to it for validity. Because authorized programs are given special capabilities which may allow system integrity controls to be passed, they are not subject to the GCS system integrity definition explained earlier.

In GCS, an authorized program is defined as a program that runs in an authorized state. This means GCS runs with a capability to obtain a system key (protection key other than 14), and/or in virtual supervisor state. An unauthorized program is a program that executes in problem state and PSW protection-key 14.

Customer Responsibilities for GCS

The protection of the customer's data remains the customer's responsibility. For system integrity to be assured, proper use of security controls is essential.

The following controls should be considered when using GCS:

- Restrict authorized machines and programs
- Restrict certain CP commands.

The CP commands BEGIN, DISPLAY, DUMP, STORE, VMDUMP, PER, ADSTOP, and TRACE let users view or alter common storage. These CP commands should only be permitted to users who are responsible for maintaining and debugging the system. For more details on the CP commands and restructuring of the privilege classes, see “Controlling Access to CP Commands” on page 95.

Note: IBM accepts APARs that describe exposures to the system integrity of VM/SP GCS. IBM also accepts APARs that describe problems encountered when a program, running in a GCS virtual machine (not authorized by a mechanism under the customer's control), introduces an exposure to VM/SP or GCS system integrity.

Chapter 2. Planning Device Configurations for VM/SP

Contents of Chapter 2

Introduction

Establishing the device configuration for a VM/SP system, although not difficult, does require some careful planning. Before you generate a VM/SP system, make sure you have the minimum configuration supported by VM/SP and the features and facilities VM/SP requires. The minimum configuration serves as a good starting point.

VM/SP Minimum Configuration

The following table shows the minimum configuration VM/SP supports (based on a starter system):

Number Needed	**Device Type**
One	Processor (2MB/4MB Storage)
One	System Console Device
One	Printer
One	Card reader[1]
One	Card punch[1]
Two	Spindles of direct access storage
One	9-track magnetic tape unit
One	Multiplexer channel
One	Selector or Block multiplexer channel

To determine the amount of real storage and direct access storage necessary for a configuration, see Chapter 3, "Estimating VM/SP Storage Requirements" on page 41.

[1] This device is not needed for a cardless system.

Figure 3 shows a sample VM/SP configuration.

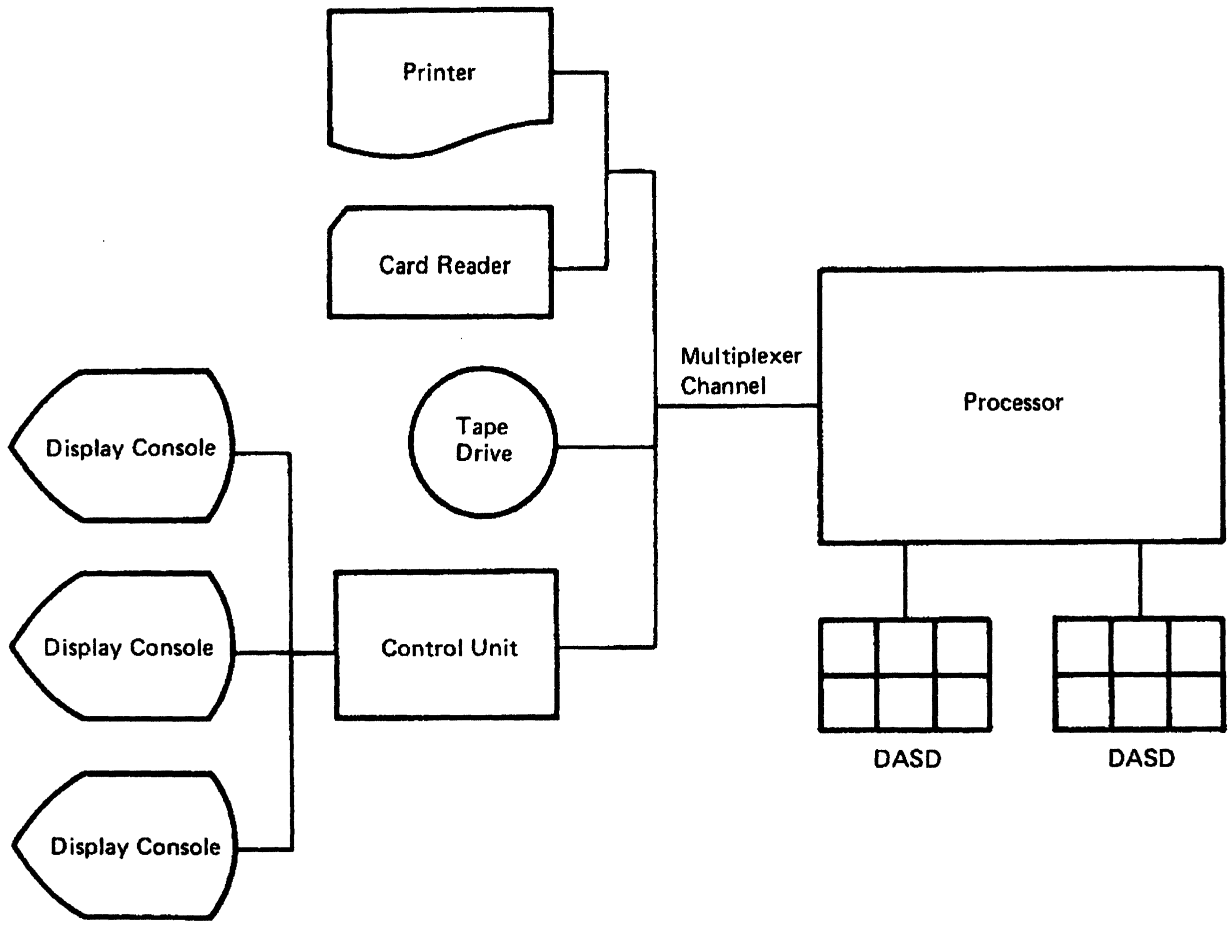


Figure 3. VM/SP Sample Configuration

Configurations Supported by CMS

CMS supports:

- Virtual storage size: minimum of 256K bytes, almost 16 million bytes in multiples of 4K.
- Virtual console: any terminal supported by VM/SP as a virtual machine operator's console.
- The same unit record devices (card readers, punches, and printers) supported by VM/SP as spooling devices, except the 2520 Punch. See "Unit Record Devices" on page 23.
- Up to 26 logical 3340, 3330 Model 1, 2, or 11, 3333 Model 1 or 11, 3350, 3375, 3380, or FB-512 DASDs. See Appendix B, "VM/SP Restrictions" on page 411 for the maximum size of a CMS minidisk.
- Up to sixteen 2400, 2415, 2420, 3410 (9 track only), 3420 (7 or 9 track), 3422, 3430, 3480 (18 track), or 8809 (9 track only) magnetic tape units.

In addition to the preceding, CMS users may be authorized to use DASD space within CMS Shared File System (SFS) *file pools.* A file pool is a collection of minidisks managed by a virtual machine. One file pool can contain the files for many CMS users. Virtual machines that manage file pools are called *file pool server machines.*

Any number of file pools can be defined on your VM/SP system. A CMS user can be authorized to use space in more than one file pool.

Devices Supported by VM/SP

VM/SP supports the following devices except as otherwise noted:

- Processors
- Direct access storage devices
- Magnetic tapes
- Unit record devices (printers, readers, and punches)
- Terminals
- Transmission control units and communication controllers
- Other devices (your IBM Marketing Representative can provide you with a complete list of the devices that are currently supported by VM/SP).

  This section does not include SNA supported devices. For information about SNA devices, see the VM/VTAM Communications Network Application (VM/VCNA) books listed in the *Related Publications* section on page 449.

Processors

VM/SP supports these IBM processors:

Processor	Model No.
System 370	135 Submodel 3 138 145 Submodel 3 148 155 II 158 UP/AP/MP 158 Submodel 3 165 II 168 Submodel 3, UP/AP/MP
43xx	4321 4331 All models 4341 All models 4361 All models 4381 All models
30xx	3031 UP/AP 3032 UP 3033 UP/AP/MP 3042 AP Model 2 3033 Model Groups N and S 3081 Model D16

Processor	Model No.
937x	9373 Model 20 9375 Models 40 and 60 9377 Model 90

Note: VM/SP does not support the 3090, 3083, 3084, or other 3081 models.

Processor Required Features and Facilities

VM/SP requires the processor features and facilities in the following list. The list includes only features and facilities that are not standard on a particular processor. For example, the Word Buffer feature is standard only on the Model 148; therefore, the feature number and requirements are described only for the Models 145 and 145-3.

- System Timing Facility (No. 2001), which includes the clock comparator and the processor timer, on the Models 135 and 145
- Clock comparator and processor timer (No. 2001) on the Model 145-3
- Floating Point feature
  - Model 135, Feature No. 3900
  - Model 145, Feature No. 3910.
- Extended Precision Floating feature (No. 3840) on the Model 135-3
- Channel Indirect Data Addressing feature on each of the 2860, 2870, and 2880 stand-alone I/O channels on the Model 165 II or 168
  - For 2860, Features Nos. 1861, 1862, and 1863
  - For 2870, Feature No. 1861
  - For 2880, Features Nos. 1861 and 1862.

  Note: The stand-alone channels that attach to the System/370 Models 165 II and 168 require that the Channel Indirect Data Addressing feature be ordered as a separate feature. This is to ensure proper operation of the I/O channels in a DAT environment.
- Word Buffer feature (No. 8810) on System/370 Model 145-3. Model 145 also requires it if:
  - A 2305 Model 2 Fixed Head Storage device is attached
  - A 3340, 3344, or 3350 Direct Access Storage Facility is attached
  - A 3330 configuration includes an integrated file adapter and two selector channels, or three or more selector channels.

    Note: This feature is recommended for selector channels if 3330, 3340, or 3350 devices are attached.

Desirable Features

The following processor features are desirable for VM/SP:

- Virtual Machine Assist (VMA) - improves performance of VM/SP systems that run guest operating systems. "Improving Performance" on page 160 describes how various VM/SP processors support virtual machine assist and VM/370:ECPS.

- Extended Control Program Support (ECPS) - improves performance of VM/SP through CP assist and expanded virtual machine assist capabilities
- Extended floating point - improves running of programs that use Extended Floating Point instructions under VM/SP on Models 135, 155 II, and 158. The feature numbers for each model are:
  - Model 135, Feature No. 3840
  - Model 155 II, Feature No. 3700
  - Model 158, Feature No. 3700.
- APL Assist - helps performance when used with the VS APL program product. It is available as hardware Feature No. 1005 on System/370 Models 135 and 145.
- Conditional swapping - provides additional instructions needed to run VTAM programs. It is available as Feature No. 1051 on System/370 Models 135 and 145.
- Advanced Control Program Support - provides additional instructions needed to run MVS (OS/VS2 Release 2 and above) and/or VTAM. It is available only on System/370 Model 145 as Feature No. 1001.

  Note: The Conditional Swapping feature and the Advanced Control Program Support feature are mutually exclusive.
- ECPS Expansion (No. 1601) - increases performance when MVS is run along with VM/SP. It allows concurrent operation of ECPS:MVS and ECPS:VM/370 and includes the functions of the Shadow Table Bypass Assist. It is available on the 4341 Processor Model Group 2 and 12).
- Channel-to-Channel Adapter (No. 1850) interconnects two channels (either S/360, S/370, or 43xx processors). Only one of the processors requires this feature.
- 3088 Multisystem Channel Communication Unit (MCCU) - I/O device used to interconnect as many as eight processors using block multiplexer channels. The 4341, 4381 303x, 3042 Attached Processor Model 2, and 308x processors support the 3088.
- Data streaming (No. 4850) - modifies the first two block multiplexer channels of a channel group to permit each to operate at a higher data rate. This feature is standard on the 3081, 4341, and 4381 processors. It is available on the 303x and 3042 AP-2 processors. See the processor manuals to see which channels support data streaming.

Storage Devices

VM/SP supports these IBM direct access storage devices (DASDs) as system residence, paging, and spooling devices and as virtual devices for virtual machines. VM/SP supports all of these as dedicated devices.

Storage Device	Model No.
2305 Fixed Head	1 and 2
3310 Direct Access	-
3330 Disk	1,2, and 11
3333 Disk and Control	1 and 11
3340 Direct Access[2]	A2,B1, and B2
3350 Direct Access	A2 and B2
3370 Direct Access	A1,A2,B1, and B2
3375 Direct Access	-
3380 Direct Access	All models
9313 Direct Access	A01/B01
9332 Direct Access	400, 402, 600, and 602
9335 Direct Access	A01 and B01

Note: CMS supports all of the storage devices mentioned in the previous table except the 2305.

Storage Control Units

VM/SP supports these IBM direct access storage control units:

- 3345 Models 3, 4, and 5 on the Models 145, 145-3, and 148 with the standard Integrated Storage Control (ISC) for the following:

Storage Device	Model No.
3330	1 and 2
3333	1 and 11
3340	A2 and 3344 Model B2
3350	A2
3990	1 and 2

- 2835 Models 1 and 2 for 2305 Models 1 and 2
- 3830 Model 1 for 3330 Models 1 and 2 only
- 3830 Model 2 for 3333 Models 1 and 11, 3340 Model A2, and 3350 Model A2
- 3830 Model 3 for 3330 Models 1 and 11, 3333 Models 1 and 11, and 3350s
- 3880 Model 1 for 3330 Models 1, 2, and 11, 3333 Models 1 and 11, 3340 Model A2, 3350 Models A2 and A2F, 3370 and 3375

[2] 3348 Data Modules, Models 35, 70, and 70F, and the 3344 Direct Access Storage, Model B2.

- 3880 Model 2 for 3330 Models 1, 2, and 11, 3333 Models 1 and 11, 3340 Model A2, 3350 Models A2 and A2F, 3370, 3375, and 3380s on the 303x processor with the Data Streaming feature (No. 4850)
- 3880 Model 3 for 3380s on the 303x processor with the Data Streaming feature (No. 4850)
- 3990 Models 1 and 2 for 3380 DASD
- Integrated File Adapter feature (No. 4655) on the System/370 Models 135, 135-3, and 138 or the Integrated Storage Control No. 4660) on the System/370 Model 145, 145-3, and 148 for the following:

Storage Device	Model No.
3330	1 and 2
3333	1 and 11
3340	A2 and 3344 Model B2
3350	A2

- Integrated Storage Control on the System/370 Models 158 and 168 for the following:

Storage Device	Model No.
3330	1 and 2
3333	1 and 11
3340	A2 and 3344 Model B2
3350	A2

Special Features Required with the 3350

Expanded Control Store Special feature (No. 2151) gives you additional control storage for microprogramming. You need this feature for 3350 disks attached to the 3830 Model 2, or for the 3345 Integrated Storage control units Models 3, 4, and 5 attached to a System/370 Model 145, 145-3, 148, 158 or 168.

If you have feature No. 2151, you must also have the Control Store Extension feature (No. 2150).

Note: You must install the Word Buffer feature (No. 8810) with the System/370 Models 145, 145-3, and 148 in order to attach a 3330, 3340, 3350 or 2305 Model 2. (Feature No. 8810 is standard on the Model 148.)

Desirable Features

3880 Storage Control Unit Buffer Features:

- Feature No. 6550 for 3380 DASDs attached to 3880 Models 2 and 3
- Feature No. 6560 for 3375 DASDs attached to 3880 Models 1 and 2

  The Speed Matching Buffer feature (No. 6550) for the 3380 supports extended count-key-data (CKD) channel programs.

If the 3380 attached to the 3880 Controller Model 3 with the Speed Matching Buffer feature (No. 6550) is part of your installation, you can run extended CKD channel programs.

Note: The Speed Matching Buffer feature is not supported for 3380 Models AD4/BD4, AE4/BE4, AJ4/BJ4, or AK4/BK4.

These features modify the direct access data transfer path by adding a buffer feature between the DASD device and the multiplexer channel. When the 3880 control unit is equipped with the respective buffer feature, the 3380 or 3375 devices can be attached to channels that operate at data rates slower than that of the DASD device.

The respective buffer features allow attachment of 3380 or 3375 devices to the following types of channels:

- 1.5MB block-multiplexer channels on S/370 Models 145, 148, 155-II, 158, 153-3, 165-II, 168, 168-3, 3031, 3032, 3033, and 3042 Model 2
- 2.0MB block-multiplexer channels on the 4341 Processor and high-speed block multiplexer channels on the 4331 Model Group 2 Processor
- 3.0MB block-multiplexer channels on 3031, 3032, 3033, and 3042 Model 2 when these processors are equipped with the Data Streaming feature (No. 4850)
- 3.0MB block-multiplexer channels on 4341, 3081, and 3083 Processors.

The speed matching operation for writing records to the 3375 DASD over a 1.5MB block-multiplexer channel requires that the data transfer across the channel and into the buffer begin before the data is transferred from the buffer to the 3375. To accomplish this with minimum loss of disk revolutions, special channel commands provide write-prenotification whenever possible.

Magnetic Tapes

VM/SP supports the following IBM magnetic tape devices and tape control units:

Device	Model No.
2401, 2402, 2403, and 2404	-
2415	1, 2, 3, 4, 5, and 6
2420	5, 7
3410/3411	1, 2, 3
3411 Tape Unit and Control	1, 2, 3
3420	3, 4, 5, 6, 7, 8
3422 Tape Unit and Control	-
3430	-
3430 Tape Unit and Control	-
3480	-
8809	-
9347	-
Tape Control Unit	
2803	-
2804	-
3411 Unit and Control	-
3422 Unit and Control	-
3430 Unit and Control	-
3480	-
3803	-

For more information on tape devices and tape control units see Appendix A, "VM/SP Configuration Aid" on page 401.

Unit Record Devices

Printers

VM/SP supports the following IBM printers:

Printer	Model No.
1403	2, 3, 7, and N1
1443	N1 (with 144 print positions)
3203	4 (only on processor models 138 and 148), 5 (Right indexing only)
3211	(in 3215 emulator mode)
3213	1, 5, and 11
3262	2 and 2C
3268	1, 1C, 2, and 2C
3287	4
3289	1, 3, and 8
3800	-
3812	-
3820	-
4245	1
4248	(dedicated only)
4250	-
5210	-
6262	-

Note that the 3268 printer is not directly supported by VM/SP.

Card Readers and Card Punches

VM/SP supports the following IBM card readers and card punches:

Device	Model No.
2501 card reader	B1 and B2
2520 card punch	B2 and B3
2540 card read punch	1
3505 card reader	B1 and B2
3525 card punch[3]	P1, P2, and P3

[3] VM/SP does not support the 3525 interpreter function.

Unit Record Control Units

VM/SP supports the following IBM unit record control units:

- 2821 Control Unit
- 3811 Printer Control Unit
- Integrated Printer Adapter (IPA) on the System/370 Model 135
- Integrated Printer Adapter Basic Control (No. 4670), and one of the following on the Models 135-3 and 138:
  - 1403 Printer Models 2 or N1 Attachment (No. 4672)
  - 1403 Printer Model 7 Attachment (No. 4677).
- Integrated 3203 Model 4 Printer Attachment, first printer (No. 8075) and optionally, second printer (No. 8076) on the Model 138 and 148.

Terminals

VM/SP supports these IBM system consoles as virtual machine consoles (in 3215 emulation mode only):

- 2150 Console with 1052 Printer-Keyboard Model 7
- 3066 System Consoles Models 1 and 2 for the System/370 Models 165 II and 168
- 3210 Console Printer-Keyboard Models 1 and 2
- 3215 Console Printer-Keyboard Model 1
- System console for the System/370 Models 138 and 148 in printer-keyboard mode (3286 printer required)
- System console for the System/370 Model 158 in printer-keyboard mode (with the 3213 Printer Model 1 required)
- 7412 Console (via RPQ AA2846) with 3215 Console Printer-Keyboard Model 1.

VM/SP supports these IBM system consoles as virtual machine consoles (in 3215 emulation mode or in 3270 mode):

- System console for the System/370 Models 138 and 148 in display mode
- System console for the System/370 Model 158 in display mode
- 3036 Console with the 3031, 3032, or 3033 processor
- 3278 Model 2A Console (in display mode) with the 4300 and 3081 processors (3287 printer optional)
- 3279 Model 2C Console (in display mode) with the 4300 processors (3287 printer optional).

Note: During system initialization only, the primary system operator's console cannot be connected to the system by a teleprocessing line.

VM/SP supports these IBM terminals as virtual machine consoles (in 3215 emulation mode only):

- 2741 Communication Terminal
- 1050 Data Communication System

- Terminals on switched lines compatible with the line control used by the Telegraph Control Type II Adapter (8-level ASCII code at 110 bps) such as the CPT-TWX (Model 33/35) terminals
- 3101 Display Terminal, Models 10, 12, 13, 20, 22, and 23 (supported as teletype Model ASR 33/35 teletypewriter)
- 3232 Keyboard-Printer Terminal Model 51
- 3275 Display Station, Model 2 with integral control unit (remote attachment only)
- 3276 Control Unit Display Station Models 2, 3, and 4 with integral control unit
- 3767 Communication Terminal Models 1 and 2 (operating as a 2741).

VM/SP supports these IBM terminals as virtual machine consoles (in 3215 emulation mode or in 3270 mode):

Terminal	**Model No.**
PC/AT/370, PC/XT/370	All Models
3180	via 3276 Control Unit Models 2, 3, and 4
3277	2, via 3272 Control Unit Model 2, (local attachment) 2, via 3271 Control Unit Model 2, (remote attachment) 2, 3, and 4 via 3274 Control Unit Models 1B and 1D (local attachment)
3278	2, 3, and 4 via 3274 Control Unit Model 1B (local attachment) 2, 3, 4, and 5 via 3274 Control Unit Model 1D (local attachment) 2, 3, 4, and 5 via 3274 Control Unit Model 1C, 41C, 51C, and 61C 2, 3, and 4 via 3276 Control Unit Models 2, 3, and 4
3279 (Color)	2A, 2B, 3A, and 3B via 3274 Control Unit Models 1B and 1D (local attachment) 2A, 2B, 3A, and 3B via 3274 Control Unit Models 1C and 51C 2A and 2B via 3276 Control Unit Models 2, 3, and 4 3A and 3B via 3276 Control Unit Models 3 and 4

Terminal	Model No.
3290 Information Panel	via 3274 control unit Models XC's and XD's

Notes:

1. Any system console or terminal that defaults to being defined to the system as a 3277 and/or 3278 will work with directory entry "CONSOLE cuu 3270."
2. A 3215 system console simulation for graphic devices excludes processing multiple output channel programs that contain CCWs without carriage returns (X'01' CCW op code) on one line of the screen. These channel programs are treated separately and VM/SP uses a new line for each one.

Special Considerations and Required Features

Terminals that are equivalent to those explicitly supported may also function satisfactorily. You are responsible for establishing equivalency. IBM assumes no responsibility for the impact that any changes to the IBM-supplied programs or products may have on such terminals.

IBM does not guarantee that an RPQ available earlier is now or will again be available. Contact your IBM branch office for ordering information concerning the RPQs mentioned in this book.

2741 Communication Terminal

The 2741 Communication Terminal can be used in almost any application calling for an electric typewriter and, in addition, incorporates the power of a computer. Principle areas of application include text processing, data entry, time sharing, and incidental calculations. The 2741 operates point-to-point only.

VM/SP supports the 2741 Communication Terminal on either duplexed switched or point-to-point nonswitched lines connected to a Western Electric 103A2 (or equivalent data set).

1050 Data Communication System

The 1050 Data Communication System is a stand-alone nonbuffered terminal consisting of a 1051 Control Unit and an option of any or all of various cable connected devices. This terminal contains an asynchronous communications adapter. The 1050 can communicate with another 1050 system or a host system cabled as an asynchronous communication adapter.

VM/SP supports the 1050 Data Communication System on either switched or point-to-point nonswitched lines.

3270 Information Display System Terminal Features

The 3271/3272 and 3274 control units are terminal control units that can attach up to 32 displays, serial matrix printers and/or line printers.

There are two categories of these terminals; Category A and Category B. The Category A terminals attach to the 3274 Control Unit while Category B terminals attach to the 3271/3272 control units. You can attach the Category B terminals to the 3274 control unit with certain limitations. A maximum of 16 of the 32 attachable terminals on a 3274 can be Category B terminals.

The Category A terminals are:

Terminal	Model No.
3262 Line Printer	3 and 13
3278 Display Station	1, 2, 3, 4, and 5
3279 Color Display Station	2A, 2B, 3A, and 3B
3287 Printer	1, 2, 1C, and 2C
3289 Line Printer	1 and 2
3290 Information Panel	1 and 2
4250 Printer	-

Note: You cannot attach the 3290 Information Panel to port zero.

The Category B terminals are:

Terminal	Model No.
3268 Printer	2 and 2C
3277 Display Station	1 and 2
3284 Printer	1 and 2
3286 Printer	1 and 2
3288 Line Printer	2

Note: Category A and Category B terminals (attachable printers) should not be confused with system printers *attachable* to CP. For a list of system printers *attachable* to CP, see the section on "Unit Record Devices" on page 23. Note also that the 3268 printer is not directly supported by VM/SP.

The basic 3271/3272 Control Unit can attach up to four devices. You can attach up to 32 devices in sets of four devices by adding up to seven device adapters (No. 3250).

The basic 3274 Control Unit can attach up to eight Category A terminals. Two categories of terminal adapters can be featured in various combinations to give you the maximum terminal configuration of 32 terminals. A maximum of 16 of the 32 terminals can be Category B units and you need at least one Category A Display Station with a keyboard for diagnostic purposes.

Terminal Adapter Type A1 through A3 (Nos. 6901, 6902, and 6903): You can install one of each of these adapters on a 3274 Control Unit. Each adapter provides for the attachment of an additional eight Category A terminals.

You must install these terminal adapters in the following sequence:

Adapter Type	Terminal Sequence
A1	9-16
A2	17-24
A3	25-32

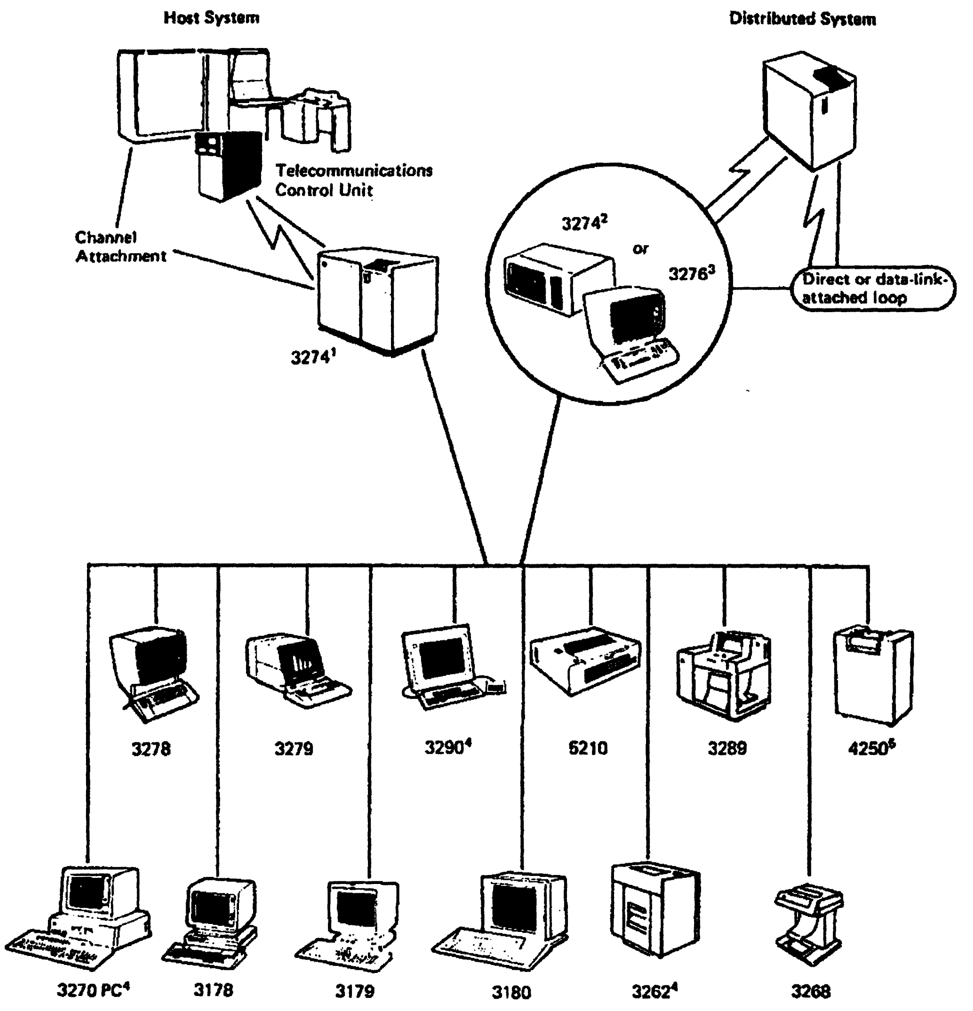
Terminal Adapters Type B1 through B4 (Nos. 7802, 7803, 7804, and 7805):
Terminal Adapter Type B1 permits the attachment of four Category B terminals to a 3274 Control Unit. It also provides for the installation of terminal Adapters Type B2, B3, and B4 when you want additional Category B terminals. Terminal Adapters Type B2 through B4 permit the attachment of four additional Category B terminals each. You can install a maximum of one each of the B1, B2, B3, and B4 adapters for a combined total of 16 Category B terminals on a 3274 Control Unit.

You must install these terminal adapters in the following sequence:

Adapter Type	Terminal Sequence
B1	1-4
B2	5-8
B3	9-12
B4	13-16

For remote bisynchronous terminals, the TERMINAL macro in the Real I/O configuration file requires that terminals on a Type A adapter must come before terminals on a Type B adapter. See Chapter 8, "Preparing the Real I/O Configuration File (DMKRIO)" on page 279 for more information on the TERMINAL macro.

Figure 4 on page 29 provides an overview of the components that can be attached in a 3270 Information Display System.



Legend:

——— Direct Connect

⚡ Communication Facility

[1] 3274 Models with up to 32 terminals: Models 21A, B, D; 41A, D (channel), and Models 21C, 31C, 41C (remote).

[2] 3274 Model 51C with up to 8 terminals, and 61C with up to 16 terminals.

[3] 3276 with up to 8 terminals.

[4] The 3270 Personal Computer, 3290 and 3262 Model 3 do not attach to a 3276.

[5] The 4250 attaches only to 3274 Models 31A, D; 41A, D.

Figure 4. Overview of the 3270 Information Display System Attachment

Note that the 3178, 3179, and 3268 devices are not directly supported by VM/SP.

See the *IBM 3270 Information Display System Library User's Guide* for more information on 3270 Information Display Terminals.

Local Configurations Supported

Control Units: The following IBM control units can be locally attached on a byte multiplexer, block multiplexer, or selector channel to support 3270 devices:

- The 3272 Control Unit Model 1 and 2 (for attachment of up to 32 Category B terminals).

These 3272 configurations require:

- Device Adapter feature (No. 3250) if more than four devices are attached to the 3272. You can attach up to four additional devices with each device adapter.
- A 3271/3272 Attachment (No. 8330) to attach each 3287 printer.

- The 3274 control units Model 1B, 1D, 21A, 21B, 21D, 31A, 31D for the attachment of up to 32 display stations and printers. All of the 32 devices can be Category A Terminals. VM/SP does not support the 3278 Display Station Model 5 with the 3274 Control Unit Model 1B. A maximum of 16 of the 32 devices can be Category B terminals.
- The 3274 Control Unit Model 41A and 41D for the attachment of up to 32 display stations and printers. All of the devices must be Category A terminals because VM/SP does not support Category B terminals.

Figure 5 shows an example of a local configuration supported by a 3274 control unit (all models but the 51C).

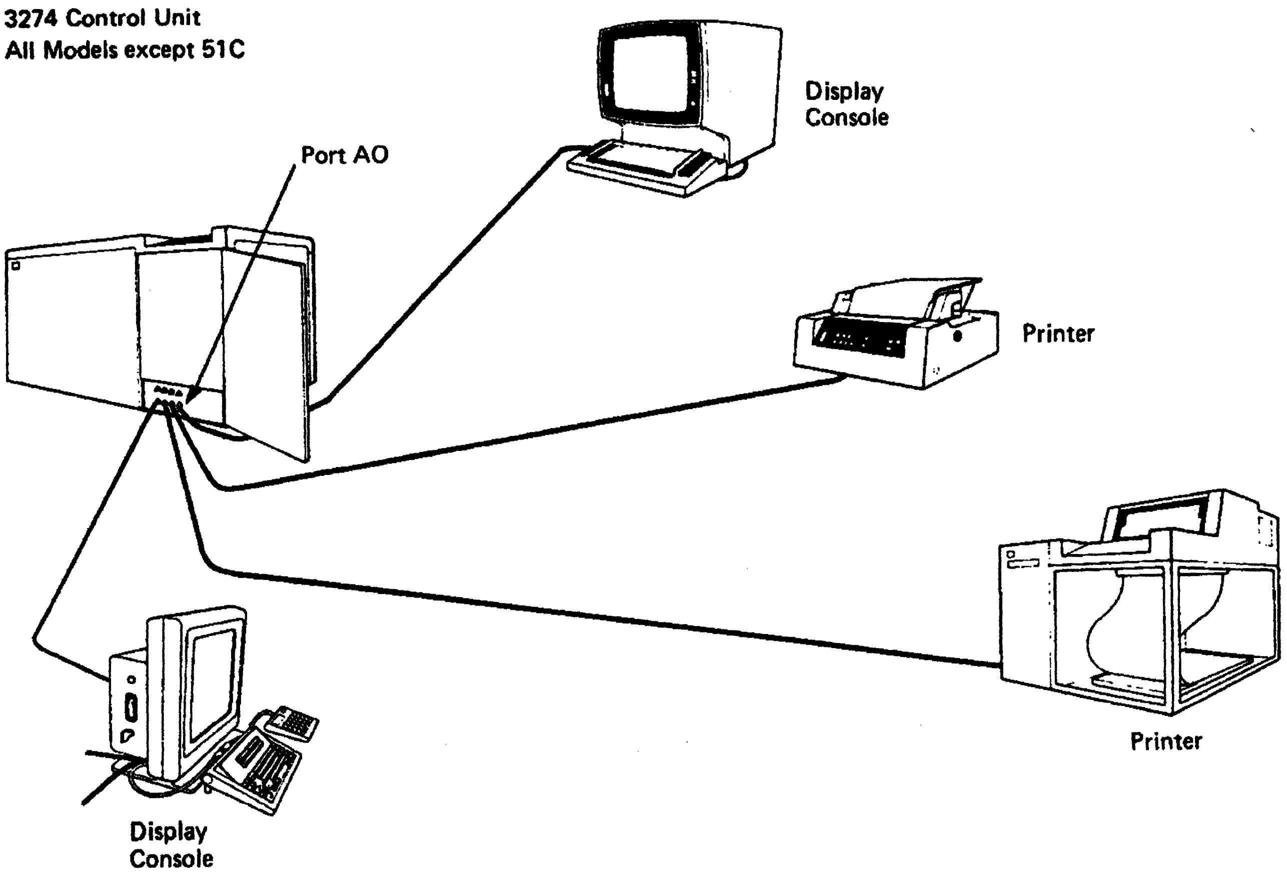


Figure 5. Local Configuration Supported by a 3274 Control Unit

Remote Configurations Supported

Control Units: The following IBM control units can be remotely attached to leased lines by a:

- 2701 Data Adapter Unit
- 2703 Transmission Control Unit
- 3704/3705/3725 Communication Controllers in emulation mode
- Integrated Communication Adapter (ICA).

3271 Control Unit Model 2 for attachment of up to 32 Category B terminals.

This configuration requires:

- Device Adapter feature (No. 3250) if more than four devices are attached to the 3271. You can attach up to four additional devices with each device adapter.
- A 3271/3272 Attachment (No. 8330) to attach each 3287 Printer Model 1 or 2
- Copy feature (No. 1550) to use the full screen copy function
- Transmission Speed feature (No. 7820 or No. 7821).

3271 Control Unit Model 11 and 12 for attachment of up to 32 Category B terminals.

3274 Control Unit Model 1C, 21C, and 31C for the attachment of up to 32 display stations and printers. All of the 32 devices can be Category A terminals. A maximum of 16 of the 32 devices can be Category B terminals.

3274 Control Unit Model 41C for the attachment of up to 32 display stations and printers. All of the devices must be Category A terminals because Category B terminals are not supported.

3274 Control Unit Model 51C for the attachment of up to 12 display stations and printers. Eight of the 12 devices can be Category A terminals. You can attach up to four Category B terminals via Terminal Adapter Type B1.

3274 Control Unit Model 61C for the attachment of up to 16 display stations and printers. All of the devices must be Category A terminals because Category B terminals are not supported.

3276 Control Unit Display Station Models 2, 3, and 4 for attachment of up to seven additional:

- 3278 Display Stations Models 2, 3, and 4. 3278 Model 3 cannot be attached to a 3276 Model 2. 3278 Model 4 cannot be attached to a 3276 Model 2 or 3.
- 3279 Color Display Stations Models 2A, 2B, 3A, and 3B. 3279 Models 3A and 3B cannot be attached to a 3276 Model 2
- 3287 Printers Models 1, 1C, 2, and 2C

  Note: VM/SP does not support extended color printing, highlighting, and programmed symbols for the 3276.
- 3289 Printer Models 1 and 2.

To support this configuration, you must have the following:

- The basic 3276 Control Unit Display Station. This contains one integral display station and can attach to one of the following:
  - 3278 Display Station Model 2, 3, or 4
  - 3279 Model 2A, 2B, 3A, or 3B. The 3279 Models 3A and 3B cannot be attached to a 3276 Model 2.
  - 3287 Printer Models 1 or 2.
- A 3274/3276 Attachment (No. 8331) is required for each 3287 Printer Model 1 or 2.
- Each 3276 requires one of the communications features (No. 6301 or No. 6302) and either the External Modem Interface (No. 3701) or the 1200 bps Integrated Modem feature (No. 5500).
- Color Display Attachment (No. 1950) can attach to 3279 Color Display Terminals (Models 2A, 2B, 3A, and 3B). This feature is not available for the 3276 Models 1 and 2. The 3276 does not support programmable symbol sets, extended color, or extended highlighting. The 3279 Models 2B and 3B are supported on the 3276 for base color. The Extended Function Base (No. 1068) is prerequisite.

The following control unit is remotely attached to either leased or switched lines by a:

- 2701 Data Adapter Unit
- 2703 Transmission Control Unit
- 3704/3705/3725 Communication Controllers in emulation mode
- Integrated Communication Adapter (ICA).

3767 Communication Terminal

The 3767 Communication Terminal contains all functions and controls in one integrally designed desk-top unit. This terminal is a keyboard and/or bidirectional printer device for entering data into and retrieving data from a processor. These functions are done through a 2701 Data Adapter Unit.

VM/SP supports the 3767 Communication Terminal, Models 1 and 2 when it operates as a 2741 Communication Terminal and is attached to a 3704 or 3705 Communication Controller. It requires special features on either switched or nonswitched point-to-point lines.

Transmission Control Units

VM/SP supports the following IBM transmission control units:

Device	Unit Type or Model
2701	Data type
2702	Transmission control
2703	Transmission control
Integrated Communications Attachment (ICA)	No. 4640
3704, 3705-I, 3705-II, 3725	Communication controllers

2701 Features

- For line control of CPT-TWX (Model 33/35) terminals and the 3101 display terminals, the Telegraph Adapter Type II (No. 7885) is required.
- For 2770, 2780, 3270, 3770 (as a 2770; 3776 also as a 3780), and 3780 terminals, the following are required:
  - Synchronous Data Adapter Type II (No. 7698)
  - EBCDIC code (No. 9060)
  - EBCDIC transparency (No. 8029).
- For 1050 and 2741 terminals, the following are required:
  - Terminal Adapter Type I, Model II (No. 4640)
  - Selective Speed, 134.5 bps (No. 9581)
  - 2741 Break Feature (RPQ No. M53193), and Break Command (RPQ No. 858492).
- The Expanded Capability feature (No. 3815) is required if there are:
  - More than two low-speed adapters (either Type I Model II, or Telegraph Type II), or
  - More than one high-speed adapter (Synchronous Data Adapter Type II), or
  - One high-speed and at least one low-speed adapter attached to the same 2701 control unit.
- The Expansion feature (No. 3855) is required for each line adapter after the first.

2702 Features

- For 1050 and 2741 terminals, the following are required:
  - Terminal Control Base for Terminal Control (No. 9696)
  - Terminal Control Type I (No. 4615)
  - Selective Speed, 134.5 bps (No. 9684)
  - Type I Terminal Interrupt (No. 8200)
  - Data Set Line Adapter (No. 3233) or Line Adapter (No. 4635), 4-wire Terminal Control Type I (No. 4615).

- For line control of CPT-TWX (Model 33/35) terminals and the 3101 display terminals, the following are required:
  - Terminal Control Base for Telegraph Terminal Control (No. 9697)
  - Telegraph Terminal Control Type II (No. 7912)
  - Pluggable End Characters (return key generates an interrupt) (RPQ No. E62920), optional
  - Data Set Line Adapter (No. 3233)
  - Terminal Control Expansion (No. 7935), required only if both of the terminal bases (Nos. 9697 and 7912) are attached to the same 2702.
- The 31 Line Expansion (No. 7955) is supported as needed.

2703 Features

- For 1050 and 2741 terminals, the following are required:
  - Start-Stop Base Type I (No. 7505) or Type II (No. 7506)
  - Terminal Control Base (No. 4619)
  - Terminal Control Type I (No. 4696)
  - Line Speed Option, 134.5 bps (No. 4878)
  - Type I Terminal Interrupt (No. 8200)
  - Data Line Set (No. 3205) and/or Line Set 1B (No. 4687).
- For line control of CPT-TWX (Model 33/35) terminals and 3101 display terminals, the following are required:
  - Telegraph Terminal Control Base (No. 7905)
  - Telegraph Terminal Control Type II (No. 7912)
  - Line Speed Option, 110 bps (No. 4877)
  - Data Line Set (No. 3205), and Data Line Set Expander (No. 3206)
  - Pluggable End Characters (return key generates an interrupt) (RPQ No. E66707), optional.
- For 2770, 2780, and 3780 Terminals, the following are required:
  - Synchronous Base (Nos. 7703, 7704, or 7706)
  - Synchronous Terminal Control for EBCDIC (No. 7715)
  - Transparency (No. 9100)
  - Synchronous Line Set (No. 7710).
- The Base Expansion feature (No. 1440) is required if more than one base type is attached to the same 2703.

Integrated Communications Attachment Features

The Integrated Communications Attachment (ICA) (No. 4640) is available on the System/370 Models 135, 135-3, and 138. Additional lines (Nos. 4722-4728) are supported.

- For 1050, 2741, and 3767 (as a 2741) terminals, the following are required:
  - Terminal Adapter Type I Model II (Nos. 9721-9728)
  - Switched Network Facility (Nos. 9625-9632), optional
  - Write Interrupt (Nos. 9745-9752)

  - Read Interrupt (Nos. 9737-9744)
  - Unit Exception Suppression (Nos. 9729-9730), optional
  - For the 3767 only, as a 2741, 200 bps (Nos. 2711-2718) or 300 bps (Nos. 9593-9600).

- For 2770, 2780, 3270, 3770 (as a 2770; 3776 also as a 3780), and 3780 terminals, the following are required:
  - Synchronous Data Adapter Type II (Nos. 9649-9656)
  - Half-Duplex Facility (Nos. 9617-9624)
  - EBCDIC Transparency (Nos. 9673-9680).

- For line control of CPT-TWX (Model 33/35) terminals and 3101 display terminals, the following are required:
  - Telegraph Adapter Type II (Nos. 9785-9792)
  - Switched Network Facility (Nos. 9625-9632).

3704/3705/3725 Features

The 3704, 3705, and 3725 Communication Controllers are supported in 2701, 2702, and 2703 emulation program mode if controlled by CP.

Note: VM/SP supports the CPT-TWX (Model 33/35) terminals at 110 bps and the 3101 display terminals at any line speed defined in EP or NCP. The defined line speed must be supported by internal clocks in 3705 or external clocks in modems used.

VM/SP supports all models of 3704, 3705, and 3725 communication controllers. The features required on a communication controller do not depend on VM/SP. Other 3704/3705/3725 features depend on the planned use of the communication controller and the type of 3704/3705/3725 control program (emulation) to be run.

VM/SP does not support the following 3704/3705 features:

Line Set Type	Number
2A	4721
3A	4731
4B	4742

Other Considerations for Planning Your Configuration

Two-Channel Switch

The two-channel switch lets a single control unit and its attached devices be accessed by two different channels on the same or different processor. At any one point in time, only one of the channels can be transmitting data to or from a processor and one of the attached *switched* devices.

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

In Figure 6, while channel B has the switch, channel A cannot access any device on the control unit (either the same one channel B is using or a different one).

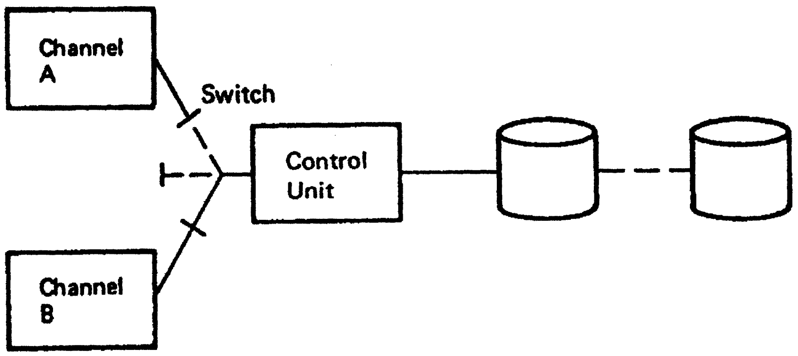


Figure 6. Two-Channel Switch

See the *VM Running Guest Operating Systems* book for more information about using the two-channel switch.

Multisystem Channel Communication Unit

The Multisystem Channel Communication Unit (MCCU) Models 1 and 2 are I/O devices that interconnect multiple systems by their block multiplexer channels. They provide a connection between any two selected channels in a network so messages or data can be exchanged. The 3088 is fully compatible with existing channel-to-channel adapters (CTCAs) and may form a loosely coupled multiprocessing system. Model 1 allows up to four processors and Model 2 allows up to eight processors to be interconnected. (See Figure 7 on page 37.)

In configurations using the 3088, you must define the device using the ADDRESS and DEVTYPE operands of the RDEVICE macro, and the ADDRESS, CUTYPE, and FEATURE=xxx-DEVICE operands of the RCTLUNIT macro.

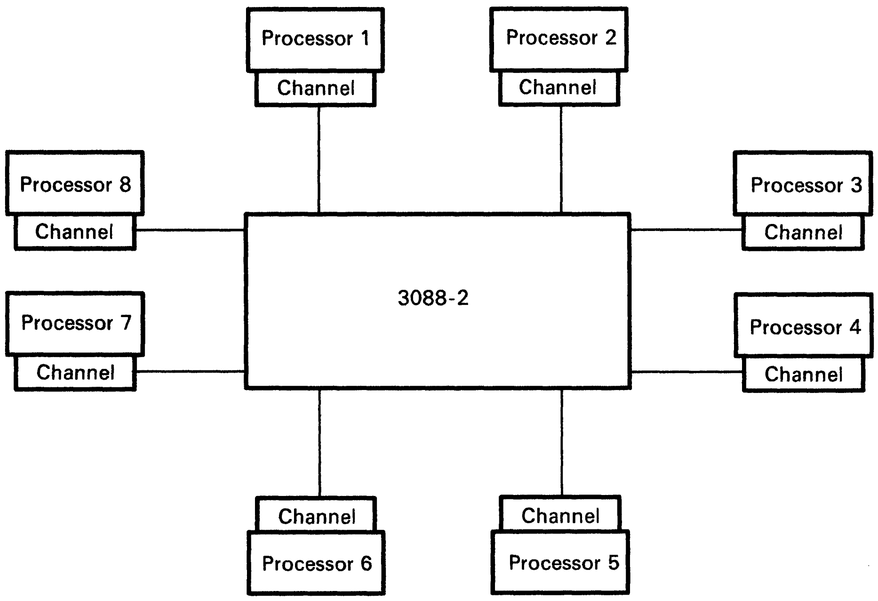


Figure 7. 3088 Model 2 Configuration

Special Hardware for National Languages

Some languages have special characters that may require certain terminal hardware. If you have a variety of languages available on your system, make sure that you have appropriate terminals to display characters in these languages. Also, all terminals on your system should be capable of displaying the character set of the system national language.

Devices Used Only by an Operating System in a Virtual Machine and Not by VM/SP

Any I/O device that can be attached to the processor can be used by a virtual machine under VM/SP as long as there are:

- No timing dependencies 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
- No violations of the other restrictions outlined in Appendix B, "VM/SP Restrictions" on page 411.

Dynamically modified channel programs (except those that have I/O involving page 0) are permitted if run in a virtual=real machine.

I/O 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.

Service Record File

On 3031, 3032, and 3033 processors, each console station of the 3036 System Console has a 7443 diskette attached to it. This diskette is usable when the console station is in Service Record File (SRF) mode. In the usual console setup, one of the processor's console stations is an operator's console, and the other console station is a service console. It is through the service console that SRF capability is provided. When one console station serves as both operator and service console, there is no SRF capability. The SRF address you specified on the RIOGEN macro when you generated VM/SP should be the address of the service record file attached to the service console.

Multiple Service Record Files

In a 3033 attached processor or multiprocessor system, there are two 3036 consoles. This configuration has four service record file devices (one console per station).

The 3033 attached processor and multiprocessor systems support more than one service record file device. For VM/SP systems operating on a 3033AP or 3033MP, specify more than one SRF device when you generate VM/SP. Code DEVTYPE=7443 in the RDEVICE macro and CUTYPE=7443 in the RCTLUNIT macro to generate support for the SRF devices. Also, code the ADDRESS=cuu operand in both macro statements. Identify the SRF device addresses in the RIOGEN macro as SRF=(cuu,cuu,...). The SRF addresses you specify in the RIOGEN macro should be the same as the addresses of the SRF devices attached to the service support consoles.

In 3033 AP or MP systems with I/O configured asymmetrically to one processor, a channel path must be available from the I/O processor to both SRF devices. This is needed to access the SRF devices in both 3036 consoles.

If an SRF device specified on the RIOGEN macro is inaccessible during initialization of the error recording cylinders, an error message is sent to the operator. Processing continues without frames from that SRF device in place on the error recording cylinders.

The RIOGEN macro produces an MNOTE warning message if you specify more than 32 SRF devices.

Processor Controller

The 3081 Processor Complex uses a processor unit and a processor controller to control system operations. The processor controller is a service processor that defines the I/O configuration to the processor complex. To do this, the processor controller requires information about the real system configuration. You define this information to the controller by running the Input/Output Configuration Program. See "Input/Output Configuration Program" on page 186 and "Considerations for Coding the Input/Output Configuration Source File" on page 324 for more information about the Input/Output Configuration Program.

Compatible Devices

The following devices function similar to the 2770 Communication System. The programming and operating manuals cited with each provide details on required features to operate such devices. These details are not contained in VM/SP manuals.

6640 Document Printer

- Programming Guide for Communicating with the 6640 Document Printer, G544-1001
- 6640 Document Printer - Communicating
  - User's Guide, S544-0507
  - Operating Instructions, S544-0506.

Office System 6 Information Processors (6/650, 6/440, 6/430)

- Programming Guide for Communicating with the Office System 6 Information Processors, G544-1003
- 6/450, 6/440, and 6/430 Information Processors - Communicating
  - User's Guide, S544-0521
  - Operating Instructions, S544-0522.

Mag Card Typewriters (Mag Card II, 6240 Mag Card)

- Programming Guide for Communicating with the Mag Card II Typewriter and the 6240 Mag Card Typewriter, G544-1005
- Mag Card II Typewriter - Communicating and 6240 Mag Card Typewriter - Communicating
  - Reference Guide, S544-0549
  - Operating Instructions, S544-1005.

Chapter 3. Estimating VM/SP Storage Requirements

Contents of Chapter 3

Real Storage Requirements for CP

Table 1 lists various CP requirements and the amount of real storage required for each.

Table 1. Real Storage Requirements for CP

CP Requirement	Real Storage Allocated
Nucleus	Approximately 260K[4]
Internal trace table	Conventionally, 4K of storage is allocated for each 256K of real storage. This storage is set aside at IPL time. See the "SYSCOR Macro" on page 340 for details of how to increase the size of the internal trace table.
Real control blocks	There is a control block for each real device, control unit, and channel: • 128 bytes/real device • 80 bytes/real control unit • 96 bytes/real channel • 40 bytes for each remote 3270 or real 3704/3705/3725.
Permanently allocated free storage (virtual control blocks and tables). For installation control of free storage, use the SYSCOR macro. For the format of this macro see "SYSCOR Macro" on page 340.	The default value is a minimum of 12K, plus an additional 4K for each 64K of real storage above 256K[5]. This storage is set aside at IPL time. Each logged-on virtual machine requires a virtual machine control block (VMBLOK), a segment table (SEGTABLE), a page table (PAGTABLE), a swap table (SWPTABLE), and a control block for each virtual device, control unit, and channel.
	The storage required is: • 784 bytes for the VMBLOK • 64 bytes/1M of virtual storage for the SEGTABLE • 40 bytes/64K of virtual storage for the PAGTABLE • 136 bytes/64K of virtual storage for the SWPTABLE • 72 bytes/virtual device • 40 bytes/virtual control unit • 48 bytes/virtual channel.

Note: Remember, this is only an example. To determine your real storage requirements, the following areas may be examined:

- I/O requirements
- IUCV activity
- Number of users
- SNA requirements
- Other information related to your system's use.

[4] An additional 40K of real storage is allocated in AP or MP mode.

[5] An additional 25% of free storage is allocated in AP or MP mode.

For example, if you have:

1M of real storage
29 real devices
6 real control units
3 real channels

and 12 virtual machines defined, each with:

1 virtual reader
1 virtual printer
1 virtual punch
3 virtual disks
3 virtual channels
1 virtual machine console
3 virtual control units
1024K of virtual storage

you would estimate CP real storage requirements as follows:

260K	for the CP resident nucleus
16K	for the CP internal trace table (see "SYSCOR Macro" on page 340)
4K	for the real control blocks, calculated as follows:
	104 X 29 = 3016 bytes for the real devices
	80 X 6 = 480 bytes for the real control units
	96 X 3 = 288 bytes for the real channels
	the sum is: 3016 + 480 + 288 = 3784 bytes (approximately 4K)
60K	for permanently allocated free storage (default value)
340K	real storage required

Also, as each of the 12 (1024K) virtual machines defined logs on, approximately 3.0K of real storage is allocated to each from the permanently allocated free storage. A breakdown of the 3.0K real storage (in bytes) is:

784	- VMBLOK
84	- SEGTABLE
418	- PAGTABLE
1,418	- SWPTABLE
72	- a virtual reader
72	- a virtual printer
72	- a virtual punch
216	- three virtual disks
144	- three virtual channels
72	- a virtual machine console
120	- three virtual control units
3,472	total bytes for each of the logged-on users defined

See "Specifying a Virtual = Real Machine" on page 161 for an example of estimating storage requirements and determining the maximum virtual = real area size.

Reducing the CP Nucleus Size

You can use the Small CP option to reduce the size of the CP nucleus. This option removes specific types of support, such as virtual = real, attached processor and/or multiprocessor support, and support for the following:

Table 2. Small CP Option Table

Support	Number of Bytes*	Modules Removed
MVS Guest	7,368	DMKFPS, DMKQVM, DMKVSC[6]
SNA(CCS)	23,283	DMKVCP, DMKVCR, DMKVCT, DMKVCV, DMKVCX, DMKVCQ, DMKVCS, DMKVCU, DMKVCW, DMKVCY
TTY terminal support	461	DMKTTZ
3066	1,536	DMKGRH
Remote 3270	17,055	DMKRGA, DMKRGB, DMKRGE[7], DMKRGC, DMKRGD
3340 Alternate Track	1,762	DMKTRK
3375/3380	5,560	DMKDAD
3704/3705/3725	6,305	DMKRNH
3850 MSS	6,058	DMKSSS, DMKSSU, DMKSSV
3800 printers	6,125	DMKTCS, DMKTCT
*Approximate		

Removal of the support listed in the preceding table reduces the CP resident nucleus by approximately 69,400 bytes. To select the Small CP option, you need to change the CP build list (loadlist) specified in the VM/SP Product Parameter File (5664167E $PPF) from CPLOAD to CPLOADSM.

Note: The CPLOAD and CPLOADSM build lists are described in the *VM/SP Service Guide*. If you want a smaller CP nucleus, but require some of the function removed by using the Small CP option, you can tailor the build list to your own specifications. Edit the CPLOAD build list to remove only modules that are associated with functions that you do not require. (See Table 2.) The CP nucleus will be reduced by the amounts shown in the Small CP option table.

The graphic device support for locally attached terminals is handled by the modules DMKGRF, DMKGRE, DMKGRF, and DMKGRI. Removal of local graphics support is not a part of the Small CP option. If you do not require local graphics support, you may remove the preceding modules and reduce the CP resident nucleus by approximately 14,600 bytes.

[6] Removal of module DMKVSC also removes V = R support.

[7] Both remote 3270 and SNA(CCS) Support use the DMKRGE module. This module can only be removed if both areas of support are being removed.

Removing Modules from the Loadlist

Caution should be exercised before removing any modules from the loadlist. If you generate a system that includes a device in the I/O configuration, you cannot remove the modules associated with that device from the loadlist. If you do remove those modules, unpredictable results may occur.

The following table shows what names are undefined during the VMFLOAD procedure if certain modules are removed.

Module(s) Removed		Name(s) Undefined			
DMKTRK		DMKTRKIN	DMKTRKFP	DMKTRKVA	
DMKRNH		DMKRNHIC	DMKRNHND	DMKRNHTG	
		DMKRNHIN	DMKRNHCT		
DMKRGA	DMKRGB	DMKRGACC	DMKRGAIN	DMKRGATM	DMKRGASP
DMKRGC	DMKRGD	DMKRGA2	DMKRGBCL	DMKRGBCO	DMKRGBEN
DMKRGE		DMKRGBFM	DMKRGBIC	DMKRGBMT	DMKRGBPL
		DMKRGBRE	DMKRGBSL	DMKRGBSN	DMKRGBUP
		DMKRGDOB	DMKRGDOI	DMKRGEIO	DMKRGESK
DMKSSS	DMKSSU	DMKSSSAS	DMKSSSHR	DMKSSSMQ	DMKSSSVM
DMKSSV		DMKSSSCA	DMKSSSL1	DMKSSSNS	DMKSSUCF
		DMKSSSCV	DMKSSSL2	DMKSSSNV	DMKSSUI1
		DMKSSSDE	DMKSSSL3	DMKSSSRL	DMKSSUI2
		DMKSSSEN	DMKSSSLN	DMKSSSVA	DMKSSULO
		DMKSSSSQ	DMKSSVHV	DMKSSVUS	
DMKFPS	DMKVSC	DMKQVMRT	DMKQVMEP	DMKVSCVR	DMKQVMTS
DMKQVM		DMKQVMCU	DMKFPS	DMKVSCAN	DMKQVMCP
		DMKQVMRS	DMKQVMRX	DMKQVMVP	
DMKVCP	DMKVCR	DMKVCPIL	DMKVCRFW	DMKVCTER	DMKVCTCN
DMKVCT	DMKVCV	DMKVCVCE	DMKVCRNR	DMKVCTLO	DMKVCTEN
DMKVCX	DMKVCQ	DMKVCTTM	DMKVCXIO	DMKVCRRD	DMKVCTCH
DMKVCS	DMKVCU	DMKVCTDA	DMKVCXD2	DMKVCRTY	DMKVCVER
DMKVCW	DMKVCY	DMKVCVEB	DMKVCVIN	DMKVCVIX	DMKVCVKS
		DMKVCVLD	DMKVCVLY	DMKVCVND	DMKVCVUT
		DMKVCXFU	DMKVCXGF	DMKVCXGO	DMKVCXOR
		DMKVCXOX	DMKVCXSA	DMKVCQAT	DMKVCQEX
		DMKVCQRE	DMKVCQSR	DMKVCSFS	DMKVCSMO
		DMKVCSWT	DMKVCUIL	DMKVCWCN	DMKVCWQS
		DMKVCWRM	DMKVCWSV	DMKVCYMT	
DMKDAD		DMKDADER			
DMKTTZ		DMKTTZLF			
DMKTCS	DMKTCT	DMKTCSET	DMKTCSTR		
		DMKTCSCO	DMKTCSSP		
		DMKTCSML	DMKTCTET		
DMKGRH		DMKGRHIN			

Notes:

1. If you do not use the NAMENCP macro statement, label DMKSNTRN is undefined during the VMFLOAD procedure.
2. If you do not use the NAME3800 macro statement, label DMKSNTQN is undefined during the VMFLOAD procedure.
3. If you do not have a V=R defined, label DMKSLC is undefined during the VMFLOAD procedure.

Direct Access Storage Requirements for CP

In the following paragraphs and in "Part II. Defining Your VM/SP System," there are many references to *DASD space*. With the support of fixed block DASD architecture, it is important to understand the fundamentals of *DASD space* to avoid confusion when dealing with various DASD types.

It is helpful to understand CP's requirements for DASD space in general. It is also helpful to understand the differences and similarities between CP's view of count-key-data (CKD) DASD and fixed block (FB-512) devices. For example:

| CKD - (3330, 2305, 3340, 3350, 3375, and 3380)

| FB-512 - (3310, 3370, 9313, 9332, and 9335)

CP's reference to DASD space is always done in units called DASD pages. A DASD page is 4096 bytes of contiguous DASD storage. This means that CP requires that all its DASD space (nucleus, error recording, warm start data, checkpoint data, directory, saved systems, dump space, paging, and spooling space) be formatted as 4096 byte records (pages). CP also requires that you identify what specific pages on DASD are allocated to each type of CP reference. For example, you must identify pages for the nucleus, for paging, for the directory, and other areas.

CP provides the Format/Allocate service program (DMKFMT) to do the formatting and allocating functions. A DASD volume containing any of the types of space listed above is called a CP volume and must be processed by the Format/Allocate service program. Space not used by CP on CP-owned volumes is available for general use. Typically, although not necessarily, this space is used for user minidisks. CP has no format requirements for this space, but does require that it be allocated as PERM if it is CP owned. If not CP owned, the default is TEMP.

Count-Key-Data Device Geometry

CP views Count-Key-Data (CKD) devices by their geometric characteristics (such as number of cylinders). Each cylinder has a fixed page capacity, meaning that a fixed number of 4K records *fit* on a cylinder. This number varies for each CKD DASD type. CP references its data by a cylinder number and a page number on that cylinder. For CP space you must format and allocate pages in groups of cylinders.

In Chapter 9, "Preparing the CP System Control File (DMKSYS)" on page 327, you are asked to figure your particular DASD requirements as a number of records or pages, then convert this number to an equivalent number of cylinders. This means dividing the page requirement by the number of pages per cylinder for particular device types. Allocating space for minidisks on CP-owned volumes is also done in units of cylinders. Use of space within this allocation is also done in units of cylinders by way of the MDISK directory control statement. This is convenient because no conversion is required in this case.

FB-512 Device Geometry

FB-512 devices appear as a linear address space of 512-byte blocks. The blocks are consecutively numbered from 0 to n, where n is the highest block number on the volume. CP groups eight consecutive blocks to form a CP page. CP then views the volume as a collection of pages numbered from 0 to (n-8)/8. For example, blocks 0-7 make up page 0. There is no concept of cylinder boundaries in this structure.

You must allocate space on FB-512 volumes in units of pages (contrasted to the unit of cylinder on CKD). When you figure your DASD space requirements as a number of pages, you can use these numbers directly in the system generation macros and in the Format/Allocate service program. No conversion to other units is required.

Although convenient for CP DASD space, this causes an inconvenience when assigning minidisks because of the difference in the unit of input between the Format/Allocate service program (pages) and the MDISK control statement (blocks). When assigning minidisk space, you must know the extents of your available space in block numbers. Be careful that you provide input to Format/Allocate in page numbers and assign minidisks by block number.

To obtain the corresponding block number, take the allocation results, which show page numbers, and multiply the page number by 8. For example, in the sample layout for a 3310 shown below, pages 2000 through 9999 were allocated as PERM space for use as minidisks. The allocation results would show:

```
PERM   2000   9999
```

This corresponds to block numbers 16000 through 79999 by doing the following:

For the beginning block number:

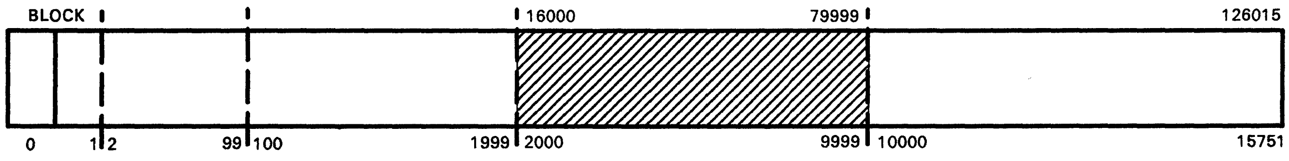
Multiply 2000 (1st page) X 8 (8 blocks per page) =
16000 (beginning block Number)

For the ending block number:

Multiply 9999 X 8 = 79992, which is the 1st block of
the last page.

Add 79992 + 7 (remaining blocks in last page) =
79999 (last block in last page)

The block numbers must be used on the MDISK statement.



Pages 0 and 1 reserved for system use (see the *VM/SP Operator's Guide* for details.)

Pages 2-99 for directory (DRCT)
Pages 100-1999 for nucleus and error recording (PERM)
Pages 2000 to 9999 for minidisks (PERM)
Pages 10000 to 15751 for paging and spooling (TEMP)

Here is an example of defining three minidisks within the range of PERM space:

```
MDISK 191 FB-512 16000 2000 LABEL
MDISK 19F FB-512 18000 5000 LABEL
MDISK 296 FB-512 23000 7000 LABEL
```

DASD Space Requirements

Table 3 shows minimum CP DASD space requirements by DASD type for the starter systems. The following paragraphs describe in detail how you determine the DASD space CP requires for the nucleus, error recording, warm start data, checkpoint data, directory, saved systems data, paging, and spooling space.

Table 3. DASD Space Requirements by DASD Type

	3350	3375	3380	FB-512
CP Nucleus	varies	varies	varies	varies
Error Recording[8]	2	2	2	128
Warm Start	2	2	2	128
Checkpoint Start	1	1	1	64
Directory	3	4	2	228
Override Space	optional	optional	optional	optional
Saved Systems	varies	varies	varies	varies
Paging Space	10	10	11	1000
Spooling Space	15	15	20	1700
Total System[9]	33 cyl	34 cyl	38 cyl	3248 pgs

CP Nucleus DASD Requirements

The CP nucleus (without a virtual=real area) requires about 350 pages of disk space for resident and pageable functions.

To determine the number of cylinders required for the CP nucleus, see the load map produced during system generation. One DASD page is required for each page of fixed and pageable nucleus. When you have a system with a V=R area, you should subtract the total number of pages of the V=R area from the last module page number to find the size of CP in pages.

For example, if the last module entry in the load map ends at page DC (hexadecimal), 350 pages of disk space are required for CP nucleus residence,

[8] The default is 2 cylinders but up to 9 cylinders may be specified by the SYSERR operand of the SYSRES macro.

[9] These figures do not include space for the nucleus or saved systems.

because hex DC converts to decimal 350. The number of cylinders required depends on the system residence device used; see the "Saved System DASD Requirements" section that follows for the number of pages per cylinder each device can accommodate.

The following table shows the number of cylinders usually required for CP nucleus residence.

Device Type	Space Required (No. of Cylinders)
3330 or 3333	4
3350	2
3375	3
3380	2

The amount of space required on FB-512 devices is equal to the number of pages as computed in the preceding section. (Subtract the total number of pages of the V=R area from the last module page number.)

Error Recording DASD Requirements

Error recording space varies from 2 to 9 cylinders and is established by the SYSERR operand of the SYSRES macro instruction as described in "SYSRES Macro" on page 333.

Warm Start Data DASD Requirements

Formulas for calculating the warm start space needed are under the discussion of the SYSWRM operand of the SYSRES macro in "SYSRES Macro" on page 333.

Checkpoint Start Data DASD Requirements

The space required for dynamic checkpointing of the VM/SP spool file system is discussed under the description of the SYSRES macro in "SYSRES Macro" on page 333.

Dump Space DASD Requirements

This space is optional. If you want to use the dump area for CP, format the area and allocate it as DUMP using the Format/Allocate program. The size should usually be large enough to contain an entire dump of real memory. If no DUMP space is allocated, spooling space (TEMP) will be used for dumps.

The following formula may be used to approximate the amount of DASD dump space needed.

$$\text{Number of pages (FBA)} = \text{number of pages real memory} + 5$$

$$\text{Number of cylinders (CKD)} = \frac{\text{number of pages real memory} + 5}{\text{number of pages/cylinder}}$$

The number of pages per cylinder for the various CKD DASD devices is listed under the heading "Paging and Spooling DASD Requirements" later in this chapter.

VM/SP Directory DASD Requirements

The VM/SP directory usually requires 2 cylinders so that it can be rewritten without disturbing the active directory and swapped after a successful update.

Before you create a VM/SP directory using the Directory program, be sure you have enough DASD space allocated as directory space (DRCT). Use the CP Format/Allocate service program to format and allocate space to be used for the VM/SP directory. The space must be allocated DRCT. To calculate the total space required, first calculate the total number of records used.

If your directory has any ACIGROUP control statements, use this formula:

$$NR = \frac{(NU+NP)}{169} + \frac{((NU+NP) \times 4) + (NM \times 3) + (NI \times 2) + NOS}{170}$$

Otherwise, use this formula:

$$NR = \frac{(NU+NP)}{169} + \frac{((NU+NP+NM) \times 2) + NOS}{170}$$

where:

NR =	Total number of records used
NU =	Number of USER control statements
NP =	Number of PROFILE control statements
NM =	Number of MDISK control statements (except for temporary disks)
NI =	Number of IPL control statements with the PARM option
NOS =	Number of other control statements (except INCLUDE statements

Then, use the following table to calculate the number of cylinders (NC) required.

Device Type	Space Required (No. of Cylinders)
3330	NC = NR/57
3350	NC = NR/120
3375	NC = NR/96
3380	NC = NR/150
FB-512	Space Required = NR

Note: Effective use of the PROFILE and INCLUDE control statements can reduce the space required for the directory.

You should initially format and allocate enough space for two VM/SP directories. You can then build a new directory whenever needed, without overlapping the current one, and without formatting and allocating space each time a new directory is created. If you wish to reallocate the area in which the directory resides, you must reallocate the DASD space and then rerun the directory program. When a VM/SP directory is written to a count-key-data DASD, space is allocated from the available cylinders on a cylinder-by-cylinder basis, and a minimum of 2 cylinders is used as DRCT space.

When a directory is written to an FB-512 DASD, the space allocation proceeds as follows:

- If there are already two DRCT extents (one in use and the other available for use) the new directory is written to the available extent. The available extent is flagged as in use, and the previously in use extent is flagged as available (the directories are swapped).
- If there is only one DRCT extent (it must be available), an attempt is made to divide it into two DRCT extents, allowing succeeding directories to be swapped. This is done as follows:
  1. If insufficient space is specified for the current directory, it is not written to the DASD volume.
  2. If sufficient space exists, the directory program attempts to divide it into two equally sized DRCT extents (one to hold the current directory and one to be available for future swapping of directories).
  3. If there is not enough space to create two equally sized DRCT extents, the current directory is built and the remaining space is reserved as available DRCT space.

If space for two directories is not initially allocated, each time you want to create a new directory, you must allocate space for the directory before you create it.

CP National Language Files DASD Requirements

This space is optional. If you plan to have more languages than the default language available on your VM/SP system, you must allocate DASD space for CP message repositories.

Each language requires its own repository file for CP messages. The feature tape for a language has two versions of this repository: a source file and a compiled object file. You must reserve DASD space for the compiled object file. Note that the LISTING file created for the complied message repository has a message that notes the total number of storage pages. Then, when you code the NAMELANG macro for a particular language, use the page number from this LISTING file on the NLSPGCT operand.

For more information about the NAMELANG macro see Chapter 10, "Preparing the System Name Table File (DMKSNT)" on page 369.

Override File DASD Requirements

This space is optional. If you plan to override the IBM-defined privilege classes, you must allocate override (OVRD) space on the same volume as your directory. The OVRD space will contain the internal override file. Use the CP Format/Allocate service program to format and allocate this space. Add 1 cylinder for count-key-data devices or 2 pages for fixed-block-address devices to your CP-owned space requirements.

Saved System DASD Requirements

To save page image copies of a virtual machine, space must be reserved on a CP-owned volume for the saved system. Saved systems require one page for each page saved, plus the additional information page(s). CP uses the information page to save the virtual machine's register contents, PSW, and storage keys. The number of extra information pages required by CP depends on the number of virtual machine pages being saved. The following table shows the relationship.

Number of Virtual Machine Pages Being Saved	Number of Information Pages Required by CP
1 - 1948	1
1949 - 3896	2
3897 - 4096	3

For example, to save a complete 16MB virtual machine, the system programmer must reserve 4099 pages on DASD.

4096 pages to be saved + 3 information pages

When coding the NAMESYS MACRO, the SYSPGCT parameter is set to the exact number of virtual machine pages being saved. Using the above example, the SYSPGCT parameter should be set to 4096. The SYSSTRT parameter indicates the starting location on the DASD where the 4099 pages for the system are being saved.

Warning

To ensure that there is no destructive overlapping of the saved system with another one being defined, the system programmer must be aware that the information pages were allocated for the named saved system.

The maximum size 3704/3705 EP image is 256K. Therefore, include 1 additional informational page in SYSPGCT for CP.

Paging and Spooling DASD Requirements

Paging and spooling space requirements are installation dependent. (The values shown in Table 3 on page 48 are examples.)

Paging space is allocated at a rate of:

Device Type	Pages per Cylinder
2305 or 3340	24
3330 or 3333	57
3350 (in native mode)	120
3375	96
3380	150

Note: The 2305 is usually used for paging only.

Paging space depends on the number and size of logged-on virtual machines, processor storage size, and workload. In general, the following calculation will yield adequate paging space.

```
number of logged-on users X average virtual machine size / 4K
= number of pages in process
```

where:

`number of logged on users`	`= the number of users logged on the system`
`average virtual machine size`	`= the average size of the logged on virtual machines`
`4K`	`= the number of bytes in one page`
`number of pages in process`	`= the number of 4K pages available in the processor`

Thus, if you have 10 logged-on users with an average virtual machine size of 500K (125 4K pages) you would need 1250 pages of DASD space for paging (10 x 125 = 1250). This would take 9 cylinders of a 3380 (1250/150 = 8.3). Spooling data is placed in a 4K buffer with the necessary channel programs required for each record. Data capacity of spooling cylinders thus varies with the data and CCWs used.

The examples in Table 3 on page 48 assumed a maximum of 200 spool files of 8-9 blocks each. If separate DUMP space is not allocated, spooling space (TEMP) will be used for dumps.

The primary system operator is warned when the paging/spooling space becomes 90% full. The *VM/SP System Messages and Codes* book tells the operator what to do if this warning occurs.

Facilities exist to dump spool files to tape when the spool space is full or nearly full. When spool space is again available, the system operator can restore the dumped spool files to the system for processing.

VSAM and Access Method Services Requirements

VSAM and Access Method Services support in CMS requires both DASD space and virtual storage. For information on the DASD space needed by VSAM and Access Method Services, see the *VM/SP Installation Guide.*

VSAM and Access Method Services support adds approximately 2K to the size of the CMS nucleus. In addition, this support uses free storage to run the VSE logical transients, and for buffers and work areas. VSAM issues a GETVIS macro to request free storage.

If CMS/DOS is entered with the VSAM option

```
set dos on (vsam
```

part of the CMS/DOS virtual storage is set aside for VSAM use.

Estimating DASD Storage Requirements for CMS Users

You can let CMS users use two forms of DASD space:

1. You can allocate minidisks directly to the user's virtual machine.
2. You can authorize them to use DASD space within an SFS file pool. This space is known as the user's *file space*. A user can organize the files in his or her file space into a hierarchy of directories. When the file space is created, one

directory is automatically created for the user. This directory is known as the *top directory*.

In many commands, SFS directories are used as rough equivalents to minidisks. For instance, to make CMS aware of either a minidisk or SFS directory, users enter an ACCESS command for the minidisk or directory.

In a VM/SP system, CMS users can have either or both kinds of DASD storage. You might, for example, allow one user to use only an SFS file space, a second user to use only a minidisk, and a third user to use both a file space and a minidisk. When planning for VM/SP, you should decide what kinds of DASD storage your CMS users will have. In making your decisions, consider the following:

- SFS does not allocate DASD space until it is actually used. When a minidisk is allocated to a user, on the other hand, any unused space on the minidisk is wasted.
- SFS lets users organize their files into a hierarchy of directories. Files that reside on minidisks cannot be organized into directories.
- SFS lets users share files concurrently. Any number of readers and one writer can share a file at the same time. Note that although multiple users can share minidisks by using the CP LINK command, the sharing is limited. Multiple users can read from a minidisk, but only one user can safely write to it at the same time. More than one user writing to the same virtual device can result in a permanent loss of data. (CMS does not protect a user from loss of data when multiple users have write access to the minidisk.) Furthermore, unpredictable results can occur when one user has a read-only link to a device that is being updated by a user who has the device in write status.
- SFS eases DASD management. Rather than divide a DASD volume into multiple user minidisks, you can allocate an entire DASD volume to an SFS file pool and let SFS manage that space.
- Remote users can access data in file pools.
- Minidisks provide better performance than SFS file spaces. If users have applications that demand optimal file I/O performance, consider giving those users minidisk space instead of or in addition to an SFS file space.
- The *VM/SP CMS Primer* and the *VM/SP CMS Primer for Line-Oriented Terminals* assume that users have an SFS file space. If you have new users that will be reading those manuals to learn VM/SP, you should plan on giving them file spaces.

For more about the CMS Shared File System, see the *VM/SP CMS User's Guide* and the *VM/SP CMS Shared File System Administration* book.

Estimating Storage Requirements for CMS Minidisks

The following table can help you allocate enough DASD space for CMS minidisks.

Device Type	Approximate 80-byte lists
3310	6.4 per 512-byte block
3330	2300 per cylinder
3340	960 per cylinder
3350	5000 per cylinder
3370	6.4 per 512-byte block
3375	3200 per cylinder
3380	6250 per cylinder
9313	6.4 per 512-byte block
9332	6.4 per 512-byte block
9335	6.4 per 512-byte block

Each physical disk contains file control information as well as your data. Data requires more file control information if put into many small files instead of a few large files.

For an average CMS user, the following minidisk space should be sufficient.

Device Type	Space Required
3310	1700 512-byte blocks
3330	4 cylinders
3340	11 cylinders
3350	2 cylinders
3370	1700 512-byte blocks
3375	3 cylinders
3380	2 cylinders
9313	1700 512-byte blocks
9332	1700 512-byte blocks
9335	1700 512-byte blocks

If the user also has an SFS file space, you can reduce the preceding amounts proportionally.

Estimating Storage Requirements for SFS File Pools

Because SFS allocates DASD space dynamically, the initial amount of DASD space you would need for an average user is somewhat less than the amounts shown in the previous section. SFS does require, however, additional DASD space beyond the space used strictly for user data. This additional DASD space is needed for file pool control data and for file pool log minidisks.

File pool server processing uses the control data to keep track of the objects in a file pool. The control data also maps the DASD space used within a file pool. The log minidisks help protect the integrity of a file pool from system failures or device errors.

The amount of space needed for file pool control data and for log minidisks varies with factors such as the number of files and directories, the rate of change activity, and the size of the file pool.

Even with the extra DASD overhead, however, the initial total DASD space requirement per average user is less than the amounts shown in the previous section. For a rough estimate, reduce the amounts shown by about 25 percent. Remember that this is a rough estimate, because your average user could consume more or less space than the numbers shown in the chart.

After VM/SP is installed and users begin consuming file pool space, you can more accurately project your DASD needs and plan accordingly. SFS file pools are expandable. You can add DASD storage to them as it is needed.

For more information on file pools and determining the allocations needed for them, see the *VM/SP CMS Shared File System Administration* book.

Chapter 4. Planning for Other VM/SP Areas

Contents of Chapter 4

Planning for CMS

What CMS Does

The Conversational Monitor System (CMS) provides conversational facilities for virtual machine users. CMS operates only in a virtual machine, and together with CP, provides a time-sharing system suitable for program development, problem solving, and general time-sharing work.

Storage Requirements

CMS requires virtual storage and auxiliary storage. A minimum of 1MB of virtual storage is recommended for a CMS virtual machine. This virtual storage is distributed as follows:

- CMS buffers, pointers, and control blocks (DMSNUC)
  - 36K.
- Loader tables
  - 16K (for virtual machines with more than 384K of virtual storage).
- User area and CMS free storage
  - 928K.

Auxiliary Storage

The CMS auxiliary storage requirements are distributed as follows:

- System residence for CMS (190 minidisk) –

Device Type	190 Allocation	Nucleus Size	nnnnn (CMS)
3330	173 cyl	18 cyl	155 cyl
3340	429 cyl	45 cyl	384 cyl
3350	80 cyl	10 cyl	70 cyl
3375	120 cyl	12 cyl	108 cyl
3380	78 cyl	8 cyl	70 cyl
FB-512	72,000 blk	8,256 blk	63,744 blk

Notes:

1. The CMS system and the CMS nucleus reside on the 190 minidisk. The CMS system must reside on a minidisk. It cannot reside in a file pool.
2. The 3310, 3330, and 3340 starter systems are no longer supported.

- Resident minidisk space or SFS file pool space for application programs (CMS commands, user programs, IBM licensed programs)—the space needed is program-dependent, and must be assigned by you.
- Work space (either minidisk space or file pool space) for application programs—the space needed is program-dependent, and must be assigned by you.

Device Support

Table 4 shows the IBM virtual machine devices supported by CMS.

Table 4. Devices Supported by a CMS Virtual Machine

Virtual Device Type	Virtual Address[1]	Symbolic Name Default	Device Use
3210, 3215, 1052, 3066, 3270	vdev[2]	CON1	System console
3310, 3330, 3340, 3350, 3370, 3375, 3380, 9313, 9332, 9335	190	DSK8	CMS System disk (read-only)
	191[3]	DSK1	Primary disk (user files)
	vdev	DSK2	Minidisk (user files)
	vdev	DSK3	Minidisk (user files)
	192	DSK4	Minidisk (user files)
	vdev	DSK5	Minidisk (user files)
	vdev	DSK6	Minidisk (user files)
	vdev	DSK7	Minidisk (user files)
	19E	DSK9	Minidisk (user files)
	vdev	DSK0	Minidisk (user files)
	vdev	DSKH	Minidisk (user files)
	vdev	DSKI	Minidisk (user files)
	vdev	DSKJ	Minidisk (user files)
	vdev	DSKK	Minidisk (user files)
	vdev	DSKL	Minidisk (user files)
	vdev	DSKM	Minidisk (user files)
	vdev	DSKN	Minidisk (user files)
	vdev	DSKO	Minidisk (user files)
	vdev	DSKP	Minidisk (user files)
	vdev	DSKQ	Minidisk (user files)
	vdev	DSKR	Minidisk (user files)
	vdev	DSKT	Minidisk (user files)
	vdev	DSKU	Minidisk (user files)
	vdev	DSKV	Minidisk (user files)
	vdev	DSKW	Minidisk (user files)
	vdev	DSKX	Minidisk (user files)

Notes.

[1]The device addresses shown are those that are preassembled into the CMS resident device table. These need only be modified and a new device table made resident to change the addresses.

[2]The virtual address of the system console may be any valid multiplexer address.

[3]The 191 minidisk is automatically accessed as file mode A unless it is dynamically changed by an ACCESS at CMS initial program load (IPL). If the user ID is enrolled in an SFS file pool, the virtual machine can be set up so that the user's top directory is accessed as file mode A instead of a 191 minidisk. In this case, a 191 minidisk is optional. (The top directory is the SFS directory that is automatically created when the user ID is enrolled in an SFS file pool.)

Under CP, unit record devices and the system console may be simulated and mapped to addresses and devices other than the real ones. For instance, CMS expects a

3215, 3210, 1052, or 3270 type of operator's console, but some terminals are 2741s. Regardless of the real device type, the virtual system console is a 3215. The control program (CP) of VM/SP handles all channel program modifications necessary for this simulation. CMS virtual disk addresses are mapped by CP to different real device addresses.

Disks

The read-only CMS system disk (S-disk), usually located at virtual address 190, contains the CMS nucleus functions and disk-resident CMS command modules. The CMS nucleus is loaded into virtual storage when you enter the CP IPL command. CMS remains resident until you enter another IPL command or until you log off. The disk-resident modules are loaded into virtual storage only when their services are needed.

In addition to the system disk (file mode S) and the primary DASD storage, which can be either a minidisk or an SFS directory, each CMS user can access up to 24 additional disks or SFS directories at one time. Files that you wish to retain for later use can be stored on one of your accessed minidisks or SFS directories. Information stored on a minidisk or SFS directory remains there until you or some other authorized user erases it. An exception to this is the temporary minidisk (or temporary disk). Files written on a temporary disk are lost when you log off. See the *VM/SP CMS User's Guide* for more information about CMS disks and SFS directories.

Libraries

CMS supports simulated partitioned data sets that contain:

- CMS and OS macro/copy files to be used at compilation or assembly time (source/macro libraries). The CMS file type for these files is MACLIB.
- Object routines to be referred to at load and/or execution time (text libraries). The CMS file type for these files is TXTLIB.
- Executable routines that are loaded by OS SVCs that CMS simulates. The CMS file type for these files is LOADLIB.
- Executable routines that are fetched by DOS SVCs that CMS simulates. The CMS file type for these files is DOSLIB.

These simulated partitioned data sets can reside on a minidisk or in an SFS directory.

System Macro Libraries

The System Macro libraries, located on the CMS system disk, are:

Library	Contents
DMSSP MACLIB	CMS and DOS macros versioned by VM/SP
CMSLIB MACLIB	CMS macros not versioned by VM/SP
OSMACRO MACLIB	The selected OS macros from SYS1.MACLIB that are supported under CMS
OSMACRO1 MACLIB	The remaining distributed OS macros from SYS1.MACLIB
OSVSAM MACLIB	A subset of OS/VS VSAM macros that are supported under CMS
TSOMAC MACLIB	The OS macros distributed in SYS1.TSOMAC
DOSMACRO MACLIB	Internal macros used by CMS/DOS support routines

If you have previously created a CMS macro library and called it DOSMACRO MACLIB, you should rename it so that it does not conflict with the DOSMACRO MACLIB supplied with the system.

If you plan to assemble VSE programs containing VSE macros in CMS/DOS, you must first create a CMS macro library that contains all the VSE macros you need. The *VM/SP Installation Guide* contains a procedure for copying an entire macro library. The procedure for copying individual macros is described in the *VM/SP CMS User's Guide.*

If you plan to assemble VSE programs containing VSE/VSAM macros, you must first create a CMS MACLIB containing the VSE/VSAM macros. The VSEVSAM EXEC, distributed with VM/SP, can be used in conjunction with the VSE/VSAM optional source distribution tape to create such a library. See the *VM/SP Installation Guide* for additional information on the VSEVSAM EXEC.

System Text Libraries

The System Text libraries, located on the CMS system disk, are:

Library	Contents
CMSLIB TXTLIB	The CMS System Text library
TSOLIB TXTLIB	Selected TSO routines necessary to support certain features of the language licensed programs

Callable Services Library

The Callable Services Library, located on the CMS system disk, is:

Library	Contents
VMLIB CSLSEG	The CMS Callable Services Library, which is placed into a saved segment with the SEGGEN command

Other Libraries

The CMSBAM DOSLIB is also located on the CMS system disk. This library is used to build the CMSBAM saved segment used with CMS/DOS, and the PROPLIB LOADLIB, which supports the Programmable Operator Facility.

Note that execution-time libraries are available with the licensed program language processors.

You can generate your own libraries and add, delete, or list entries in them by the MACLIB and TXTLIB commands. You can also specify which libraries (system and user) to use for program compilation and execution by way of the GLOBAL command. Up to 63 libraries may be specified (subject to system limits, such as command line length). Although CMS library files are similar in function to OS partitioned data sets, OS macros should not be used to update them. Libraries are contained on the CMS system disk.

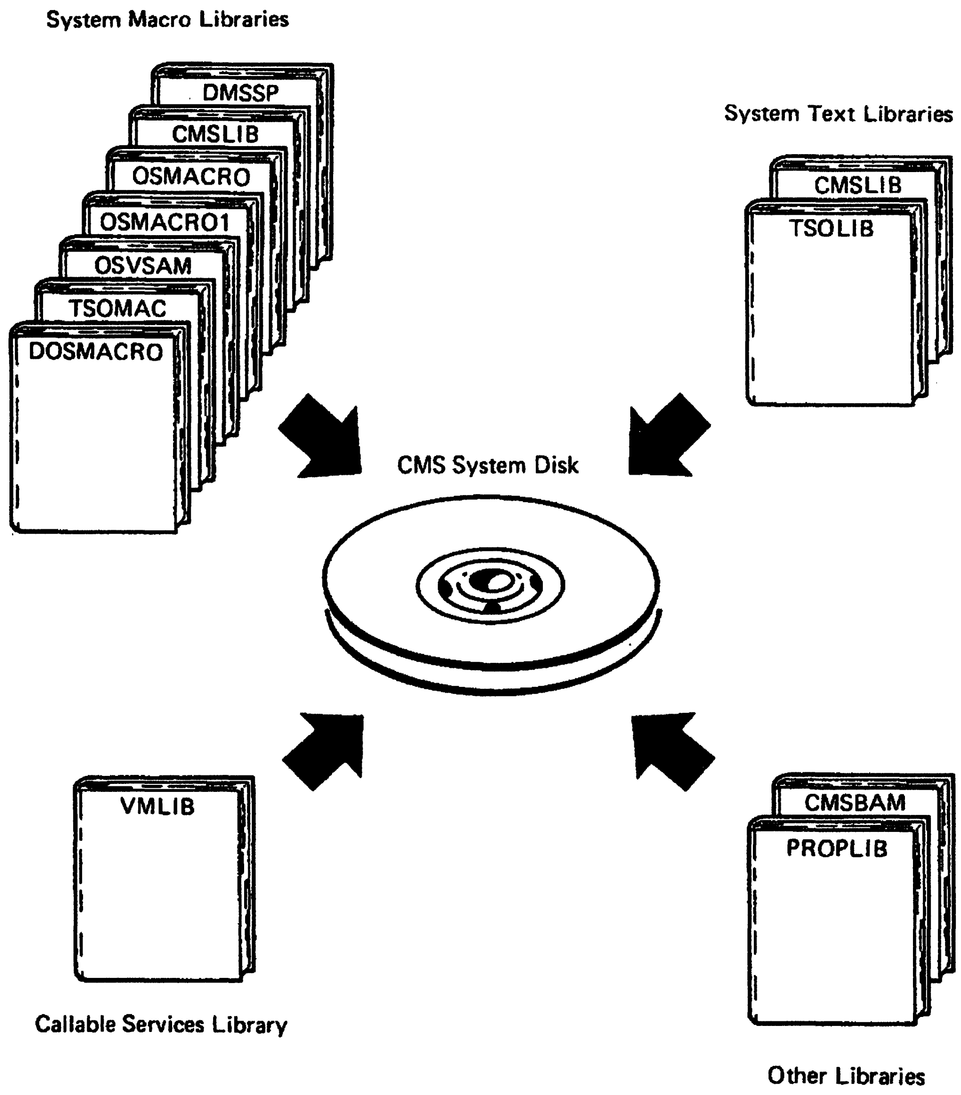


Figure 8. CMS System Disk Libraries

CMS Command Language

The CMS command language lets you converse with CMS. This command language lets you use the following:

- Language compilers
- An assembler
- The CMS file management system
- Context editing and line editing
- Execution control
- Debugging programs
- Generalized HELP facility.

You can also use the CP commands available to all virtual machines under VM/SP directly from CMS. By using these CP commands, you can send messages to the operator or to other users, change your virtual machine's configuration, and use spooling facilities.

To use CMS, you must first gain access to a virtual machine by the CP LOGON command and then IPL CMS. Then, you can enter commands or data from the terminal. Upon completion, each command returns control to you.

Note: If your system was built with your national language, you may be able to enter CMS commands in your own national language, rather than American English.

For information about how to use CMS, see the *VM/SP CMS User's Guide*; for a complete description of all the CMS commands, see the *VM/SP CMS Command Reference*.

Program Language Facilities

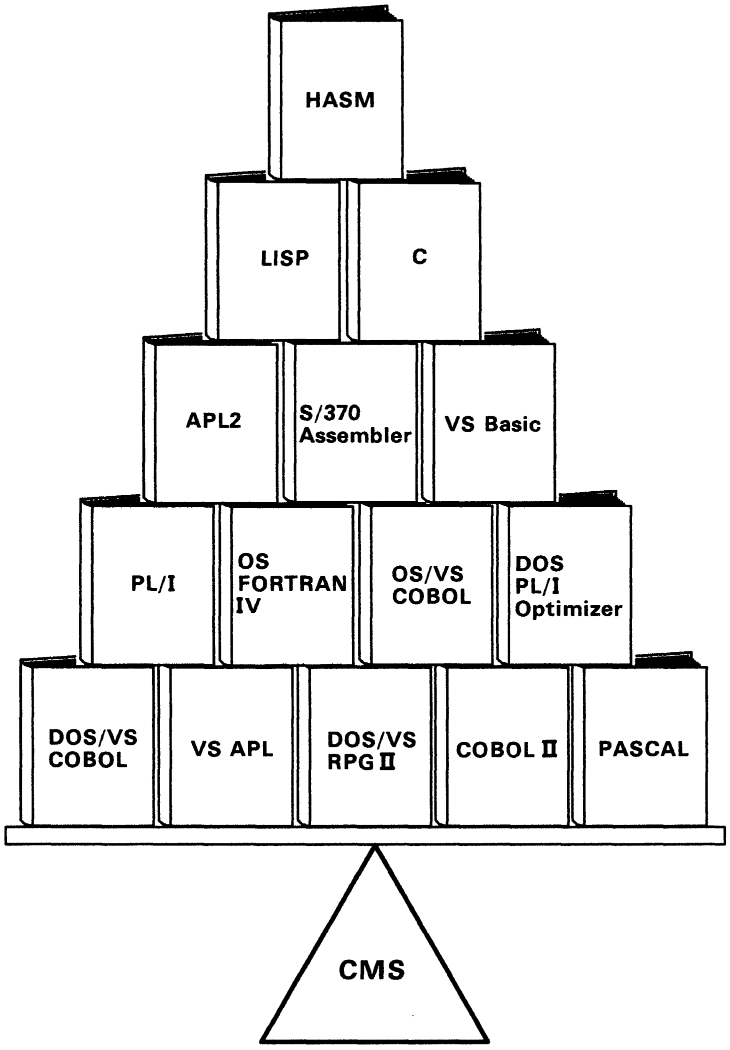


Figure 9. Some Programming Languages Supported by CMS

As Figure 9 shows, at least 15 programming languages can be supported by CMS. Note the assembler is distributed with VM/SP. The language compilers that are licensed programs must be ordered separately.

CMS runs the compilers by way of the interface modules. Users should always recompile their programs and compiler interfaces under the system they are using to ensure any interface changes are incorporated (that is, control block changes). CMS commands are provided to use the compilers within the conversational environment of CMS.

Limited Support of OS and VSE in CMS

Object programs (TEXT files) produced under CMS or OS can be executed under CMS if they do not use certain OS functions not simulated by CMS. Object programs using nonsimulated OS macro functions must be transferred to an appropriate real or virtual OS machine for execution.

Sequential and partitioned data sets residing on OS disks can be read by OS programs running under CMS. Also, certain CMS commands can process data sets on OS disks.

CMS does not support multibuffering for non-DASD devices. There is one DCB per device, not per file.

CMS simulates the control blocks, supervisor and I/O macros, linkage editor and fetch routines necessary to compile, test, and run VSE programs under CMS. The support for the VSE user is comparable to that for the OS user.

CMS supports VSAM and Access Method Services for VSE and OS users. See the *VM/SP CMS Command Reference* for the restrictions on using VSE/VSAM and Access Method Services in CMS. CMS also supports the VSE/VSAM macros and their options and a subset of the OS/VSAM macros.

CMS/DOS support of VSAM is based on the VSE/VSAM program product.

Application programmers who usually use CMS to interactively create, modify, and test programs may require facilities not supported in CMS (for example, an OS program using ISAM). They can alternately run CMS and another operating system in the same virtual machine.

A description of the actual processes for reading OS or VSE files is in the *VM/SP CMS User's Guide*. The *VM Running Guest Operating Systems* book contains a description of alternating operating systems.

DL/I in the CMS/DOS Environment

Batch DL/I application programs can be written and tested in the CMS/DOS environment. This includes all batch application programs written in COBOL, PL/I, RPG II, and Assembler languages.

You can run any data base description generation and program specification block generation. The data base recovery and reorganization utilities must be run in a VSE virtual machine.

See the *VM/SP CMS User's Guide* and the *DL/I DOS/VS General Information* book for more information.

Disk and File Management

| For many CMS commands and application programs to use DASD storage, CMS must be made aware of that storage. To do this, the user must enter an ACCESS command. Up to 26 units of DASD storage can be accessed at the same time. The accessed items can be either SFS directories, minidisks, or full packs. Minidisks or full packs (often referred to as *virtual disks*) can be in the following formats:

CMS	CMS disks are formatted with the CMS FORMAT command. Files contained on these disks are in a format unique to CMS, and cannot be read or written using other operating systems.
OS or VSE	OS or DOS disks or minidisks may be used in CMS. OS or VSE programs running in CMS may read data sets or files on OS or DOS disks, but may not write or update them. OS and DOS minidisks can be formatted with the Device Support Facility, or with an appropriate OS/VS or VSE disk initialization program, if the disk is a full pack.

VSAM	Minidisks used with VSAM must be formatted with the Device Support Facility. Full disks must be initialized using the appropriate OS/VS, Device Support Facility, or VSE disk initialization program.

Unlike minidisks or full packs, SFS file spaces and the directories created within them are not *formatted*. Instead, file spaces are allocated when someone administering a file pool enrolls a user and gives them space in the file pool (by using a CMS ENROLL USER command). Within this file space, one directory (known as the top directory) is automatically created for the user. The user can create a hierarchy of directories beneath this top directory by using the CMS CREATE DIRECTORY command.

Within an SFS directory, users can create and use CMS files, just as they can on CMS formatted minidisks. And, like CMS formatted minidisks, SFS directories can contain partitioned data sets of the forms supported by the CMS simulations of OS and DOS.

Although SFS directories can contain partitioned data sets, they cannot represent an OS, VSE, or VSAM disk. This means, you cannot *format* an SFS directory using the Device Support Facility. Also, you cannot format an SFS directory with an OS/VS or VSE disk initialization program.

While users typically access SFS directories, many CMS commands and program functions do not require the SFS directory to be accessed. Instead, the desired SFS directory is specified in the command or program function. CMS then directly references the chosen directory. There is no limit to the number of SFS directories that CMS can reference at the same time in this manner.

Finally, in addition to the forms of DASD storage discussed earlier, CMS can make use of a Mass Storage System (MSS). When VM/SP MSS support is installed, and the VM/SP processor is attached to an MSS, MSS 3330V storage can be accessed just as minidisks are accessed.

Disk Access

Disks can be accessed so that files can only be read (read-only), or so that files can be read and written (read/write).

Both CP and CMS can control read/write access. If a disk is designated read/write by CP, then CMS determines if the access is read or write. If a disk is designated read-only by CP, then it can only be read by CMS.

To access a disk, you must:

- Identify the disk to CP as part of your virtual machine configuration. This disk is available if it is defined in your VM/SP directory entry, or it can be acquired with the CP LINK or DEFINE commands.
- Identify the disk to CMS by assigning it a file mode letter. You do this using the ACCESS command in CMS.

You may have many virtual disks known to CP in your virtual machine configuration at one time. CMS lets a maximum of 26 disks be accessed, with file mode letters A through Z. File mode S (usually at virtual address 190) is the CMS system disk. File mode A is your primary read/write file mode. It may be either a minidisk, in which case it is usually at virtual address 191, or an SFS directory.

SFS Directory Access

Like disks, SFS directories can be accessed so that files can be read (read-only), or so that files can be read and written (read/write).

Unlike minidisks, however, CP has nothing to do with the mode in which SFS directories are accessed. Only CMS controls whether a particular SFS directory is accessed read or read/write. Directories that the user owns can be accessed read or write depending on the ACCESS command options specified. Directories that the user is authorized to access, but does not own, can only be accessed as read.

It is not necessary to use CP LINK and DEFINE commands to make an SFS directory a part of the virtual machine configuration. An authorized user only needs to enter an ACCESS command.

File Sharing

The CMS Shared File System lets users share files with others. Any number of readers and one writer can use a file at one time. SFS includes an automatic locking mechanism to prohibit multiple writers. The files can be shared within a processor, or, if TSAF is used, with users of other processors. For more information about using SFS to share files, see the *VM/SP CMS User's Guide.*

If, for some reason, SFS is not available to users, they can use CP to share disks and minidisks. In this case, entire minidisks or disks are shared by using CP LINK commands. LINK command operands determine the type of access (multiple users read-only or read/write). CMS does not provide any control for multiple writes to minidisk files. Therefore it is *not recommended* that CMS disks be used with multiple write access. Note that password protection is provided.

Disk File Format

All disks that contain CMS files must be formatted before first use. A disk can be formatted into one of five disk block sizes:

```
512, 800, 1024, 2048, and 4096 bytes
```

To format the disk, use the CMS FORMAT command. The CMS FORMAT command initializes disks in CMS format and writes a label on the disk.

See the *VM/SP CMS Command Reference* for information on formatting the disk.

Count-key-data (CKD) devices use an 800-byte format to provide compatibility with earlier releases. The volume label is written on record 3 of cylinder 0, track 0 for CKD devices, and on block 1 (origin zero) for FB-512 devices. The volume label contents depends on the formatting block size as detailed in Table 5 on page 71.

Table 5. Volume Label Contents for CMS Formatted Disks

Field Description	Byte Displacement Length	Contents by Disk Block Size (800 byte)	Contents by Disk Block Size (512, 1K, 2K, 4K byte)
Label identifier	0,4	C'CMS='	C'CMS1'
Volid	4,6	user label	user label
Version identifier	10,2	X'0000'	X'0000'
Disk block size	12,4	not used, zeros	F'512', F'1024', F'2048', or F'4096'
Disk origin pointer	16,4	not used, zeros	F'4' or F'5'
Number of usable cylinders/blocks	20,4	not used, zeros	F'n'
Maximum number of formatted cylinders/blocks	24,4	not used, zeros	F'n'
Disk size in CMS blocks	28,4	not used, zeros	F'n'
Number of CMS blocks in use	32,4	not used, zeros	F'n'
FST size in bytes	36,4	not used, zeros	F'n'
Number of FSTs per CMS block	40,4	not used, zeros	F'n'
Disk FORMAT date	44,6	not used, zeros	X'yymmddhhmmss'
Reserved for IBM use	50,2	not used, zeros	not used, zeros
Disk offset when reserved	52,4	not used, zeros	F'n'
Allocation Map Block	56,4	not used, zeros	F'n'
Displacement into HBLK data of next hole	60,4	not used, zeros	F'n'
Displacement into user part of Allocation map	64,4	not used, zeros	F'n'
Reserved for IBM use	68,12	not used, zeros	not used, zeros

On CKD devices, each 512, 800, 1K, 2K, or 4K-byte block (called CMS block in the following discussion) represents one physical record of that size on disk. For FB-512 devices, each CMS block consists of the appropriate number of contiguous FB-512 (512-byte) blocks, logically concatenated to form the correct number of data bytes for that CMS block. The 800-byte disk format is not supported for FB-512 devices.

Files placed on CMS disks can have logical records that are fixed or variable length. In either case, the CMS file system places these file records contiguously into fixed length CMS blocks, spanning blocks where necessary. As a file grows or contracts, its space is expanded or reduced as needed.

Files on a CMS disk are identified by a file directory. The file directory is updated when a command is entered that changes the status of the file on the disk.

For a minidisk formatted in 512, 1024, 2048, or 4096-byte CMS blocks, a single CMS file can contain up to approximately $(2^{31}-1)$-132,000 disk blocks of data, grouped into as many as $2^{31}-1$ logical records, all of which must be on the same minidisk. The maximum number of data blocks available in a variable format file on a 512-byte blocksize minidisk is about 15 times less than $2^{31}-1$. This number is the maximum number of data blocks that can be accessed by the CMS file system.

To ensure that the saved copy of the S-STAT or Y-STAT[10] is current, a validity check is done when a saved system is IPLed. This check is done only for S-DISKs and Y-DISKs formatted in 512-, 1024-, 2048-, or 4096-byte CMS blocks. For 800-byte block disks, the saved copy of the S-STAT or Y-STAT is used. The validity checking consists of comparing the date when the saved directory was last updated with the date when the current disk was last updated. If the dates for the S-STAT are different, then the S-STAT is built in user storage. If the dates for the Y-STAT are different, then the Y-disk is accessed using the CMS ACCESS command: ACCESS 19E Y/S * * Y2[11]. This means that even when the S- and Y-disks are accessed in read/write mode and then RELEASED, the message DMSINS100W S-STAT and/or Y-STAT NOT AVAILABLE will result. Table 6 on page 73 compares the disk devices supported by CMS in the case of 800-byte CMS blocks.

[10] The S-STAT and Y-STAT are blocks of storage that contain the file status tables associated with the S-disk and Y-disk respectively.

[11] The DASD address of the Y-DISK will be whatever was specified when CMS was generated. For the standard system this will be 19E.

Table 6. CMS Disk File Statistics for 800-Byte CMS Blocks

Other Restrictions:	
Maximum number of files per minidisk	For other DASDs is - 3400 (3500)
Maximum number of logical records per file	65,535
Maximum number of data bytes per file	12,848,000 bytes or 16,060 CMS blocks
Maximum number of 800-byte CMS blocks per minidisk.	65,535

Minidisk allocated on device type	Number of 800-byte blocks per cylinder	Maximum usable minidisk size in cylinders
3330	266	246
3340 model 35	96	348
3340 model 70	96	682
3350	570	115
3375	360	182
3380	540	121

There are more restrictions for a minidisk formatted in 800-byte physical blocks. A minidisk cannot contain more than 3400 files for the DASDs supported by CMS. A single file can contain up to 12,848,000 bytes of data only, grouped into as many as 65,535 logical records. The number of 800-byte CMS blocks is limited to 65,535 per minidisk. This results in a maximum usable minidisk size in terms of cylinders depending on the DASD type.

Identifying Disk Files

CMS commands can list the identifications of files on CMS and non-CMS formatted disks and minidisks. The LISTFILE and FILELIST commands list the entries in the master file directory for CMS disks. The LISTDS command lists the entries in the VTOC (Volume Table of Contents) for OS and DOS disks, or data spaces on VSAM volumes.

Tape Support

The following table lists the IBM magnetic tape drives supported by CMS.

Virtual Device Type	**Virtual Address**[1]	**Symbolic Name Default**	**Device Use**
2401, 2402, 2403, 2415, 2420, 3410, 3411, 3420, 3430, 3480, 8809, 3422, 9347	180-187 288-28F	TAP0-TAP7 TAP8-TAPF	Tape drives

Note: [1]The device addresses shown are those that are preassembled into the CMS resident device table.

For 7-track and 9-track tape subsystems, the tape handling commands ASSGN, FILEDEF, and TAPE let you specify the modeset of the tape; track and density. For 7-track subsystems only, you can specify the tape recording technique (odd or even parity, converter on or off, and translator on or off).

If you do not specify the modeset for a 7-track tape, CMS issues a modeset indicating 7 track, 800 BPI (bytes per inch), odd parity, converter on, and translate off. If the tape is 9-track, the density is assumed to be the highest density (or whatever BPI the tape drive was last set) for dual density drives; for single density drives, the featured density (800, 1600, 6250 BPI) is assumed. If the tape is 18 track, the density is assumed to be 38K BPI.

For the 18-track 3480 Tape Subsystem, the FILEDEF command lets you specify the track and density option, and the TAPE command lets you specify the track density, and write-mode options. The ASSGN command does not support the 3480 Magnetic Tape Subsystem.

As an alternative to specifying mode in each command that uses the tape (for example, FILEDEF), you can enter a CMS TAPE MODESET command.

For example:

```
tape modeset (181 9track den 6250
```

TAPE MODESET sets the mode for the tape, which stays in effect until the command is reentered. You must do this if one of your programs is to use tapes in other than the default mode. The tape must be at load point for the MODESET to take effect.

Before using labeled tapes in CMS, see the *VM/SP CMS User's Guide*. The CMS tape label processing features described there let you specify tape files with IBM standard or nonstandard labels, or to bypass label processing for nonlabeled tapes.

Note that CMS supports multivolume tape files for OS simulation.

Note: These restrictions only apply when you run CMS. VSE and OS systems running in virtual machines can continue to read and write tapes with standard labels, nonstandard labels, or no labels on single and multireel tape files.

The VM/SP operator must attach tape drives to your CMS virtual machine before any tape operation can take place.

For information about tape handling in the CMS/DOS environment, see "Planning for CMS/DOS" on page 143.

Unit Record Support

The following table lists the IBM unit record devices supported by CMS.

Virtual Device Type	Virtual Address[1]	Symbolic Name Default	Device Use
2540, 2501, 3505	00C	RDR1	Virtual reader
2540, 3525	00D	PCH1	Virtual punch
1403, 1443, 3203, 3211, 3262, 3800, 4245, 3289-4, 6262	00E	PRN1	Line printer

Note: [1]The device addresses shown are those that are preassembled into the CMS resident device table.

Under VM/SP, these devices are spooled. CMS does not support real or dedicated unit record devices, nor does it support a virtual 2520 Card Punch. Table 4 on page 61 lists the devices supported as virtual devices by CMS.

Shared Segments

The default system definition files place the CMS nucleus in high storage, right below 16MB. An installation may choose, however, to change the location of the CMS nucleus.

Many factors influence the location of the shared CMS nucleus. If a user's virtual machine size extends beyond the start location of the CMS nucleus, then the CMS nucleus will exist within the user's virtual storage. This may prevent a user with a small virtual machine size from acquiring a large contiguous GETMAIN area. The CMS nucleus should be generated at an address high enough to satisfy user storage needs.

When the location of a shared segment such as the CMS nucleus is discontiguous from a user's virtual machine, CP includes entries in a user's segment tables for all of the segments between the end of the virtual machine and the start of the shared segment. These segment tables occupy real storage. An installation with real storage constraints may choose to locate the CMS nucleus at a lower address than the default.

The procedure for changing the location of the CMS nucleus is done during the installation process and documented in the *VM/SP Installation Guide*.

Planning for Minidisks

Overview

- A minidisk (virtual disk) is a subdivision of consecutive cylinders on a Count Key Data DASD storage volume and a subdivision of consecutive blocks on a FBA device.
- A minidisk can be defined for read/write access or read-only access
- A minidisk can be defined for a virtual machine on a temporary or permanent basis
- Each minidisk has a virtual disk address
- A CMS minidisk is accessed according to a letter (file mode), A through Z. This letter establishes the order that CMS searches through minidisks when accessing files.

Minidisks are controlled and managed by CP. If a system runs on both a virtual and real machine, system minidisks must start at real cylinder or block zero. For a detailed list of minidisk restrictions, see Appendix B, "VM/SP Restrictions" on page 411.

Defining Minidisks

The VM/SP directory entry for a virtual machine defines each minidisk by an MDISK statement. Any minidisk so defined in the directory is a permanent part of the configured virtual machine. You can access data on the minidisk whenever you log on.

Temporary need for direct access space requires using a pool of reserved disk space, called T-DISK space. You specify T-DISK space in an MDISK statement in the directory. The pool size is chosen when you allocate disk space with the stand-alone Format/Allocate program. Minidisks created from the T-DISK pool must be initialized at logon time and exist in the virtual machine for only that session. When the virtual machine logs off or issues a CP command to release the T-DISK, the space returns to CP.

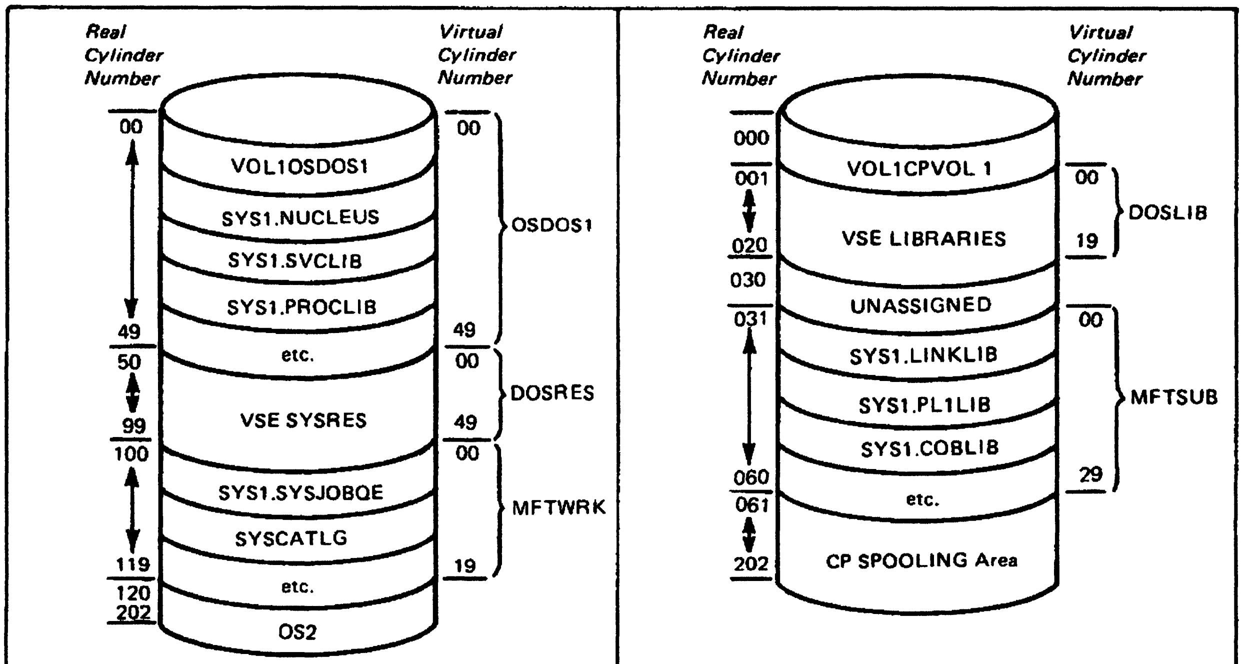
When you allocate minidisks on VM/SP, care must be taken to reduce the following:

- Amount of motion forced on the read/write heads
- Degree of physical overlap for the arm.

To find how to reduce these problems read "Preparing the System Name Table File (DMKSNT)" on page 377.

Note: The DISKMAP EXEC, which IBM provides, checks and flags minidisk extents that may overlap or duplicate. This exec also provides DASD space records for unused or used space. For more information on this exec, see page 226.

Figure 10 on page 77 shows how minidisks are defined and used. The disk labeled OSDOS1 contains many minidisks, some formatted to OS standards and others to VSE standards.



```
VM/SP User Directory Entry for user ABC ( An OS user)
USER       ABC       123      512K
  ACCOUNT 985
  CONSOLE  009   3215
  MDISK    230   2314  000   050   OSDOS1 W
  MDISK    231   2314  100   020   OSDOS1 W
  MDISK    232   2314  031   030   CPVOL 1 W
  SPOOL    00C   2540  READER A
  SPOOL    00D   2540  PUNCH A
  SPOOL    00E   1403  A
```

```
VM/SP User Directory for user XYZ (A VSE user)
USER XYZ PASSWORD
ACCOUNT NUMBER BIN14
 CONSOLE  01F   3215
 SPOOL    C     2540 READER
 SPOOL    D     2540 PUNCH
 SPOOL    E     1403
 MDISK    130   2314  050   050   OSDOS1 W
 MDISK    131   2314  001   020   CPVOL 1 W
```

Note: VM/SP allows cylinders on a 2314 or 2319 normally reserved for alternate track assignment (cylinders 200 to 202) to be optionally used for normal data storage if included within the limits of a minidisk.

Figure 10. Use and Definition of Minidisks

The following information from Figure 10 describes the directory entry for user ID ABC, an OS user.

Virtual Device	Disk Area (Cylinders)	Real Volume
230	50	OSDOS1
231	20	OSDOS1
232	30	CPVOL1

The following information from Figure 10 describes the directory entry for user ID XYZ, a VSE user.

Virtual Device	Disk Area (Cylinders)	Real Volume
130	50	OSDOS1

Virtual Device	Disk Area (Cylinders)	Real Volume
131	30	CPVOL1

Note: On a 3330, 3340, 3350, 3375, or 3380 device, an OS/VS or OS minidisk must start at real cylinder 0 unless VTOC is limited to 1 track. "Minidisk Restrictions" on page 413 provides more detail.

Allocating Minidisk Space

A minidisk always begins at virtual cylinder or block zero. For CKD, its minimum size is 1 cylinder. Except for the 3350 in 3330-1 or 3330-11 compatibility mode or in native mode, a minidisk must exist on its real counterpart. For example, a virtual 3340 minidisk must reside on a real 3340. An FB-512 minidisk can be any number of whole blocks up to the maximum for the volume.

A DASD volume with multiple minidisks has some tracks where the cylinder addresses in the count fields of records R0 and R1 disagree. 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 function rewrites R0 and R1 on each track so that the count fields agree with each other.

VM/SP controls the boundaries of minidisks. For a DASD address outside boundaries listed in MDISK directory statements, CP presents a command reject (seek check) I/O error to the virtual machine.

Note: If the cylinder or block addresses in MDISK statements overlap, data in them may possibly be compromised with no obvious error.

Operating System Minidisks

Operating System (OS) bases its parameters to allocate space on the Volume Table of Contents (VTOC) written on each disk. The amount of space provided for that volume is determined from the format-5 (space accounting) data set control block (DSCB). Thus, for OS to support minidisks, a VTOC must be written whose format-5 DSCB reflects the desired minidisk size. OS treats the remaining disk space on the real disk as permanent dedicated space and not usable as format-5 DSCBs. The Device Support Facility service program should format minidisks for OS or Virtual Storage Extended (VSE).

Virtual Storage Extended Minidisks

Virtual Storage Extended (VSE) allocates space as specified in the EXTENT job control card. You are responsible to see that EXTENT cards refer to valid minidisk cylinders. On a 3330, 3333, 3340, or 3350 minidisk, no cylinder is reserved for alternate tracks within a minidisk. Therefore, a 10-cylinder minidisk must be defined in the EXTENT card as cylinders 0 through 9.

Mass Storage System Minidisks

When Mass Storage System (MSS) minidisks are defined on MSS 3330V volumes, minidisks are virtual 3330-1 disks. The presence of MSS and 3330V system volumes is transparent to a virtual machine accessing minidisks.

Initializing Minidisks

Before data is stored on a minidisk, the minidisk must be formatted to be compatible with the type of records being stored. As with real disks, minidisks are formatted by the appropriate service program. (See the table below.)

Permanent minidisks retain their formatting across sessions; therefore, they need to be formatted only once. However, CP destroys temporary minidisks when the owner logs off, or when the user returns the temporary minidisk to CP (DETACH command). Temporary minidisks must therefore be formatted for each session in which they are used.

For CKD, any program to initialize disks that supports the operating system's use of the DASD type may initialize full disks.

To initialize a minidisk, run one of the following service programs in a virtual machine:

Service Program	**Disk Type**	**Formatting Information**
CMS FORMAT command	CMS (except CMS/VSAM)	Formats specified tracks into 512-, 800-, 1024-, 2048-, or 4096-byte blocks or physical records
CP Format/Allocate program (See Note)	CP	Formats specified tracks into 4096-byte blocks.
Device Support Facility	OS, VSE, and CMS/VSAM (on CKD devices)	Writes read-only track descriptor records for each track and sets the remaining portion of each track to binary zeros. It also writes a format-5 DSCB. The contents of which reflect minidisk size (the amount of free space ready to allocate).

Note: These service programs cause high channel utilization and may cause delay to other users using the channel.

Labeling Minidisks

Any volume containing minidisks must have a label. The label must be on real cylinder 0, track 0, record 3 for CKD devices or on block 1 for FB-512 devices. It identifies the physical volume to VM/SP and must be in the form:

```
VOL1xxxxxx
(Also form for labels created by the
Device Support Facility, IEHDASDR, or INTDK)

- OR -

CMS=xxxxxx
(for disks using an 800-byte format)

- OR -

CMS1=xxxxxx
(for disks using a 512, 1K, 2K, or 4K format)
```

where xxxxxx is a 6-character volume label.

In addition, all virtual machine minidisks should have a label at the same respective, *virtual* locations for CKD or FB-512 devices.

A physical volume holding only virtual machine minidisks can have the first of them starting at real cylinder or block 0. CP recognizes the physical volume if the first minidisk has a valid label.

In Figure 10 on page 77, volume OSDOS1 has its real cylinder 0 assigned to a minidisk formatted for OS. The volume serial number of that minidisk is OSDOS1, the label of the real volume. The minidisk label must correspond to the physical volume. Changing the label affects the directory entries of all users who have minidisks on that volume.

Do not assign real cylinder or block 0 to a user as a data area. If you do, that user has read/write access to the disk and the label can be destroyed.

Also, do not assign user minidisks that begin with real cylinder or block 0 for any physical volumes to contain CP controlled areas (for paging, spooling, and so on). On these volumes, cylinder 0, track 0, and record 4 contain control data required by CP. The VTOC labels for them are compatible with OS, but indicate to OS that no space exists on that DASD. If the space is assigned to a user minidisk, the programs that format OS, VSE, and CMS/VSAM minidisks write over it. The necessary control information is destroyed and CP system failures occur.

Sharing Minidisks

CP assures that access to a minidisk is limited to the minidisk owner, or to other virtual machines given access authorization by the owner. Sharing of minidisks may be initiated by the CP command LINK. A password that authorizes minidisk sharing must be included as an operand on the MDISK directory statement that defines the shared disk. The authorized access may be for read/write or read-only access.

Other virtual machines can link to the minidisk by one of the following:

- LINK control statement in their own VM/SP directory entry

- CP LINK command with the proper password entered during a terminal session.

For example, assume a virtual machine called USERA owns a minidisk at address 150. The VM/SP directory entry for USERA contains:

```
MDISK 150 3380 050 010 SYS003 W READPASS
```

USERA's virtual disk is on the volume labeled SYS003 and occupies real cylinders 050-059. Any other virtual machine that issues the CP LINK command with the proper password can read the 150 minidisk belonging to USERA.

If the following LINK statement is in its VM/SP directory entry, USERA's 150 minidisk can also be read:

```
LINK USERA 150 vdev RR
```

The virtual device address vdev shows where the 150 minidisk belonging to USERA is linked to another virtual machine.

Assume another virtual machine, USERB, with the following statement in its VM/SP directory entry:

```
LINK USERA 150 151 RR
```

USERB can read data from USERA's 150 virtual disk whenever issuing a read for data on its own 151.

A link to any minidisk defined in the VM/SP directory is created if both of the following conditions are met:

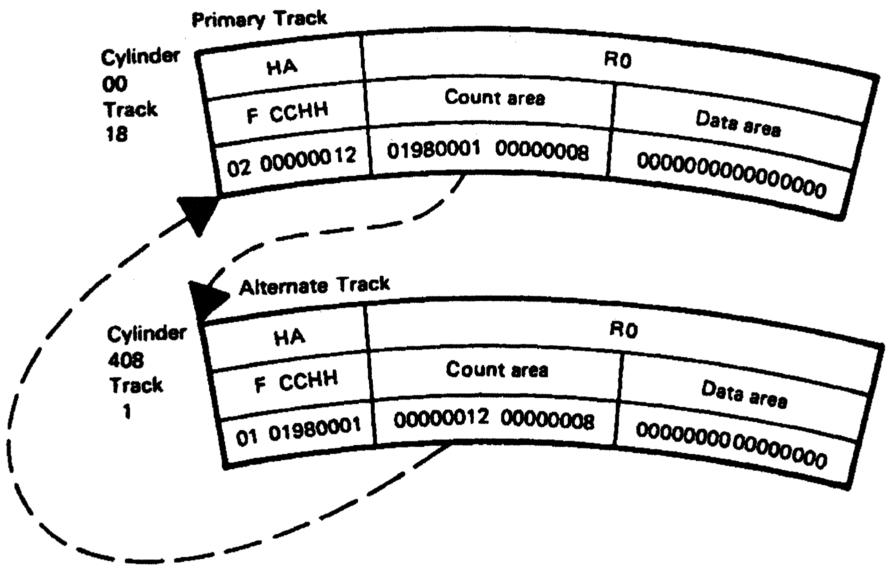
1. The minidisk being linked to has a password specified in the MDISK directory control statement. It must represent the type of link requested.
2. The type of access requested (R, RR, W, and so on) is feasible at the time of the link.

Three primary types of sharing may exist for a minidisk. These correspond to the three passwords that may be entered on the MDISK statement; Read-only, Write, and Multiple.

Note: See the description of the CP LINK command in the *VM/SP CP General User Command Reference* for more information about linking to minidisks.

Specifying Alternate Tracks, Blocks, or Both

As a preventive action when minidisks are made, alternate tracks are assigned as replacements for defective primary tracks. The Device Support Facility can assign these replacements in the field. Figure 11 on page 82 is an example of how primary and alternate track associations are recorded on a cylinder volume.



```
HA     = Home address
R0     = Record 0
F CCHH = Hexadecimal track address
```

Figure 11. Primary and/or Alternate Track Association

3330/3350 DASD

On the 3330 or 3350 DASD devices, the control unit handles the alternate track and/or block assignments. The 3830 control unit assigns primary cylinders as such:

Cylinders	Assigned Area
0 to 403 404 to 411	Minidisks on the 3330 Model 1 or 2 Alternate tracks
0 to 807 808 to 814	Minidisks on the 3330 Model 11 Alternate tracks
0 to 554 555 to 559	Minidisks on the 3350 Alternate tracks

3340/3344 DASD

The 3340 DASD uses a hardware logic that lessens the dependence on alternate track use. The 3340 can bypass the defective portion of a data track and write the balance of the record in the space remaining. When an alternate track is required, it can be assigned by the Device Support Facility. A dedicated 3340 device uses the 3348 Data Module to reserve cylinders as such:

Cylinder	Assigned Area
0 to 347 348	Minidisks on the 3340 Model 35 Alternate tracks
0 to 695 696 and 697	Minidisks on the 3340 Model 70 Alternate tracks

Note: The cylinder values for the 3340 and 3344 DASD are the same.

Once the Device Support Facility has assigned the alternate track, the disk may be used for any common purpose. For example, the disk, including the cylinder with the defective track, could be CP system residence or CMS minidisks. However, three restrictions apply:

- A minidisk should not be located where its track 0 and cylinder 0 fall on a defective track. If it is, the CP IPL command will not function for that minidisk.
- Any operating system doing START I/O (SIO) to this disk must be able to recover common alternate track errors.

  Note: CMS qualifies here because it uses DIAGNOSE in place of SIO.
- Alternate track recovery for overflow record processing is not provided by VM/SP. If a defective track is encountered during overflow record processing, the operation results in a fatal I/O error.

Error Recovery Support: When an I/O issued to a defective 3340 or 3344 track generates a track condition check, an alternate track is needed. Software error recovery procedures provide the means to switch to an alternate track.

- For CP and DIAGNOSE I/O's issued from a virtual machine, switching is automatic and transparent to the issuer of the I/O request.
- For an SIO issued from a virtual machine, the track condition check is sent to the virtual machine. This lets the operating system in the virtual machine run its own error recovery procedures.

Alternate tracks are assigned from the high-order cylinders at the end of the real 3340. Thus, the virtual machine will attempt to seek outside of the minidisk to recover. The VM/SP CCW translation process allows seeks outside of the minidisk to an alternate track. However, the particular alternate track must be assigned to a defective track within that minidisk. After the alternate track is assigned, any attempts at head switching to an unowned track in this cylinder are prevented.

Allocation Conversion: Previously, *alternate track* cylinders of 3340/3344 devices were often used as primary data cylinders. Now these cylinders must be reserved solely for alternate track use. Therefore, changing from an old system (before Release 6 Programming Level Change (PLC) 6) to a current system requires some revisions. You need to revise space allocation and minidisk layouts on any 3340/3344 disk where primary data areas are on *alternate track* cylinders.

System Residence Devices: The system residence device containing *alternate track* cylinders formerly used as the primary data area needs revising. Files of existing control statements should be examined before you generate a new system. The allocate function done on the system residence disk and other CP-owned disks may have to be revised. Following this revision, review how the SYSRES, NAMESYS, and NAMENCP macros are specified.

Minidisk Devices: Any 3340/3344 minidisks extending into *alternate track* cylinders need to be moved. Use the DASD dump restore (DDR) utility to copy them to another area of the disk or to another disk. In the past, when a 3340/3344 had a defective track, the cylinder with the bad track was not usable. The minidisks would be reserved next to that cylinder, but not including it. In such a case, use any version of the DDR utility to dump all cylinders of the real disk to tape. Take care when revising alternate track cylinders (formerly used for primary data area) with the new version of the DDR utility. Make sure that you specify the cylinder range exactly. For example, enter:

```
dump 0 to 697
```

rather than specifying ALL. This dumps only tracks that have been assigned as alternates from the final cylinders. Next, run the Device Support Facility program to assign alternate tracks to defective ones so that all cylinders become usable. Then, use the new DDR utility to restore minidisks from the tape. It may be possible to reorder them into cylinders previously not usable.

Note: If the location or size of a minidisk is changed, you must revise the related MDISK entries in the system directory.

After alternate tracks have been assigned, use only the most current versions of the DDR, DIR, and FMT utilities with 3340/3344 devices.

3375/3380 DASD

Defective tracks encountered on 3375/3380 DASD volumes are handled by the control unit. It provides automatic hardware recovery for handling defective tracks. When a defective track is encountered, the hardware switches to an alternate track. If the end of the alternate track is reached, the hardware switches back to the first track following the defective track.

FB-512 DASD

When FB-512 disks are made, alternate blocks are flagged and replacements assigned for defective primary ones. In the field, they are assigned by the Format Defective Block procedure and used with one of the following:

- Device Support Facility
- LOCATE CCW as described in the hardware manual.

Given the number of the defective block, hardware replaces it with an alternate and then sets up the appropriate pointers.

Reference: *The VM/SP CMS User's Guide*, SC19-6210, discusses minidisks as virtual direct access storage.

The *VM/SP CP General User Command Reference*, SC19-6211, describes the format and usage of CP commands available to the general user.

The *VM/SP CMS Command Reference*, SC19-6209, describes the format and use of CMS commands.

Planning for IPCS

What IPCS Does

The Interactive Problem Control System (IPCS) is an integral part of VM/SP. It provides VM/SP installations with an interactive, online facility for reporting and diagnosing software failures and for managing problem information and status. IPCS uses dump information from CP ABEND, CMS, GCS, PVM, RSCS, and any others created by the CP command VMDUMP, Standalone, or DIAGNOSE code X'94'. By using VMDUMP, the operator makes available to IPCS a dump of user-detected software problems. Figure 12 shows IPCS's approach to handling software problems.

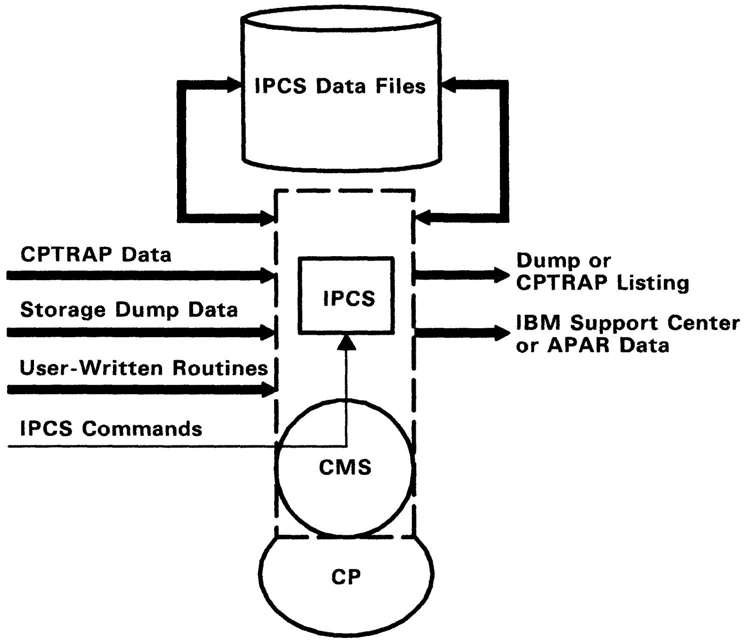


Figure 12. IPCS Handling Software Problems

Machine Requirements

VM/SP IPCS operates on any IBM system that meets the minimum requirements for VM/SP Release 6. It also runs in a CMS virtual machine with at least 2048K of virtual storage and uses any terminal supported by CP as a logon device.

If the output device is a display terminal, the output of IPCSSCAN is written in blocks of twenty-two 80-character lines. The two 80-character lines at the bottom of the display are used as a user input area of 136 characters and a screen status display area of 24 characters. Therefore, for a proper display, the user's terminal must be capable of accepting this format.

Programming Requirements

VM/SP IPCS is used with the VMDUMP facility and CP dumps. It also operates under the CMS component of VM/SP Release 6.

Conversion Considerations

The format of various internal files and data areas as well as internal interfaces for reading and writing dumps, creating problem reports, and more, have been changed for the VM/SP IPCS. Therefore, IPCS cannot view dumps or update symptom summary status for dumps taken by the IPCS supplied with the VM/370 system (Component 5749-DMM), and vice versa.

If your installation is using the VM/370 system IPCS, or the VM/IPCS Extension, and plans to upgrade to VM/SP IPCS, the system IPCS symptom summary file and PRBnnnnn dumps can be converted to the format required by VM/SP IPCS by using the CONVERT command. Any existing system IPCS problem report files need not be converted for use with the PRB, PROB, or APAR commands of the VM/SP IPCS. The format of the problem report files varies from the VM/370 IPCS to the VM/SP IPCS, but the data is accurate.

CONVIPCS EXEC

If your installation has existing PVM Release 2 or RSCS Release 3 IPCSE, these help files must also be converted to the proper format for IPCS usage. The CONVIPCS EXEC will aid you in this conversion.

For details on these conversion utilities, see the *VM/SP Interactive Problem Control System Guide and Reference.*

IPCS can diagnose dumps from systems either with or without the Enhanced Subsystem support.

Storage Requirements

External Storage

The disk storage needed by VM/SP IPCS can be divided into two parts. The current IPCS map, problem report files, and the symptom summary files are contained in the first part. The first part varies in size only slightly as the number of problem reports and symptom summaries vary. For example, for CP dumps, the following amount of data can be stored in 4 cylinders on a 3330:

- 75% (3 cylinders) = 100 problem reports plus symptom summary
- 25% (1 cylinder) = CPIPCS MAP.

The second part of disk storage contains the dumps and supplemental data (for example, console files, trace output). The size of the dump depends mainly on the size of the system being dumped, and the operand of the CP SET DUMP command, either ALL or CP. Table 7 shows approximate space usage by device type for the fixed area and for one dump.

Table 7 (Page 1 of 2). Approximate Space Usage for Fixed Area and One Dump by Device Type

Files	3330 cyl.	3340 cyl.	3350 cyl.
Fixed Area IPCS CPNUC MAP 100 problem reports and symptom summary	4	10	2

Table 7 (Page 2 of 2). Approximate Space Usage for Fixed Area and One Dump by Device Type

One Dump			
512K CP	1.5	4	1
512K ALL	2.5	7	1.5
1024K CP	3	8	1.5
1024K ALL	6	16	3

Calculating DASD Space

Count-Key-Data (CKD) DASD Space: The amount of DASD space on the CMS file system required for a processed virtual machine dump can be calculated from the following formula:

$$C = \frac{S/4 + 5 + R}{P}$$

where:

C = number of cylinders required. If there is a remainder, C must be rounded up.

S = virtual machine storage size in K-bytes.

R = number of pages required for appended load map. R is the number of 4096-byte records occupied by the output of the MAP command of IPCS.

P = number of pages per cylinder.

For:

Device Type	Pages per Cylinder
2305 or 3340	24
3330 or 3333	57
3350 (in native mode)	120
3375	96
3380	150

Note: The 2305 is usually used for paging only.

Table 8 on page 88 shows the number of cylinders needed to contain a dump from various sizes of virtual machines. The figure assumes the entire virtual storage is included in the dump, and a 5-page load map is appended. In the table, K equals 1024 and M equals 1024K (1,048,576).

Table 8. Cylinders Required to Contain a Dump Depending on Virtual Storage Size

Virtual Machine Storage Size	**3330/3333 cyls.req.**	**2305/3340 cyls.req.**	**3350 cyls.req.**	**3375 cyls.req.**	**3380 cyls.req.**
64K	1	2	1	1	1
128K	1	2	1	1	1
256K	2	4	1	1	1
512K	3	6	2	2	1
768K	4	9	2	3	2
1M	5	12	3	3	2
2M	10	22	5	6	4
4M	19	44	9	11	7
8M	37	86	18	22	14
12M	55	129	26	33	21
16M	74	172	36	44	28

FB-512 DASD Space: The amount of FB-512 type DASD space on the CMS file system required for a processed virtual machine dump can be calculated from the following formula:

```
P  =  S/4 + 5 + R
```

where:

P = number of FB-512 DASD pages required. If there is a remainder, C must be rounded up.

S = virtual machine storage size in K-bytes.

R = number of pages required for the appended load map. R is the number of 4096-byte records occupied by the output of the MAP command of IPCS.

Planning for GCS

What GCS Is

The Group Control System (GCS) is another component of VM/SP. It consists of a named, shared segment in storage that you can IPL and run in a virtual machine. GCS provides a different type of execution environment within that virtual machine than CMS provides. While CMS allows only one active operation or task at a time, GCS does *multitasking*. The multitasking services of GCS lets numerous tasks remain active in the virtual machine at one time.

GCS also is a virtual machine supervisor. It bands many virtual machines together in a group and supervises their operations. See Figure 13.

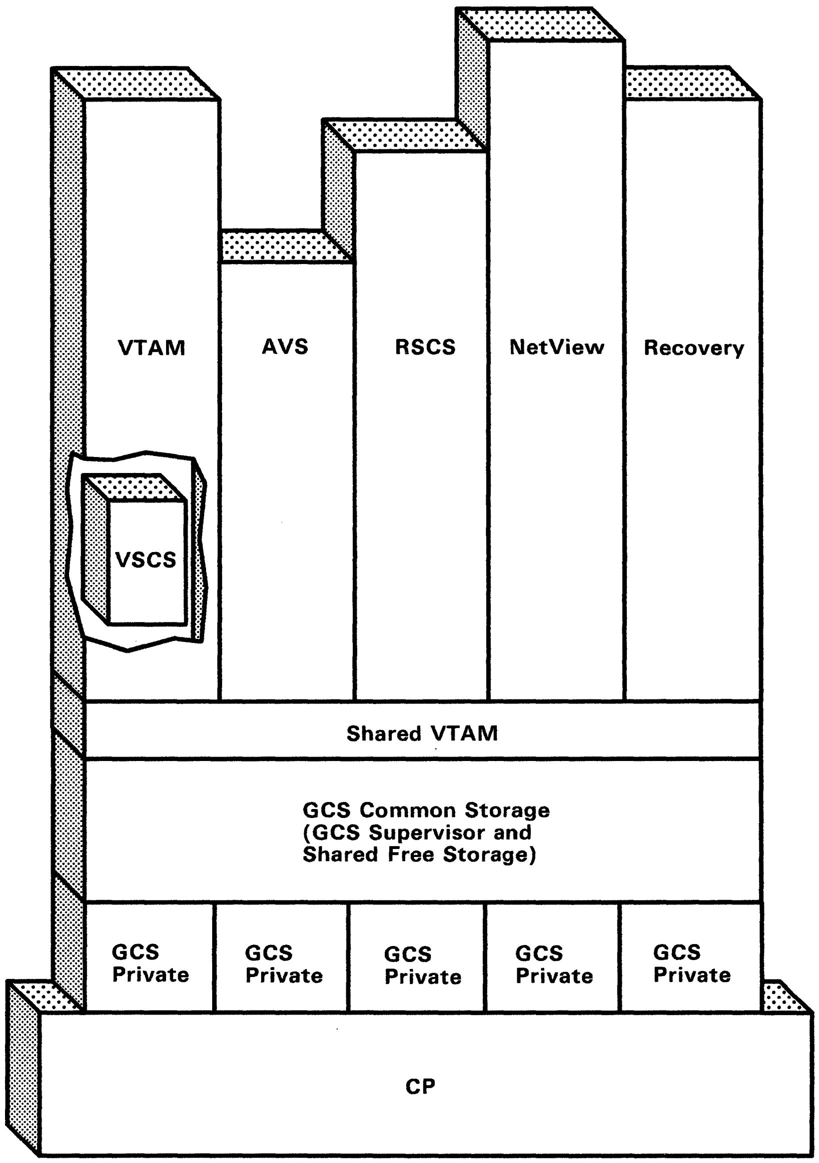


Figure 13. A Virtual Machine Group and Supported Applications

What Applications GCS Supports

GCS supports the following applications:

VTAM (Virtual Telecommunications Access Method)
The specific version of VTAM designed for GCS is ACF/VTAM Version 3 (for VM/SP). ACF/VTAM controls data flow between SNA network devices and programs running in other group machines. Part of ACF/VTAM provides a shared VTAM interface that other program products like RSCS, NCCF, and NetView™ pass information through. (See Figure 13 on page 89.) RSCS uses this shared VTAM interface to communicate with SNA devices; NCCF and NetView perform network management functions through it. For more information, see the *ACF/VTAM General Information (for VM)* book.

VSCS (VTAM SNA Console Support)
This is a VTAM component that lets SNA-connected terminals function as virtual machine consoles. VSCS succeeds the earlier VM/VCNA product, and makes a guest System Control Program (SCP), like VSE or VS1, unnecessary. For more information, see the *ACF/VTAM General Information (for VM)* book.

SSP (Systems Support Program)
With GCS, parts of SSP are VTAM subtasks. SSP performs utility functions for the SNA network's communication control unit. Actually, SSP aids the Network Control Program (NCP), which governs the communication control unit. That control unit, in turn, manages network lines and routing of data. For more information, see the *ACF/VTAM Network Program Products Planning* book.

AVS (APPC/VM VTAM Support)
This is a VTAM application that runs in a GCS virtual machine. It provides the functions necessary for APPC/VM programs within a TSAF collection to communicate with APPC programs anywhere in an SNA network. VTAM provides the LU 6.2 services necessary to communicate with a remote LU. AVS handles the transformation between APPC/VTAM and APPC/VM. AVS can coexist with VSCS in the same system, in the same GCS group, and in the same virtual machine.

RSCS (Remote Spooling Communications Subsystem) Networking Version 2
RSCS Version 2, designed as a GCS application, runs in a group virtual machine and relies on ACF/VTAM to help transfer information via the SNA network. RSCS also can run in a group by itself, spooling files and transmitting messages through non-SNA links. For more information, see the *RSCS Networking Version 2 General Information* book.

NetView™
NetView is an enhanced network management program. It is an optional but recommended VTAM application that facilitates the operation and control of a SNA network. It permits your network operator to control any portion of the network regardless of its physical location. NetView includes the function of the following network management products that are also supported by GCS, plus enhancements in the areas of function, usability, installability, and operability:

- NCCF
- NPDA (Network Problem Determination Application), and
- NLDM (Network Logical Data Manager).

NetView is a trademark of the International Business Machines Corporation.

For more information, see the *ACF/VTAM Network Program Products Planning* book.

How GCS Relates to CMS

GCS, like CMS, is a VM/SP component in Release 6. Although these two components share a few similarities, they have very different functions. For instance, while CMS allows one active operation at a time, GCS has a multitasking capability (See "Multitasking" on page 98.) that lets complex applications have many active tasks at one time.

GCS and CMS do provide some of the same macros and commands to control applications. GCS supports more than seventy OS macros. Over 50 of them have CMS counterparts (though some of the supported parameters differ), while the remaining macros are unique to GCS. CMS supports its OS macros at the OS/360 Release 21.7 level, while GCS supports its OS macros at an MVS/SP 1.3.1 level. See the *VM/SP Group Control System Command and Macro Reference* for a complete list of GCS macros.

Likewise, some GCS commands resemble ones that exist in CMS. The following nine commands share the same or slightly modified formats in both environments:

ACCESS	GLOBAL	QUERY
DLBL	HX	RELEASE
FILEDEF	OSRUN	SET

See the *VM/SP Group Control System Command and Macro Reference* for the actual command formats.

In addition, the VSAM interface supported by GCS is the same as the one used by CMS, with VSAM disks in VSE/VSAM format. In fact, the VSAM macros GCS uses reside in a CMS macro library named OSVSAM MACLIB.

Also, GCS has many of the same REXX capabilities as CMS. The section entitled "Entering Commands to GCS" on page 96 lists the exceptions.

Beyond these similarities and differences, GCS and CMS have another relationship: GCS relies on CMS for its interactive capabilities. For example, you have to complete the GCS build and installation process using CMS. See the *VM/SP Installation Guide* for an explanation of the process. Even after you have created GCS, you still need CMS for:

- Editing, assembling, and link-editing GCS programs
- Initializing disks and creating catalogs (utility functions)
- Creating VTAM's network definition files
- Creating REXX files of file type GCS (VM/SP System Product Interpreter files)
- Building VSAM saved segments
- Examining and printing dumped storage information.

| Files used by GCS **cannot** reside in an SFS file pool. SFS is a part of CMS. GCS
| does not use CMS to do its file I/O.

Virtual Machine Group

VM/SP or VM/SP HPO Release 6 with GCS installed contains a virtual machine group operating environment. A *group* is one or more virtual machines that have IPLed the same GCS shared segment. Members of a group share:

- Common storage space
- A common supervisor
- The ability to *talk* to each other.

GCS governs the group's machines. It is a base that holds the group together and a supervisor that provides a variety of services for each member machine. The type of services available depends upon the *authorization* of individual group members. Unauthorized members run only in problem state and are prevented from using certain GCS services. Authorized members can run in supervisor state and use more GCS services.

Figure 13 on page 89 shows the structure of a virtual machine group. The virtual machine group, with built-in supervisor, supports a VM/SP operating environment for programs, like VTAM, that once needed guest operating systems.

Building the Group

When you define and install GCS, you provide information that builds, or configures, your group. This information goes into a group configuration file that resides in GCS private storage. Some of your input to that file includes:

- A name for the supervisor (actually, your GCS system name)
- User IDs of machines authorized to run in supervisor state
- Addresses of disks you need to access
- A maximum group size
- The user ID of one virtual machine, called a recovery machine, to *clean up* group resources when other machines leave the group
- Names of other shared segments (like VTAM and others).

After you have built a configuration file and installed your GCS segment, the GCS supervisor admits machines that IPL the shared segment, by name, into the group. On a single VM/SP or VM/SP HPO system you can build multiple GCS segments and multiple virtual machine groups.

Joining a Virtual Machine Group

Once you have installed GCS and defined it as a named, saved system, user IDs that have proper links can IPL it and share a group copy of the GCS shared segment. Those user IDs then share access to GCS supervisor code and common storage.

To join a virtual machine group, you log on and IPL the GCS shared segment[12].

Common storage is a read/write area with two parts:

- Common free storage
  Contains free storage space for applications to use.

- Shared GCS code

[12] To leave the group, you simply IPL another system or cause a reset. See "What Makes a Machine Reset?" on page 112 for an explanation of "resets."

Contains the group's shared copy of GCS supervisor code, along with control blocks and data that all members of the group share.

Communicating between Machines in a Group

Machines in a group communicate with each other through:

- IUCV (Inter-User Communications Vehicle)
  - IUCV handles communication between virtual machines within a single VM system or between a virtual machine and a CP service. See the *VM System Facilities for Programming* book for more information on IUCV. In addition, it handles communications between routines ("task-users") within virtual machines. See "Communicating through IUCV" on page 108 for details.
- APPC/VM (Advanced Program-to-Program Communication/VM)
  - is a means of communication between two virtual machines. APPC/VM is mappable to the SNA LU 6.2 APPC interface and is based on VM/SP IUCV functions. With the Transparent Services Access Facility (TSAF) virtual machine component, APPC/VM provides communication services within a single system and throughout a group of virtual machines on different systems the same way that IUCV provides them within a system. See "Communicating through IUCV" on page 108.
- CP Signal System Service
  - Each machine receives a unique *signal ID* when it joins a group. When one machine wants to exchange information with a second group member, it:
    1. Records this information in common storage, and
    2. Notifies the second machine's signal ID of the information waiting in common storage.

Note: If you have many groups, and machines in one group want to communicate with machines in another, they must use IUCV instead of the CP Signal System Service. Although the CP Signal System Service provides unique signal IDs within a group, it reuses the same IDs across different groups.

Authorization

There are three levels of authorization in the GCS environment. With each increasing level of authorization, you receive a greater amount of access to the GCS system. (The first level has the least amount of access. The third level has the most, because it requires authorization at the previous two levels.) You authorize who gets access to each level.

At level:	User IDs have access to:
1	The GCS supervisor and common storage
2	Supervisor State (and privileged GCS functions)
3	Certain restricted CP commands

Controlling Access to the GCS Supervisor

The GCS supervisor is part of the GCS shared segment. Having access to the supervisor results from being able to IPL the segment. So if you prevent certain user IDs from IPLing your GCS system, you cut off their access to the supervisor.

To enter an IPL command and successfully access the GCS supervisor, a user ID needs a link to your GCS system disk. You can establish this required link to your GCS system with:

- A LINK directory statement, or
- A CP LINK command (entered by the user ID's virtual machine operator).

Only the user IDs you provide with LINKs will be able to IPL GCS.

Controlling Access to Supervisor State

Once a user ID has access to the GCS supervisor, it will operate in problem state unless you authorize it to run in supervisor state. You provide access to supervisor state by:

- Authorizing the user ID at build time

  When defining the virtual machine group (see the *VM/SP Installation Guide*) you provide a list of user IDs that will have access to supervisor state and authorized GCS functions. The virtual machine associated with an authorized user ID is called an *authorized machine*. And, any applications that run under these authorized user IDs are considered *authorized* too.

- Authorizing entry points

  You can select a certain entry point, a location in a shared segment, to run in supervisor state. (GCS's AUTHNAME macro lets authorized programs identify these authorized entry points.) A problem state program can pass control to that entry point, which will run in supervisor state. The program later will regain control in problem state.

Table 9 describes how problem state and supervisor state differ:

Table 9. Problem State Versus Supervisor State

Problem State	**Supervisor State**
Both authorized and unauthorized user IDs can run applications in this state.	Only authorized user IDs can run applications in this state.
User IDs in this state cannot use privileged GCS functions or instructions.	User IDs in this state can use privileged GCS functions or instructions.
User IDs can use only storage having a storage protection key of 14.	User IDs can use storage of any key.

Controlling Access to CP Commands

After IPLing GCS, many CP commands (like SPOOL, LINK, and MESSAGE) will work without disrupting or affecting your system's ability to function. But some CP commands will harm your GCS code, and others have limited usefulness.

For example, the ADSTOP command TRACE ALL, TRACE BRANCH, and TRACE INSTRUCTION overlay instructions in a virtual machine. If you set an ADSTOP or begin a TRACE at an address within the GCS common area, it can change the instruction at that address and affect the entire group. Because the PER command does the same job without changing storage, you should substitute it in place of ADSTOP and TRACE. Likewise, the CP commands BEGIN, DISPLAY, DUMP, STORE, and VMDUMP permit you to view or alter common storage. Only certain users who are responsible for maintaining and debugging your system should be able to enter them.

With VM/SP or VM/SP HPO Release 6, you can make potentially harmful CP commands unavailable to your GCS user IDs. You must either alter the lists of commands in two existing privilege classes (A through H) or else define two new privilege classes. Depending upon how you restructure these privilege classes, you may have to change the *cl* parameter specified on each GCS user ID's USER control statement in your directory.

Whether you choose to define new classes or alter existing ones, make sure you end up with two privilege classes that contain the following:

1. CP debugging commands for authorized use only

   This privilege class should include all current Class G commands, except ADSTOP and TRACE. Assign it only to authorized user IDs responsible for maintenance and debugging.

2. General-use CP commands for unauthorized user IDs

   This privilege class should include all current Class G commands, except ADSTOP, TRACE, BEGIN, DISPLAY, DUMP, PER, STORE, and VMDUMP. Assign it to unauthorized user IDs that do not need debugging commands.

   Note: With this class assignment, unauthorized GCS users cannot use the VMDUMP command. However, in case of an error, their virtual machines need a way to dump storage. Instead of VMDUMP, they can use the GDUMP command to dump storage and specify where it will go. See the *VM/SP Group Control System Command and Macro Reference* for the GDUMP command format.

Console and Command Support

Communicating through the Console

Any VM/SP or VM/SP HPO supported terminal can be a GCS console (ASCII and 3270 devices included). GCS virtual machine operators use their consoles to communicate with:

- The GCS supervisor (through GCS commands)
- Applications running in the machine (through application commands defined with the LOADCMD command in the *VM/SP Group Control System Command and Macro Reference.*

If the GCS supervisor or GCS applications want to communicate with a GCS virtual machine operator, they send messages to that operator's console using the WTO (Write To Operator) and WTOR (Write To Operator with Reply) macros:

WTO	Writes a message to the console
WTOR	Writes a message and adds a reply ID so the GCS virtual machine operator can respond. Unlike CMS, GCS lets programs keep running even though you might owe them many replies.

See the WTO and WTOR descriptions in the *VM/SP Group Control System Command and Macro Reference.*

Entering Commands to GCS

You can enter commands three ways:

- Directly from the console
- From a program, using the CMDSI macro
- From a command file (exec).

A command file contains a series of GCS commands and resides on a disk. You invoke it with a console command or with GCS's CMDSI macro or from another command file. PROFILE GCS (if you have one) is a particular type of command file that will execute automatically when you IPL your GCS system.

Besides GCS commands, a GCS command file can contain REXX statements and functions. The VM/SP System Product Interpreter processes these REXX statements and, in fact, the entire GCS command file. Therefore, most REXX capabilities you are familiar with in CMS also apply with GCS. The differences with GCS are:

- REXX programs (execs) have a file type of "GCS."
- Each task has its own program stack. With GCS, the program stack's primary use is for communication between execs. Execs belonging to the same program share data on the program stack. Execs belonging to different programs cannot. Moreover, because GCS console management routines bypass the program stack, you cannot stack commands there for execution.
- GCS has no external event queue (also called *terminal input buffer*). If you enter PULL, and a task's program stack is empty, you receive a message at the console (by way of WTOR) asking for the necessary input.
- ADDRESS GCS replaces ADDRESS CMS. (REXX's default is ADDRESS GCS.) It acts the same as ADDRESS CMS, providing full command resolution, including execution of command files and implied CP commands.

  The ADDRESS COMMAND environment acts much as it does on CMS: it executes host commands, but not command files or implied CP commands.
- You can cancel command files using HX. The commands TS, TE, and HI, which worked with REXX in CMS, have no support on GCS.
- You can invoke REXX programs from assembler language programs with the CMDSI macro. FILEBLK, a parameter on CMDSI, contains the address of the file block. FILEBLK is useful for executing in-storage command files, executing command files with file types other than GCS, and establishing an initial subcommand environment.
- Non-REXX programs can share variables with REXX programs by way of the EXECCOMM macro. GCS's EXECCOMM macro has the same capabilities as CMS's EXECCOMM service.

- GCS supports external function calls if they are written in REXX. It does not support external function libraries, like RXSYSFN, RXLOCFN, AND RXUSERFN.
- GCS supports subcommand environments (ADDRESS nnnn) set up using LOADCMD. However, there is no facility like the *non-SVC fast path* for issuing subcommands.

Note: Execs cannot have the same names as the GCS *immediate commands* ETRACE, ITRACE, HX, QUERY, REPLY, and GDUMP. Immediate commands always execute first; therefore, an EXEC of the same name would never execute. For more REXX information, see the *VM/SP System Product Interpreter Reference*.

Processing GCS Commands

GCS processes commands much the same way as CMS does. Some commands get:

- **Processed immediately**

  If you enter commands with the CMDSI macro or any one of these six *immediate* commands:

ETRACE	ITRACE	HX
QUERY	REPLY	GDUMP

  they do not get stacked, and GCS processes them right away, even if you enter them while another command is being executed.

- **Stacked and wait their turn** (regular procedure)

  All commands you define with LOADCMD and all nonimmediate commands you enter get processed serially. See the *VM/SP Group Control System Command and Macro Reference* for the LOADCMD command format. As soon as the current command finishes executing, GCS processes the next command on the stack. The first command entered is the first command executed.

Commands That GCS Supports: GCS supports commands that let you define, start, terminate, and control an application. Five commands are unique to GCS; nine others are existing or modified CMS commands:

Unique GCS Commands	GCS/CMS Commands
ETRACE	ACCESS
ITRACE	DLBL
LOADCMD	FILEDEF
REPLY	GLOBAL
GDUMP	HX
	OSRUN
	QUERY
	RELEASE
	SET

In addition, you can define your own *application* commands with the LOADCMD command. See the *VM/SP Group Control System Command and Macro Reference* for the LOADCMD command format. For a detailed explanation of each GCS-supported command, see the *VM/SP Group Control System Command and Macro Reference*.

Multitasking

GCS provides multitasking services for multiple active tasks, as opposed to CMS which supports only one active task at a time.

- **What is a task?**

  A task is a single piece of work to be done, and, for the most part, an independent routine. A program running in a GCS machine can spawn a series of tasks, each with a specific job to do. Together, these tasks contribute to the program, letting it accomplish its overall assignment.

- **What is multitasking?**

  A program can have tasks that belong to it, and those tasks can have numerous subtasks. With GCS, a single program can have many tasks active at one time, even though the CPU can process only one task at a time. Multitasking is the act of managing system resources for all those tasks as they *line up* to run.

Adding and Discarding Tasks

A GCS program starts with one initial task. And that initial task can add on additional subtasks using the ATTACH macro. Those subtasks, in turn, can add more subtasks of their own. What results is a task hierarchy like that shown in Figure 14 on page 99. All those tasks belong to one GCS application program. They vie with each other for an opportunity to execute in that application's virtual machine.

Tasks use the following two macros for adding and discarding subtasks:

ATTACH	To add on a subtask
DETACH	To get rid of a subtask.

See ATTACH and DETACH in the *VM/SP Group Control System Command and Macro Reference* for more information.

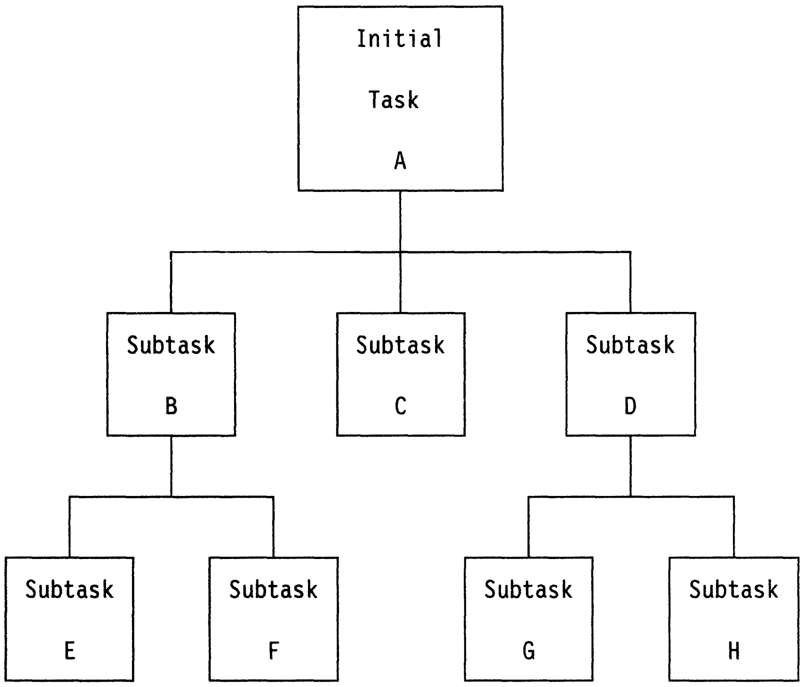


Figure 14. Diagram of a Task's Family Tree. Parent task A adds subtasks B, C, and D. Subtask B becomes the parent of subtasks E and F. Subtask C has no offspring. Subtask D becomes the parent of subtasks G and H.

Dispatching Tasks

To help GCS set up a task hierarchy, each task has a 2-byte task ID and a 1-byte dispatching priority number. Tasks that want to execute first identify themselves with the task ID. And then, GCS sets the order of dispatching according to the 1-byte dispatching priority number.

Tasks themselves determine dispatching priority numbers. Parent tasks assign priority numbers to newly created subtasks. Subtasks' priorities can be the same, higher, or lower than their parents'. To change an existing priority assignment, tasks must invoke the CHAP macro. CHAP works only for:

- A task that wants to change its own priority
- A parent task that wants to change the priority of one of its attached subtasks.

For more information, see the *VM/SP Group Control System Command and Macro Reference* CHAP entry.

Tasks with the largest dispatching priority numbers have the highest priority. Usually, dispatching follows the simple rule:

- High priority before low.

But exceptions do occur:

- When tasks have equal priority, the task dispatcher will keep timing information about the running task. If the running task exceeds the time limits the task dispatcher will switch to a ready task of equal priority.

- When the highest priority task cannot run, GCS dispatches the next-highest, runnable task.

Otherwise, when a task does get dispatched, it maintains control:

- While disabled for interrupts, or
- Until a higher priority task becomes ready to run, or
- Until it terminates, or
- Until it issues a WAIT.

Terminating Tasks

Task termination has two facets:

1. What makes tasks terminate:

   NORMALLY:
   A task ends normally for one reason:

   - It finishes its work and returns control to the GCS supervisor. The supervisor or an exit routine (specified with the GCS TASKEXIT macro) cleans up any resources the task was using.

   ABNORMALLY:
   A task terminates abnormally (abends) because:

   - It requests an abnormal termination with the GCS ABEND macro[13].
   - A parent task above it terminates. (When a parent task terminates, its immediate subtasks and all their attached subtasks terminate too.)
   - Its parent task orders it terminated with a DETACH macro.
   - The virtual machine operator cancels the entire application program.
   - The GCS supervisor cannot provide a requested service.

   The supervisor or an exit routine (specified with the TASKEXIT or ESTAE macro) cleans up any resources the abended task was using.

2. What happens because tasks terminate:

   a. Tasks call exit routines.

      Programs running in authorized machines can set up termination routines with the TASKEXIT macro. These routines reside in shared storage so that they can serve any machine in the group. When any task terminates, normally or abnormally, the GCS supervisor calls these exit routines.

      Not all terminations are final. GCS has procedures that permit tasks to appeal abnormal terminations. Tasks can set up exit routines that are local to their own virtual machine with the ESTAE macro. These routines will clean up resources and decide whether to uphold the abnormal termination. ESTAE lets an exit routine, which you have written:

      - Perform some pretermination processing
      - Diagnose the cause of the abend
      - Continue normal processing at some retry point
      - Continue termination.

      During the exit, an abended task can ask the GCS supervisor to let it recover control and continue executing. GCS will invoke this ESTAE exit

[13] When a task specifies abend with the DUMP option, it receives a dump of its virtual machine.

for any abend, unless certain circumstances prevail. See the "ESTAE" entry in the *VM/SP Group Control System Command and Macro Reference* for more information.

b. GCS *cleans up* resources when tasks terminate:

   - Closing any files the task opened
   - Releasing any storage the task used
   - Releasing any locks the task held
   - Severing all IUCV paths the task established
   - Canceling any timer intervals the task set
   - Canceling resources the task requested by way of ENQ
   - Closing General I/O devices the task opened and unlocking any locked pages of storage
   - Canceling any replies from the operator that the task requested by way of the WTOR macro
   - Subtracting the task's modules from running totals in storage (program load count and use count)
   - *Un-defining* any commands you defined with LOADCMD (only if you terminated the task with an HX command).

Coordinating Dependent Tasks

Often, tasks depend on each other to get work done. For instance, one task might have to stop running until a second task provides additional information or service. When that *event* occurs, and the first task resumes again, the two tasks have synchronized.

Events are important reference points for coordinating or synchronizing tasks. Tasks plan their actions around events by using Event Control Blocks (ECBs). An ECB is a word of storage that represents some event.

The two task management macros that use ECBs are:

- **WAIT** - Suspends the task until some event occurs
- **POST** - Notifies the task that some event has completed.

For example, when a task has to wait for an ECB, it is suspended until a POST macro is issued for that same ECB. A task can wait for a whole list of ECBs. When any one of them gets posted, the task resumes. See Figure 15 on page 102.

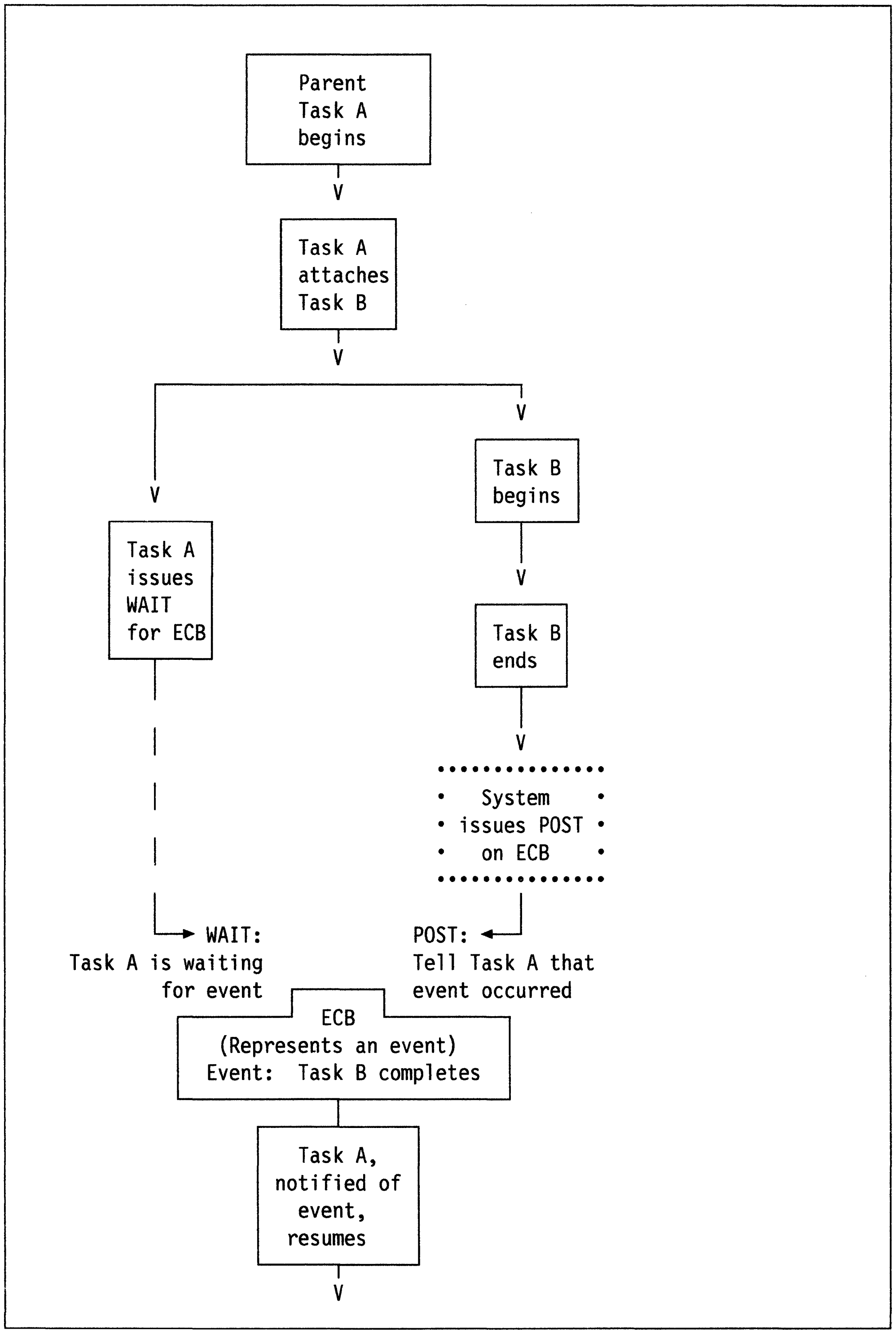

Parent Task A begins
Task A attaches Task B
Task B begins
Task B ends
Task A issues WAIT for ECB
System issues POST on ECB
WAIT: Task A is waiting for event
POST: Tell Task A that event occurred
ECB (Represents an event) Event: Task B completes
Task A, notified of event, resumes

Figure 15. How Tasks Can Use WAIT and POST Macros

WAIT and POST work only among tasks in the same virtual machine. For more information on these macros, see the *VM/SP Group Control System Command and Macro Reference*.

Coordinating Shared Resources

Sometimes tasks have to synchronize their use of a *resource.* A resource is something (perhaps a facility or service) that applications in a particular virtual machine need to use. Its assigned resource name has significance only within that virtual machine, and then only to the applications programmed to use it. When many tasks have to share such a resource, they coordinate their time using:

- **ENQ** - Enqueues a request for control of a resource
- **DEQ** - Releases previously requested resource.

With an ENQ request, a task provides a resource name, identifying the resource it wants to use, and specifies whether it can share that resource. If a task cannot share the resource, it enqueues in *exclusive mode*, requesting exclusive use of that resource. If it can share, it enqueues in *shared mode.* Sometimes tasks have to wait so they each can take separate turns using a particular resource. In other cases, many tasks share one resource at the same time.

If a task has enqueued a resource in exclusive mode, any other task that issues ENQ on that same resource must wait until the first task finishes. After the first task issues DEQ, the second can take its turn. In addition, if one or more tasks are already enqueued in shared mode, a new task cannot gain control in exclusive mode. It will be forced to wait until the others finish with the resource in shared mode.

ENQ and DEQ apply only to tasks running in the same virtual machine. For more information on ENQ and DEQ, see the *VM/SP Group Control System Command and Macro Reference.*

OS Management Services

The OS *management* services (storage management, program management, and timer management) described in this section are GCS services that resemble (but do not duplicate) MVS functions.

Storage Management

Each GCS machine in a virtual machine group has two storage areas: private and common. Private storage is local to an individual machine and not shared with other group members. This means that a program running in a neighboring machine cannot use or change another's private storage area. Common storage, however, is shared in a read/write fashion with all other machines in the group. Any program can use or look at nonfetch-protected information in common storage. But only authorized programs can obtain or otherwise modify storage space there.

GCS uses storage keys to prevent unauthorized storage allocation. Any program that wants to obtain storage must have a PSW key[14] that matches the storage key of the address range in question. Unauthorized programs, for example, have PSW keys of 14. Therefore, they cannot obtain GCS common storage that has a storage key of 0 (zero).

Obtaining Storage: A program or task that runs in a GCS virtual machine can obtain or release storage space as the need arises. It does this using GCS's GETMAIN and FREEMAIN macros. With GETMAIN, the task requests a certain-sized block of storage. GCS allocates the space and passes the block's

[14] Bits 8 through 11 in the PSW.

address along to the task. Later, when the task no longer needs that space, it issues the FREEMAIN macro and tells what block it wants freed.

When a task requests a certain size of storage with GETMAIN, it also can request other storage characteristics by specifying a subpool. A subpool is a number between 0 and 255. This number characterizes storage as:

- Private or common
- Fetch-protected or nonfetch-protected
- Task-related (automatically released when the task ends) or persistent (retained after the task ends).

Assigning Storage Keys: When allocating storage, the GCS supervisor assigns the address range a storage key that matches the requesting task's PSW key. There are sixteen possible storage keys for different types of code. A storage area's key depends upon what type of code it contains:

Key	Type of Code
0	Saved segments and reentrant code (including GCS common storage and other shared code)
1-13	Authorized (privileged) applications
14	Unauthorized (nonprivileged) applications (also the starting key for authorized applications)
15	VSAM and BAM shared segments

Switching Keys: A program can obtain storage only in the key of the PSW that it is running in. Authorized and unauthorized GCS programs both start out with the same PSW Key 14. Thus, unauthorized programs can secure only fetch-protected storage in Key 14. Authorized programs, on the other hand, can allocate storage in any key, including both fetch-protected and nonfetch-protected common storage.

An authorized program, running in supervisor state, can obtain storage in a new key by changing its PSW key. This involves:

- Specifying a new PSW key with the SPKA instruction
- Allocating storage in the new key with the GETMAIN macro.

Program Management

Programs running on GCS can load and execute modules of code that were assembled and link-edited under CMS. Some of these modules reside on a disk in a load library. Others reside in saved segments that get linked automatically when you IPL your GCS segment.

When a GCS program requests a module, the GCS supervisor first tries to find one that was previously loaded in that program's virtual machine. If no usable copy is available, the supervisor tries to locate the module in one of your system's saved segments. (In either case, the supervisor will use a copy where it locates one.) Failing to find it in a saved segment[15], the supervisor searches the load libraries specified by GCS's GLOBAL LOADLIB command. If the supervisor finds the

[15] Each saved segment has a directory created with the CONTENTS macro. The GCS supervisor searches these directories when looking for a particular module.

module there, it loads a copy into the program's private storage area. See Figure 16.

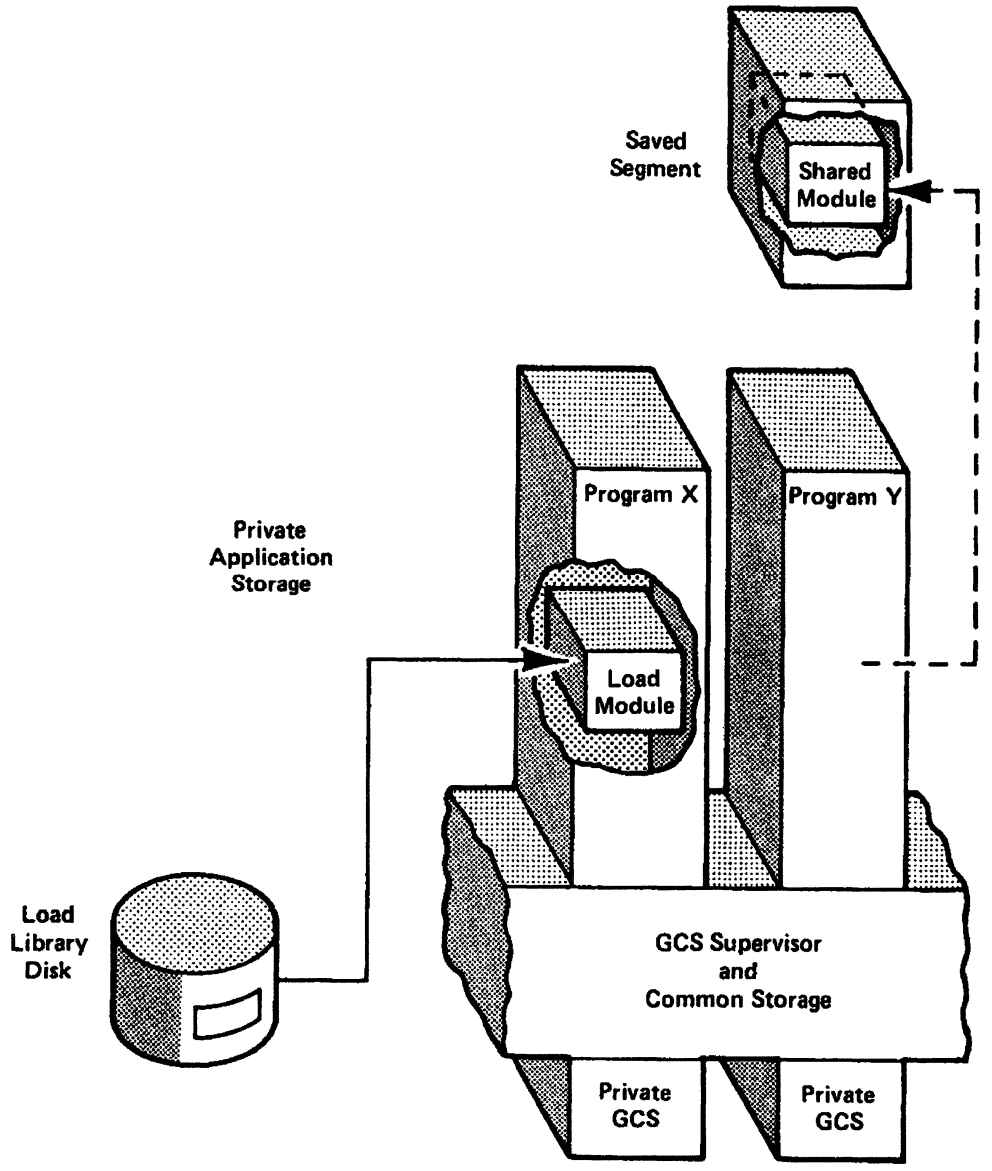


Figure 16. Obtaining Modules Requested by a GCS Program. Program X, on the left, loads a copy of a module from a disk load library. Program Y, on the right, shares a reentrant module in a saved segment, using it where it exists without actually copying it.

To load a module, a program can issue any of the following macros in Table 10 on page 106.

Table 10. Loading Functions

Macro	Action 1	Action 2	Action on Return
LINK	Finds and loads a module (if it was not already in storage) containing a specified entry point.	Passes control to the loaded module at the specified entry point.	After the LINKed module runs, control returns to the program that issued LINK. And if no other program is using that copy of the module, GCS deletes it from storage.
LOAD	Locates and loads a module (if it was not already in storage).	Returns the address of an entry point, associated with the loaded module, to the program that issued LOAD.	LOAD returns control to the program that issued it. The supervisor keeps track of the module's whereabouts until the program issues DELETE.
XCTL	Finds and loads a module (if it was not already in storage) containing a specified entry point.	Passes control to the loaded module at a specified entry point.	After the XCTLed module runs, control does not return to the program that issued XCTL, but to the program before that.

Macros associated with these loading functions include:

BLDL	Creates a directory entry list that contains information about modules you expect to invoke. (It includes their names, what load libraries they reside in, their disk addresses, and other facts).
CALL	Passes control to an entry point in the same or different control section (csect).
DELETE	Releases a module from its caller's control (and removes it from storage if no other programs want to use it).
IDENTIFY	Defines an entry point within a load module.
RETURN	Returns control to the calling program.
SAVE	Saves the contents of registers belonging to a program that is calling another program.
SYNCH	Passes control to a program, in the same or different state, at a specified entry point.

For more detailed descriptions of each macro, see the *VM/SP Group Control System Command and Macro Reference.*

Here are examples of how you might use the loading macros:

LOAD

1. Program 1 LOADs module A.
2. Program 1 gives control to module A with LINK or SYNCH.
3. Module A executes.

4. Program 1 regains control when module A finishes.
5. Program 1 DELETEs module A.

LINK and XCTL

1. Program 2 LINKs to module B.
2. Module B executes and XCTLs to module C.
3. Module C executes.
4. Program 2 regains control when module C finishes.

Timer Management

Programs or tasks that run under GCS sometimes need the services of a timer. A task, for example, may want to set a timer for a certain interval and, when that interval is up, transfer control to an exit routine. Another task might want to set a timer for a certain interval and then stop executing until that interval expires.

GCS has three macros that let tasks define and manage time limits:

STIMER Lets you set a time interval by specifying:

a time length	For the next 10 seconds, do this ...
a time-of-day	At 09:30, do this ...

TIME Asks the GCS supervisor to provide the current time-of-day and date. In effect, it asks the system, *What time is it right now?*

TTIMER Cancels any remaining interval (and exit routine) that was set with the STIMER macro.

For more information and a detailed explanation of each macro, see the STIMER, TTIMER, and TIME entries in the *VM/SP Group Control System Command and Macro Reference.*

Native GCS Services

The native GCS services described in this section make use of unique GCS functions. *Authorization* provides the basis for service. Some functions serve unauthorized programs running in problem state machines; other functions serve only authorized programs running in supervisor state machines.

Calling Authorized Programs

An unauthorized GCS program in problem state can transfer control to an authorized program in supervisor state.

When called, the authorized program executes, beginning at an identified entry point in shared storage. Upon finishing, it returns control to the unauthorized program.

This transfer of control involves two macros:

AUTHNAME The authorized program has to provide an authorized entry point, identified with the AUTHNAME macro.

AUTHCALL The unauthorized program *calls* and passes control to the authorized one by issuing the AUTHCALL macro.

Table 11 on page 108 describes the AUTHCALL macro in more detail.

Table 11. The AUTHCALL Macro

AUTHCALL does	AUTHCALL does not
Cause an authorized program to start executing at an entry point identified with AUTHNAME. The entry point always receives control in supervisor state and Key 0.	Cause a task switch to occur. (The same task is still running.)
Return control to the calling program in its original state and key, when the authorized program finishes.	Let an unauthorized program execute its own code in supervisor state or Key 0.

Communicating through IUCV

GCS supports communication within a virtual machine, or between any two virtual machines, at a routine-to-routine level. Task-users (routines running within a task) communicate through IUCV with:

- Other task-users in the same machine,
- Task-users in other virtual machines on the same system, or
- CP.

GCS also supports communication between virtual machines within a group of VM systems, or cluster. Each of these VM systems must have the Transparent Services Access Facility (TSAF) virtual machine component installed and running. For more information on installing TSAF see the *VM/SP Installation Guide.* For more information on running TSAF see the *VM/SP Connectivity Planning, Administration, and Operation* book. Task-users (routines running within a task) communicate through APPC/VM with resource task-users in:

- Same machine, or
- Other virtual machines in the same or different systems.

The target of an APPC/VM connection must be established as a resource. See the *VM/SP Connectivity Programming Guide and Reference* for details.

Task-users rely on two macros for IUCV and APPC/VM communications:

IUCVINI	Initializes or terminates a task-user's IUCV environment
IUCVCOM	Sets up, carries out, and terminates communications between two IUCV users.

See the *VM/SP Group Control System Command and Macro Reference* for a complete description of these macros.

To allow IUCV and APPC/VM communication at the task-user level, GCS provides:

1. A *nonprivileged* IUCV interface for both authorized and unauthorized task-users. This nonprivileged interface provides the following support:

Functions provided:	**Functions not provided:**
ACCEPT	DCLBFR (Declare Buffer)
CONNECT	RTRVBFR (Retrieve Buffer)
PURGE (IUCV only)	DESCRIBE (Describe)
QUERY	SETMASK (Set Mask)
QUIESCE (IUCV only)	SETCMASK (Set Control Mask)
RECEIVE	TESTCMPL (Test Completion)
REJECT (IUCV only)	TESTMSG (Test Message)
REPLY (IUCV only)	
RESUME (IUCV only)	
SEND	
SEVER	

   Note: The SEND function issues all of the APPC/VM "SEND" functions:

   - SENDCNF,
   - SENDCNFD,
   - SENDDATA,
   - SENDERR, or
   - SENDREQ.

2. A *privileged* interface only for authorized task-users that specify PRIV=YES with the IUCVINI SET function. With the privileged interface, a task-user:

   - Cannot issue IUCVINI REP to change its general exit
   - Cannot issue IUCVCOM REP to change a path-specific exit
   - Must use the IUCVCOM functions CONNECT, ACCEPT, and SEVER to establish or terminate IUCV and APPC/VM paths
   - Can issue the following functions directly (without going through the IUCVCOM macro):

IUCV PURGE	IUCV REJECT
IUCV QUERY	IUCV REPLY
IUCV QUIESCE	IUCV RESUME
IUCV RECEIVE	IUCV SEND
APPCVM QUERY	APPCVM RECEIVE
APPCVM SENDCNF	APPCVM SENDCNFD
APPCVM SENDDATA	APPCVM SENDERR
APPCVM SENDREQ	

Performing I/O

When a GCS program needs an I/O operation performed, it uses a function called General I/O. The related macro, GENIO, provides six different functions that an unauthorized application can use to execute virtual channel programs on any real or virtual I/O device except DASD:

- Open Device (OPEN)

This function identifies a task as owner of a particular I/O device. OPEN also requires the task to specify an exit. Whenever the task receives an I/O interrupt from the device, this specified exit gets control.

- Close Device (CLOSE)

  This function ends a task's *ownership* of a specified device. Once closed, the device stops passing I/O interrupts to the specified exit.

- Modify (MODIFY)

  This function requests that an active channel program be modified. An application first must modify the virtual channel program and then issue MODIFY.

- Obtain Device Characteristics (CHAR)

  This parameter returns information about an I/O device's type, class, model, and features.

- Start I/O (START)

  This function starts a virtual channel program on an open device. (The device may be either virtual or real.)

- Halt I/O (HALT)

  This halts an operation on a given device, terminating any active I/O.

The GENIO macro also provides a function for authorized programs that want to execute real channel programs on real devices:

- Start real I/O (STARTR)

  This starts a real channel program on an open real device. (The device must be real.)

Executing Real Channel I/O Programs: Authorized GCS programs can use real channel programs to move data between main storage and real I/O devices (except DASDs). Real channel programs execute directly on the real channel, without CP first translating them. Before you can execute real channel programs, you need an authorized user ID and a special entry in your VM/SP directory. You make this entry by specifying the DIAG98 parameter on the OPTION directory control statement.

To execute real I/O, authorized programs use GENIO's STARTR (start real) function. For the most part, STARTR works much like the ordinary START function for virtual I/O. However, with STARTR:

- CP does not translate the channel program before starting it.
- GCS issues a DIAGNOSE code X'98' instead of an SIOF instruction.

For more detailed information on GENIO and its parameters, see the *VM/SP Group Control System Command and Macro Reference*.

Securing Pages of Storage

An authorized program intending to perform real I/O using STARTR must first build a channel program[16] in real storage. In the process of building a real channel

[16] For information about building channel programs, see the chapter titled *Input/Output Operations* in the *System/370 Principles of Operation*.

program, the program must lock pages of virtual storage into real storage. Later, it needs a way to unlock those pages.

The two macros that do this are:

PGLOCK	Locks given pages of virtual storage into real storage
PGULOCK	Unlocks pages that were fixed by way of the PGLOCK macro.

See the *VM/SP Group Control System Command and Macro Reference* for more information on PGLOCK and PGULOCK.

Manipulating Locks

Locks are controls that help authorized programs share resources. They serve as warning signs that a particular resource is *in use*. There are two kinds of locks:

Local	Helps synchronize the use of resources within a virtual machine
Common	Helps synchronize common storage among many virtual machines.

The GCS supervisor uses the LOCKWD macro to manipulate these locks and thereby regulate access to local resources or common storage. The LOCKWD macro has parameters that:

- Identify a lock as local or common
- Test the common lock (to see whether it is *on* or *off*)
- Specify whether the lock is to be acquired or released.

When a program or task wants to use a resource within its own virtual machine, it uses the LOCKWD macro to acquire the local lock for that machine. That action prevents all other tasks in the virtual machine from running until the lock is released.

When a task wants exclusive use of common storage, it can acquire the common lock for its virtual machine. First, a task has to acquire the local lock before it tries to acquire the common lock. Next, the program should use the LOCKWD macro to test the common lock's availability. If another virtual machine already has acquired it, the lock will be *on*. Until that machine releases the lock, no other machine will be able to acquire it. In the meantime, if a program tries to acquire the common lock when it is already *on*, the GCS supervisor will suspend the requesting program until the lock gets turned off. As soon as it is off, LOCKWD informs the waiting machine that the common lock is available. This serializes (or synchronizes) group use of common storage.

See the *VM/SP Group Control System Command and Macro Reference* for a detailed explanation of the LOCKWD macro.

Validating Requests for Storage Access

An authorized program can validate another program's request for storage access. The authorized program uses the VALIDATE macro to check input (a parameter list, for example) from the other program. VALIDATE compares the other program's PSW key with the storage key of the storage area to be accessed. If those two keys match, the authorized program will honor the storage access request[17], for both read and write access. If the keys are different and the storage is nonfetch protected, the authorized program will allow read access only.

[17] The authorized program's key does not need to match that of either the unauthorized program or storage. As an authorized program, it can switch itself to Key 0 and transfer data across the different key boundaries.

See the VALIDATE entry in the *VM/SP Group Control System Command and Macro Reference.*

Scheduling Exits in Other Tasks

An authorized program can schedule an exit for any task in any group machine. With the SCHEDEX macro, the program can preempt a specific task and arrange for a designated exit routine to assume control. Instead of the task getting dispatched (providing that it is not disabled), the exit routine gets control in supervisor state and with a PSW key of 0 (zero).

After scheduling the exit, the authorized program continues executing. And after the exit routine finishes, GCS lets the interrupted task continue executing. See SCHEDEX in the *VM/SP Group Control System Command and Macro Reference.*

Establishing Exits for Group Members

Authorized programs can establish exits for the entire virtual machine group. These exit routines must reside in storage that all machines in the group can share.

- Machine exits

  Authorized programs can use the MACHEXIT macro to set up exit routines that will get control when any machine terminates or leaves the group. These routines will execute in the group's recovery machine.

  Note: The recovery machine must be the first one to join your group. It has responsibility for cleaning up system resources when other machines using them reset. (See "What Makes a Machine Reset?") This clean-up involves executing all currently existing exit routines set up with the MACHEXIT macro.

  If the recovery machine itself gets reset, the machines remaining in the virtual machine group will issue a CP SYSTEM RESET, which causes the entire group to reset.

- Task exit routines

  Authorized programs define task exit routines for programs in the same virtual machine group. Whenever a task in one of the group's virtual machines terminates, a specified exit routine gains control. An authorized program uses the TASKEXIT macro to identify the address where that exit routine begins.

- *Exits* to authorized entry points

  Defining an entry point does not define an *exit*, in the true sense of the word. However, when an authorized program identifies an entry point with the AUTHNAME macro (see "Calling Authorized Programs" on page 107), it resembles the act of identifying an exit routine's address. For details on transferring control to authorized entry points, see the AUTHNAME and AUTHCALL entries in the *VM/SP Group Control System Command and Macro Reference.*

What Makes a Machine Reset?

- Logging off
- IPLing another system (or re-IPLing GCS)
- A machine check (under certain conditions)
- Entering certain CP commands:

  SYSTEM RESET
  SYSTEM CLEAR
  DEFINE STORAGE

DEFINE CHANNELS
SET ECMODE OFF
SET ECMODE ON

Data Management Services

GCS applications can process CMS files that reside on minidisks, VSAM files, and CP spool files. GCS applications cannot process CMS files that reside in a Shared File System (SFS) file pool. With GCS's data management services, applications can perform input, output, or update operations on a file, depending on whether it is a CMS, VSAM, or CP spool file. There are two types of data management services:

1. One type (resembling, but not duplicating, MVS/BSAM and MVS/QSAM services) that allows processing of CMS disk files and CP spool files
2. Another type (resembling, but not duplicating, MVS/VSAM services) that allows processing of VSAM files.

Processing CMS Minidisk Files

A GCS program processes CMS minidisk files using BSAM or QSAM macros.

A GCS program processes CMS files using BSAM or QSAM macros. For GCS, these macros have unique constraints. In particular, GCS's data management service supports only the *extended file system* format.

GCS's QSAM/BSAM data management service supports the following command:

FILEDEF	Defines CMS minidisk files and CP spool files.

GCS data management supports the following set of macros, at the MVS/SP 1.3.1 level:

CHECK	Wait for and test completion of a read or write operation (BSAM).
CLOSE	Logically disconnect a file (BSAM and QSAM).
DCB	Construct a data control block (BSAM and QSAM).
DCBD	Provide symbolic reference to data control blocks (BSAM and QSAM).
GET	Obtain next logical record (QSAM).
NOTE	Determine relative position (BSAM).
OPEN	Logically connect a file (BSAM and QSAM).
POINT	Point to the relative position of a specific block (BSAM).
PUT	Write next logical record (QSAM).
READ	Read a block (BSAM).
SYNADAF	Perform SYNAD analysis function (BSAM and QSAM).
SYNADRLS	Release SYNADAF buffer and save areas (BSAM and QSAM).
WRITE	Write a block (BSAM).

Unlike CMS's data management service, it does not let you use:

- Files that reside in a CMS Shared File System file pool,
- OS formatted files
- 800-byte block size disk format.

Nor does it allow:

- File mode 4 (treated instead like file mode 1)
- Spanned records
- Console or tape I/O
- Utility functions (like formatting disks, loading files from tape, editing files, and others).

However, GCS's data management does follow the same rules as CMS's when two or more virtual machines want to share the same disk. Read/write privileges go to only one virtual machine at a time, while multiple disk and minidisk users must share in read-only mode. For detailed information about disk sharing, see the *VM/SP CMS User's Guide.*

Sometimes two or more tasks within the same machine need to share a single file. They can do this under two conditions:

1. If they concurrently open and use multiple Data Control Blocks (DCBs) that refer to the same, shared file .

   When many DCBs refer to a single file, the type of processing (input, output, or update) decides what programming procedures you should use. Table 12 shows you the different types of processing and the requirements that go along with each.

Table 12. Opening Multiple DCBs

Type of processing:	**Programming required:**
INPUT	Each task should issue READ and GET requests as if no file sharing were taking place. GCS keeps track of the read pointers.
OUTPUT	This sort of sharing is not supported for multiple DCBs. Unpredictable results will occur if you attempt it.
UPDATING (in BSAM)	Each task should issue ENQ before the READ macro. This helps serialize the processing of each block of records. Macros issued to complete the update are WRITE, CHECK, and DEQ, in that order. For more information on these macros, see the *VM/SP Group Control System Command and Macro Reference.*
UPDATING (in QSAM)	When updating a file, a task must avoid altering blocks containing records that other tasks are updating. GCS has no way of knowing whether different tasks are processing discrete blocks.

   Note: When you share a file with multiple DCBs, be sure you enter the FILEDEF command only once for each ddname. If you need to issue FILEDEF for the same ddname and same file later in the program, make sure you specify the NOCHANGE option. See the *VM/SP Group Control System Command and Macro Reference* for the FILEDEF command format.

2. If they concurrently open and use only one shared DCB.

   When tasks share a single DCB, GCS permits all three types of processing (inputting, outputting, and updating). However, tasks have to use the ENQ and

DEQ macros to coordinate their activities. (See "Coordinating Shared Resources" on page 103.) Because only one of them can have control at a time, the tasks must issue the ENQ macro first (to take turns at getting control) and end with the DEQ macro (to release control).

Processing CP Spool Files

BSAM and QSAM functions let GCS programs process virtual reader, printer, and punch files. Existing CP facilities, like CP Directory, DEFINE, DETACH, SPOOL, TAG, and so on, define and manipulate the various unit record devices.

Note: GCS programs cannot write to virtual readers, nor can they read from virtual printers and punches.

Processing VSAM Files

GCS programs use VSAM macros supported at the MVS/VSAM Release 3.8 level, the same level as CMS. In fact, you will find them in a CMS macro library named OSVSAM MACLIB. When you request a service with one of these macros, it gets mapped to VSE/VSAM format and executed using VSE/VSAM code.

GCS's VSAM data management service supports the following command:

DLBL	Identifies VSAM files for I/O

GCS data management supports the following macros:

ACB	Generates an access method control block at assembly time
BLDVRP	Builds a resource pool for Local Shared Resources
CHECK	Suspends processing and waits for a request to complete
CLOSE	Disconnects a program and data
DLVRP	Deletes a resource pool
ENDREQ	Terminates a request
ERASE	Deletes a record
EXLST	Generates an exit list
GENCB	Generates an access method control block, exit list, or request parameter list at execution time
GET	Retrieves a record
MODCB	Modifies an access method control block, exit list, or request parameter list dynamically
OPEN	Connects a program and data
POINT	Points VSAM to a specific record to be accessed
PUT	Stores a record
RPL	Generates a request parameter list
SHOWCAT	Retrieves information from the VSAM catalog
SHOWCB	Displays fields of a control block or list
TESTCB	Tests values in a control block or list
WRTBFR	Writes buffers that contain Deferred Writes

Note: The control blocks generated by the OS ACB, RPL, and EXLST macros are converted from OS format to VSE format the first time that these control blocks are used by GCS. Because of this, the TESTCB, SHOWCB, and MODCB macros, rather than the OS mapping macros from the OSVSAM macro library, should get or modify data in these control blocks.

Here are some other VSAM information you should keep in mind:

- VSAM data management services support the CHECK macro and RPL's "ASY" option, but no asynchronous activity is performed.
- GCS does not support utility functions. You have to perform disk initialization, catalog definition, and file definition (AMS functions) under CMS.
- VSE/VSAM governs the sharing of VSAM data within a GCS virtual machine. The way you define a VSAM file and the way you use it (for input or output) determines how VSE/VSAM handles shared data. See the *VSE/VSAM Programmer's Reference* for more information.
- When a task terminates, GCS attempts to close all open ACBs that the task opened.

Planning for GCS involves the following:

- Being familiar with the current procedures that tell how to plan for and install shared segments
- Knowing the requirements of all products you plan to run on GCS
- Making entries in your VM/SP directory
- Reserving enough pages in storage to hold your GCS shared segment
- Preparing entries for your system name table (parameters for a NAMESYS macro in DMKSNT)
- Rebuilding CP to accommodate modifications
- Defining your GCS configuration file with the GROUP EXEC.

Overview of GCS Storage Layout

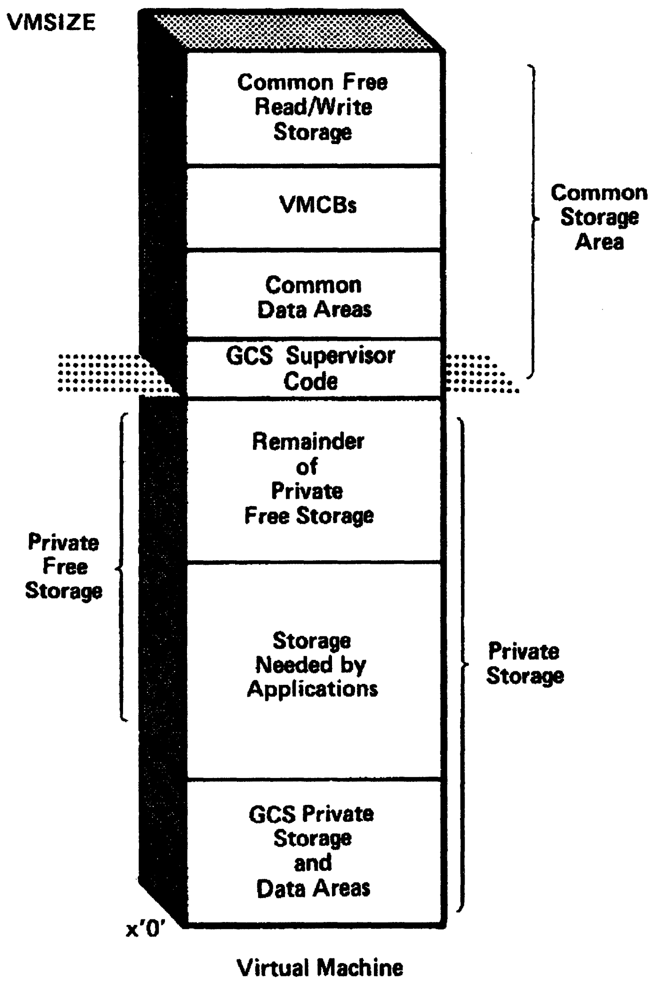

VMSIZE
Common Free Read/Write Storage
VMCBs
Common Data Areas
GCS Supervisor Code
Common Storage Area
Remainder of Private Free Storage
Storage Needed by Applications
GCS Private Storage and Data Areas
Private Free Storage
Private Storage
x'0'
Virtual Machine

Figure 17. GCS Storage Layout

GCS is divided into two pieces, private storage and common storage. Private free storage should be contiguous to make the virtual machine more efficient. Private storage is unique to each virtual machine that it is in, but common storage is shared by all users in the group.

Private Storage

Private storage is divided into two parts:

- GCS private storage
- Private free storage.

GCS private storage contains data areas and control blocks. These data areas and control blocks include system pointers, work areas, and the system configuration module. GCS private Storage begins at page 0 of the virtual machine.

Private free storage is storage that is available for GETMAINs and is where application programs are loaded.

Common Storage

Common storage is divided into many parts:

- GCS supervisor code
- Common data areas
- VMCBs
- Common free storage.

GCS supervisor code is the part of common storage that contains all executable modules required to IPL GCS.

Common data areas contains supervisor data that is shared between virtual machines.

VMCBs, Virtual Machine Control Blocks, there is one for each virtual machine in the virtual machine group.

Common free storage is storage that is used for GETMAINs, some of this storage is taken for the trace table to be created.

Common storage is a shared read/write area of the virtual machine. It is ideally located at the end of the virtual machine. By locating common storage at the end of the virtual machine you will avoid private storage fragmentation and will make the most efficient use of the virtual machine.

Planning GCS Storage Layout

In addition to calculating how much storage you need (See "Calculating Storage Requirements" on page 119), you also have to decide where to locate common storage in relation to private storage.

Common storage must begin at the same address for each group machine, an address determined by the *largest* application storage area needed in the group. Figure 18 on page 119 shows how the end of the larger application area in Virtual Machine No. 2 (and the private free storage that extends to a multiple of 4K) determines where private storage ends and common storage begins.

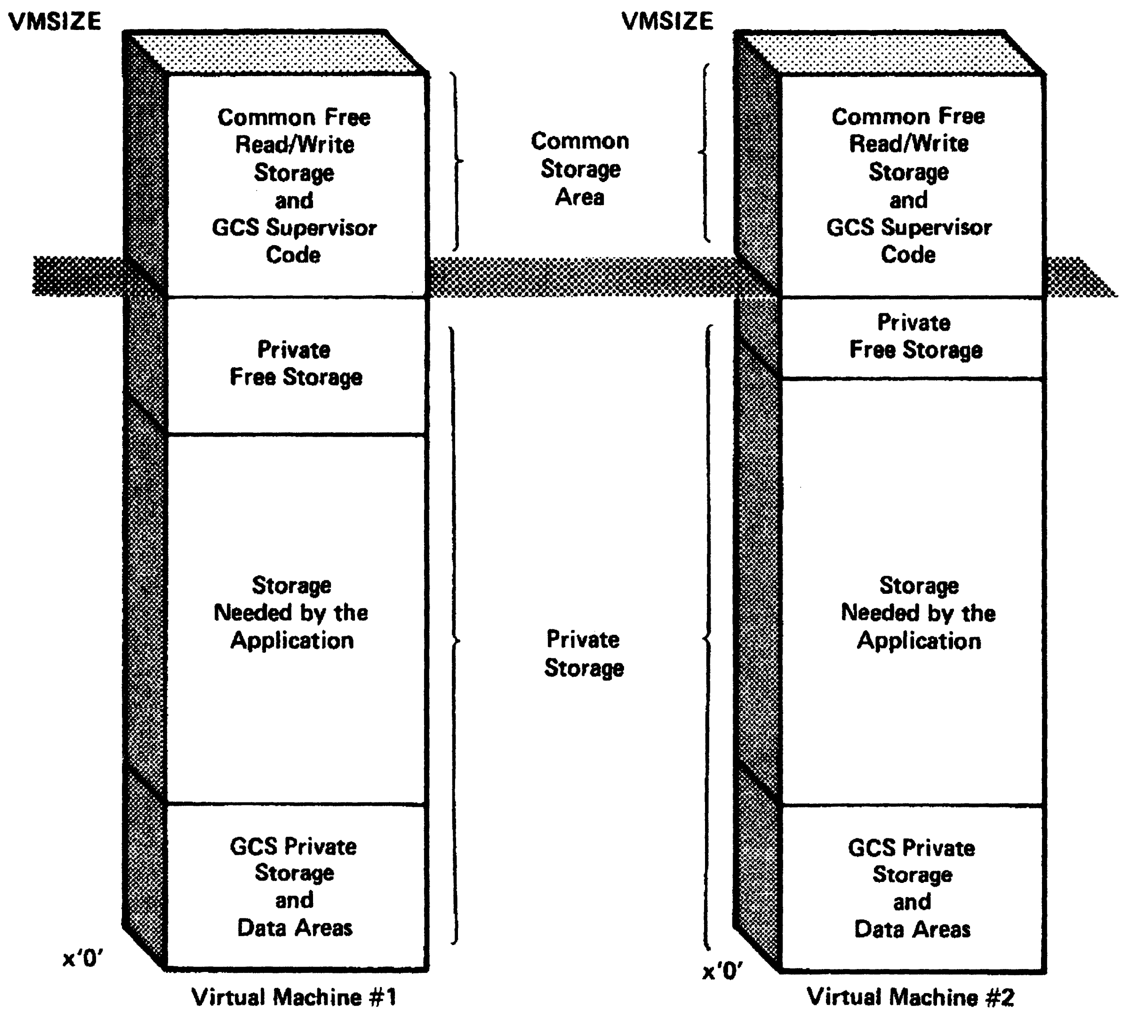

VMSIZE
Common Free Read/Write Storage and GCS Supervisor Code
Private Free Storage
Storage Needed by the Application
GCS Private Storage and Data Areas
x'0'
Virtual Machine #1
VMSIZE
Common Storage Area
Private Storage
Common Free Read/Write Storage and GCS Supervisor Code
Private Free Storage
Storage Needed by the Application
GCS Private Storage and Data Areas
x'0'
Virtual Machine #2

Figure 18. Ideal Locations of Common and Private Storage in Two Virtual Machine Group Members

By locating common storage above the largest application storage area, just inside the virtual machine's highest address (VMSIZE), you avoid fragmenting private storage.

Note: GCS common storage must be located within the VMSIZE of both the VTAM machine and the recovery machine. That way, both machines will be the same size. And, in case of an abend during a real I/O operation, the recovery machine (with DIAG98 in its directory entry) will be able to unlock any pages of storage locked by VTAM.

GCS common storage may exist outside the VMSIZE of other machines in the group.

Calculating Storage Requirements

The planning process involves calculating your GCS system's storage requirements. Later, you will use your findings to fill out parameters on the NAMESYS macro in your system name table (DMKSNT), and to fill in fields in the GROUP EXEC.

Here are guidelines to consider when reserving storage space for each of your GCS virtual machines:

Reserve space for private storage:

1. *How much for GCS private storage?*

   Seven pages of GCS private storage should be enough to hold your group configuration file along with certain control blocks and work areas that only the GCS supervisor has read/write access to. Figure 19 on page 122 shows 7 pages (0-6 on the SYSPGNM parameter) that have been saved for this purpose.

   Let's say you have a configuration file with five authorized user IDs and two shared segments. Such a file, together with the supervisor's control blocks and work areas, would fit within the 7-page (24K) estimate.

   Note: The space needed by the configuration file is your only variable here. (Most configuration files will take up less than 1K of storage.) However, if you have an exceptionally large configuration file, with long lists of authorized user IDs and shared segments, you may need more than 7 pages of GCS private storage.

   If you need more pages, you must make sure the entry point CSIBUFND in the module CSIBUF stays within the pages that are saved.

2. *How much private storage space for applications?*

   Each machine will have a different-sized application space because different applications have different requirements.

   You have to reserve application private storage space in 4K increments. However, most application sizes may not be even multiples of 4K. If the largest application code is not a multiple of 4K, you must round up to the next multiple of 4K and use this as your private storage size. Therefore, remaining space, between application code's end and the last 4K boundary is extra private free storage.

```
GCS private storage                          24576
(7 pages of 4K each)                         _____  bytes
       +
Largest contiguous block of storage needed
(determined by largest application run)      _____  bytes
       +
Amount needed to round up to 4K              _____  bytes
______________________________
Total
(Lowest possible beginning point for
 Common storage)                             _____  bytes
```

Some considerations when planning private storage include:

- Largest contiguous block of storage needed for application code
- Total amount of storage needed
- Location of GCS

- Location of other shared segments and DCSSs
- Size of the configuration module.

Reserve space for common storage:

Your common storage must be large enough to hold GCS supervisor code, common free storage space, a trace table, and control blocks required for each group member. Specifically, here's what to consider:

1. *How much space does the GCS supervisor code take up?*

   Approximately 252K. This amount should remain constant. Doublecheck it, in your load map, after you have built your GCS system.

2. *How much common free storage do each of your GCS applications require?*

   Some GCS applications will need common free storage space. ACF/VTAM, for example, requires large amounts of common free storage space for control blocks and buffers. (For the exact amounts, see each application's associated planning manual.)

3. *How many virtual machines will you have in your virtual machine group?*

   Each group member takes up one 24-byte control block (VMCB) in common storage. So, the more members you have, the more blocks you will have in common storage.

4. *What size trace table do you need?*

   Because the trace table also takes up space in common storage, you have to decide how much *history* you need in your storage dumps. The more applications you run and more activity you require of the GCS supervisor, the larger you need to make your trace table. The default size is 16K.

Use the following formula to calculate your common storage size:

```
Common free storage
(Total needed by all group applications)      ____  bytes
       +
Size of the supervisor                        252K
(approximately 252K)                          ____  bytes
       +
Control blocks (VMCBs)
(One 24-byte block for each group member)     ____  bytes
       +
Size of the trace table
(Default is 16K)                              ____  bytes
_________________________________
Total
                                              ____  bytes
```

Creating an Entry in the System Name Table

Once you have calculated your storage requirements, you can create a GCS entry in your system name table. If you are familiar with the process for building shared segments, you know you must edit the system name table (DMKSNT), define a new NAMESYS entry, and then assemble the revised system name table. Figure 19 shows a sample[18] NAMESYS entry for GCS.

```
*********************************************************************
*     HEX LOAD ADDRESS FOR SEGMENT  64 = 400000                     *
*     THE SPACE FOR GCS IS ALLOCATED ON VMPK01, AS FOLLOWS:         *
*         CYL 12, PAGE 1  TO CYL 13, PAGE 114  (264 PAGES)          *
*********************************************************************
GCS       NAMESYS SYSNAME=GCS,                                      X
              SYSVOL=VMPK01,                                        X
              SYSSTRT=(12,001),                                     X
              SYSPGNM=(0-6,1024-1279),                              X
              SYSPGCT=263,                                          X
              SYSHRSG=(064-079),                                    X
              SYSSIZE=256K,                                         X
              VSYSADR=595,                                          X
              SYSCYL=707,                                           X
              VSYSRES=VMPK01,                                       X
              PROTECT=OFF,                                          X
              VMGROUP=YES
          EJECT
```

Figure 19. Sample DMKSNT Entry for GCS.

The NAMESYS macro has many parameters. These parameters are all important but GCS is especially sensitive to the following parameters: SYSPGNM, SYSPGCT, SYSHRSG, and VMGROUP.

The following list describes each parameter on the NAMESYS macro:

Parameter	**Explanation**
SYSNAME	The name you give to your GCS system.
SYSVOL	The volume serial of the DASD you choose to receive the GCS shared segment. This must be a CP-owned volume.
SYSSTRT	The starting cylinder and page address on SYSVOL where GCS will be saved.
SYSPGNM	The page numbers to be saved. In this example, 7 pages (0-6) are saved for GCS private storage, while 256 pages (1024-1279) are saved for common storage.

[18] This DMKSNT entry is only an illustrative example. Your starter system contains a working sample.

Page 1024 identifies where common storage is located, when multiplied as follows:

```
    x'400' Page number "1024" (in hexadecimal)

X  x'1000' Size of 1 page (4096 bytes) in hexadecimal
 _________
 x'400000' Address where common storage begins
```

SYSPGCT The total number of pages you want saved for GCS. This example shows 262 (equal to the number of pages specified on SYSPGNM).

SYSHRSG A list of numbers identifying the shared segments that GCS common storage resides on. You may specify the segment numbers singly or in groups. For example, if you want to share segment numbers 0, 2, 4, and 64 through 79, use the format: SYSHRSG=(0,2,4,64-79). You obtain these shared segment numbers by dividing the first page number of common storage ("1024" from SYSPGNM) by 16 (the number of pages per segment). In this case:

$$\frac{1024}{16} = 64$$

That gives you the first segment number 64. It will hold 16 pages (1024-1039). The next 16 pages (1040-1055) will go on segment 65, and so on. Each shared segment is 64K long (4K x 16 pages).

SYSSIZE The minimum amount of storage GCS requires to run.

VSYSADR The virtual address of the system disk. (You will have to give each user ID a link to this disk before it can IPL your GCS segment.)

SYSCYL The cylinder address of the minidisk where GCS will be saved.

VSYSRES The volume serial of the DASD where GCS will reside.

VMGROUP A signal that this system has a virtual machine group associated with it, and any user ID that wants to IPL it must be linked to the disk specified on VSYSADR. Specifying "YES" automatically implies that PROTECT=OFF.

Preparing to Build Other Shared Segments

The process for setting up a shared segment for use with GCS involves making decisions similar to those you make when setting up and defining other shared segments or saved systems.

For a user-defined shared segment to be usable in GCS the various entry points it contains must be defined in a directory. This directory contains the name of each entry point in the saved segment mapped to its address. Use the CONTENTS macro instruction to create such a directory. See the *VM/SP Group Control System Macro Reference* for more information on the CONTENTS macro. To build a shared segment for GCS, you must:

1. Create a NAMESYS entry for the segment in the system name table DMKSNT. (See "Creating an Entry in the System Name Table" on page 122 and "Preparing the System Name Table File (DMKSNT)" on page 377.)
2. Create a directory for the segment with the CONTENTS macro. (See the *VM/SP Group Control System Macro Reference* for a description of the

CONTENTS macro.) This directory will reside in its own module and will contain the name and entry point of every routine in the segment.

3. Load this directory module, followed by all the other modules you want in the segment, at the desired location in storage. Use the CMS LOAD command with the ORIGIN option.
4. Set the segment's storage key using the SETKEY command.
5. Enter the CP SAVESYS command to actually save the segment you have built.

See Chapter 10, "Preparing the System Name Table File (DMKSNT)" on page 369 for more detailed information on saved segments.

There is another type of saved segment known as a *discontiguous saved segment.* These segments cannot be located within the virtual machine. These segments must remain outside (or *discontiguous* from) the boundaries of your GCS virtual machines. If you want to build a saved segment, simply follow the procedures described in Chapter 10, "Preparing the System Name Table File (DMKSNT)" on page 369 and the *VM/SP Installation Guide* . Unlike the procedure for building other GCS segments described in "Preparing to Build Other Shared Segments" on page 123, you **do not** create a directory module using the CONTENTS macro.

Figure 20 shows space reserved for CMSBAM and CMSVSAM saved segments beyond the address where common storage ends.

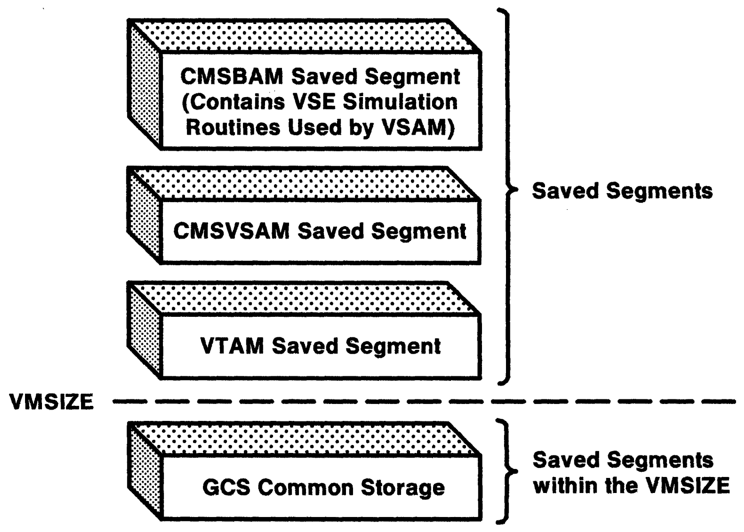

CMSBAM Saved Segment (Contains VSE Simulation Routines Used by VSAM)
CMSVSAM Saved Segment
VTAM Saved Segment
Saved Segments
VMSIZE
GCS Common Storage
Saved Segments within the VMSIZE

Figure 20. Sample Layout of Saved Segments

Both GCS and CMS share the same CMSBAM and CMSVSAM segments. However, GCS does not require the CMSDOS and CMSAMS segments for VSAM. See the *VM/SP Installation Guide* for information on how to build VSAM.

Rebuilding CP

Once you have edited and assembled the system name table (DMKSNT), you have to rebuild CP to put your modifications into effect. For instructions on rebuilding CP, see the *VM/SP Installation Guide.* After rebuilding CP, you have to IPL CP and re-IPL CMS before you can go on to define your GCS group environment.

Making VSAM Available to GCS

If you want your GCS applications to use VSAM data sets, you must install the VSE/VSAM product. You add VSAM to your system configuration as another shared segment. Its installation follows the same steps described in the *VM/SP Installation Guide*. Once installed, both your CMS and GCS applications can access VSAM data sets.

Authorizing Access to Supervisor State

You can control access to supervisor state by revising your list of authorized user IDs with the GROUP EXEC. Invoke the GROUP EXEC, and go to the screen marked "Authorized VM User IDs." Follow the directions there for adding, changing, or deleting entries. By doing that, you can provide or deny user IDs access to supervisor state and authorized GCS functions. After making changes with the GROUP EXEC, generate an updated GCS nucleus.

Authorizing Access to GCS

After you have installed your GCS segment, you can still control who has access to it. On one level, you decide which user IDs can IPL the GCS system: either you provide them with a link to the GCS 595 system disk, or you remove that link. Specifying a LINK statement in each user ID's VM/SP directory entry helps automate the process. In addition, specifying ECMODE on the directory entry's OPTION statement eliminates the need for manually setting ECMODE ON before IPLing a GCS segment.

Authorizing Commands for Virtual Machines

If you add or delete authorized user IDs with the GROUP EXEC, you will probably need to change their **privilege classes** too. For example, to protect GCS code, you have to limit what CP commands unauthorized user IDs have access to. On the other hand, you might want certain authorized user IDs to have access to all available CP commands. By changing the privilege class specified on user IDs' USER control statements in the VM/SP directory, you affect which CP commands they can use.

You should restructure your system's privilege classes so that two like these are available:

1. **A privilege class for authorized user IDs**
   This class should be for GCS user IDs that need to use the CP debugging commands BEGIN, DISPLAY, DUMP, PER, STORE, and VMDUMP. It should give access to all current Class G commands except ADSTOP and TRACE.
2. **A privilege class for unauthorized user IDs**
   This class should be for GCS user IDs that do not need to use debugging commands. It should give access to all current Class G commands except ADSTOP, BEGIN, DISPLAY, DUMP, PER, STORE, TRACE, and VMDUMP.

Authorizing Machines for Real I/O

You choose whether your GCS machines will use real channel programs to drive real, attached I/O devices. The recovery machine, for instance, should be authorized to use real I/O. To authorize a virtual machine for real I/O, you have to change your VM/SP directory and specify the parameter DIAG98 on the OPTION control statement. (See Chapter 7, "Creating Your VM/SP Directory" on page 215 for more information).

Using VMSAVE

Another parameter you may want to specify on the OPTION directory control statement is VMSAVE. This parameter saves a virtual machine's storage if that machine or the entire system terminates for any reason other than logging off.

If you use VTAM with GCS, you may want to dump a copy of your GCS machines' storage contents in the case of a CP ABEND. VMSAVE can help. If you have the required DASD space available for each virtual machine you want to save, then it will save all or just selected pages of each virtual machine in that space. To access the information copied and saved, you must IPL the DASD location by name. In addition, you can print the information with either the DUMP or VMDUMP command. If you dump the information by way of VMDUMP, you can analyze it with IPCS (under CMS).

If you do not specify VMSAVE in your OPTION directory control statement, you can activate it instead by issuing a CP SET VMSAVE command.

Using AUTOLOG Functions

To make use of the CP AUTOLOG function for GCS, you need to make a VM/SP directory entry for each user ID you want logged on automatically.

The directory entry should look like this:

```
IPL GCS PARM AUTOLOG
```

where

`GCS`

is the name given to your GCS system.

Note: "PARM AUTOLOG" will not work properly if you try to issue it from an IPL instruction on your console's command line. Use PARM AUTOLOG only in directory entries.

If one user ID has this entry in its directory, a second user ID, having a privilege class of A or B, can log on the first one automatically with the CP AUTOLOG command. (See the *VM/SP CP System Command Reference* for more information about AUTOLOG.)

Using A PROFILE GCS File

You can identify load libraries and initialize GCS applications automatically in the PROFILE GCS. When you IPL your GCS system:

1. Saved segments (specified in your GCS configuration file) are linked to your virtual machine.
2. Disks are accessed.

3. Disks are searched for a file of name and type PROFILE GCS, and if there is one, it executes[19].

By setting up enough PROFILEs, you can automate logging and initialization procedures for most of your virtual machine group. Because the recovery machine must be the first to IPL GCS, you could give it a PROFILE that would automatically log on all other group members that have IPL GCS PARM AUTOLOG specified in their VM/SP directory entries. Be sure to assign the recovery machine a privilege class of either A or B so that it has authorization to issue the CP AUTOLOG command.

By defining an "AUTOLOG1" user ID in your VM/SP directory, you can have it automatically log on the recovery machine as well. See the *VM/SP Operator's Guide* for more details.

If you set up a PROFILE GCS, you cannot prevent it from executing. But if you find a problem with it, you can interrupt and stop it with:

- The HX (halt execution) command
- The BREAK or PA1 key.

After that, you can go back to CMS and change your PROFILE GCS file.

Preparing VM/SP Directory Entries

You may need to update your VM/SP directory and prepare new entries there for user IDs that will use GCS. Table 13 is an example[20] of a directory entry for a recovery machine user ID.

Table 13. Example of a GCS User ID's Directory Entry

Directory contents:	**Explanatory notes:**
USER GCS password 7M 16M G ACCOUNT GCS RECVM OPTION DIAG98 ECMODE IPL GCS CONSOLE 01F 3215 SPOOL 00C 2540 READER * SPOOL 00D 2540 PUNCH A SPOOL 00E 1403 A LINK MAINT 595 595 RR LINK MAINT 59E 59E RR	**7M** is your calculated storage requirement; **G** is a privilege class you have redefined. **DIAG98** is an option for real I/O; **ECMODE** sets ECMODE ON (required with GCS). IPL GCS **PARM AUTOLOG** (not shown) is required for any user ID you want automatically logged on. ** The section beginning on page 125 tells more about the OPTION, IPL, and LINK directory statements. ** All user IDs that need to IPL GCS must have the **LINK** to 595; the **LINK** to 59E is optional.

See Chapter 7, "Creating Your VM/SP Directory" on page 215 for instructions on how to create directory entries with the Directory program.

[19] PROFILE GCS must contain REXX code because the System Product Interpreter is the facility that processes it.

[20] This directory entry is only an illustrative example. Your product tape contains a working sample.

Operation

Operating GCS involves initializing GCS, starting and stopping programs, replying to messages, and querying information. This section describes each activity, and the *VM/SP Group Control System Macro Reference* describes the commands related to each activity.

Initializing GCS (How to Join a Group)

Initialization is simply the act of loading (IPLing) your GCS system. It also means the same thing as *joining a virtual machine group* or *IPLing the GCS supervisor*.

To join a group, you enter:

```
set ecmode on
```

(if not already specified in your directory)

```
cp link userid 595 595 RR password
```

(if not already specified in your directory)

```
ipl gcs
```

To leave a group, you do one of the following:

- Log off
- IPL another system
- Enter one of these CP commands:
  SYSTEM RESET
  SYSTEM CLEAR
  DEFINE STORAGE
  DEFINE CHANNELS
  SET ECMODE OFF
  SET ECMODE ON

Figure 21. Joining or Leaving a Group

You can enter the IPL command (with the name of your GCS saved system) from your virtual machine console[21]. For example, if you named your GCS system "GCS" at build time, you would enter:

```
ipl gcs
```

The system will respond with a system ID message (if you specified one with the GROUP EXEC) and a "generate" message. For example:

```
GROUP CONTROL SYSTEM
Generated at mm/dd/yy hh:mm:ss
CSIACC423I S (595) R/O
CSIACC423I A (191) R/W
Ready;
```

For instructions on how to specify a system ID in GCS, see the *VM/SP Installation Guide*.

[21] Or, you can automate the loading of your GCS system by using the CP AUTOLOG and auto-IPL procedures described under "Using AUTOLOG Functions" on page 126.

When you initialize GCS, any other shared segments you identified with the GROUP EXEC become linked to your virtual machine, and disks are accessed as shown in Table 14 on page 129.

Table 14. Automatic Disk Access at IPL

Device Type	Virtual Device Address	Access Mode
Primary Disk read/write	191	A
User Disk read/write	192	D
System Disk read-only	595[22]	S/S * * S2[23]
System Disk Extension read-only	59E[22]	Y/S * * Y2[23]

Once the disks are accessed, GCS searches them for a PROFILE GCS file and, if you have one, processes it. PROFILE GCS resembles the PROFILE EXEC in CMS and is described in the section on "Using A PROFILE GCS File" on page 126.

Once you have IPLed GCS and have the proper disks accessed, you can enter GCS commands to assign files and start applications. For example, these are the commands you would use to start RSCS operations:

```
global loadlib rscs
filedef config disk rscs config *
loadcmd rscs dmtman
```

You may enter these commands from your virtual machine console or place them in a PROFILE GCS to execute automatically at IPL time. The *VM/SP Group Control System Macro Reference* describes the commands available for use.

Starting and Stopping Programs

If you want a program to run on GCS, you have two choices:

1. Write your own.
   a. Write and compile it or assemble it using CMS.
   b. Put the resulting text files in a load library using the CMS LKED command.
   c. IPL your GCS segment.
   d. Use the GLOBAL command to identify the load library where the program resides.
   e. Execute and *debug* the program using GCS commands. (If you make any corrections to the program's source code, you have to do them using CMS and then reload the program in its load library.)

[22] If you specified other disks as your S-disk and Y-disk, then they will be accessed instead of 595 and 59E.

[23] You need at least one file of file mode S2 to access your system disk properly and one file of Y2 for your system extension disk.

2. Identify one that already resides in a shared segment. The program should be listed in the segment's directory—a directory created by the CONTENTS macro when the segment was built.

You can start programs in your virtual machine by entering:

1. The OSRUN command
   (or the name of an exec that will issue OSRUN)

   Use the OSRUN command to start programs that you want to load and give control to. When you enter OSRUN to start a program, GCS will not process any other commands (except immediate commands) until the program ends. The system will not accept other commands because it allows only one active *command* at a time. So, OSRUN remains the active command as long as the program is running.

   The program will stop automatically without prompting.

2. An application command (one you have defined with the LOADCMD command).

   This lets you invoke an application that will start itself either with an OSRUN command or an ATTACH macro with the JSTCB=YES parameter. If the application's start-up module uses the ATTACH macro to start, your initial application command remains active, *and* you still can enter other application commands.

   An application started with LOADCMD stops:

   - Automatically (when it finishes its work), or
   - When prompted (you enter the command name you defined and include the necessary stop parameter).

     If a program issued an ATTACH macro to start, it must issue the DETACH macro to stop the attached program.

You can stop programs during their execution with the HX command. HX also clears any commands defined with the LOADCMD command that are stacked and waiting to be processed.

Replying to Messages

When a program needs to communicate with you, it can send a message to your console and request your reply. For this, the program uses a WTOR macro (Write To Operator with Reply). It may ask you, as a GCS virtual machine operator, to set up certain devices, provide data, or carry out some other request.

To respond to messages sent by way of WTOR, you enter the REPLY command. Each message you respond to will have an id number associated with it. You use this id number to route your response.

Unlike CMS, GCS lets programs continue running even when you owe them many replies. If you want to check for messages that require replies, you can enter a QUERY REPLY command. This will display the id numbers and text of all messages waiting for replies.

Querying Information

Sometimes you need information about the status of your virtual machine. For example, you might want to see the search order of your accessed disks or to see if external tracing is active. You can find this information using the QUERY command. QUERY can report on:

- Whether internal recording of user trace events is enabled (QUERY ITRACE)
- Trace events that are enabled for recording in a spool file (QUERY ETRACE)
- User IDs of virtual machines in your GCS group (QUERY GROUP)
- The common lock's status—whether the lock is held and what user ID is holding it (QUERY LOCK)
- The id number and text of all messages waiting for a reply (QUERY REPLY)
- Any file definitions in effect (QUERY FILEDEF)
- The status and search order of accessed disks (QUERY SEARCH)
- The load libraries GCS will search for load modules (QUERY LOADLIB)
- Names of attached saved systems and saved segments (QUERY SYSNAMES)
- The current DLBLs in effect (QUERY DLBL)
- Information about accessed disks (QUERY DISK).
- All the entry points that were loaded by the LOADCMD command (Q LOADCMD)
- All entry point names and corresponding addresses that were loaded into this virtual machine (Q LOADALL).

Planning for TSAF

What TSAF Is

The Transparent Services Access Facility (TSAF) lets users connect and communicate with local or remote virtual machines within a group of systems. With TSAF, a user can connect to a program by specifying a name that the program has made known, instead of specifying a user ID and node ID.

Machine Requirements

VM/SP TSAF runs on any VM/SP Release 6 or equivalent supported processor. The processors must support at least one of the following connections between systems:

- Channel-to-Channel (CTC) links, including 3088 links and LANs
- Leased point-to-point lines with Binary Synchronous Communications (BSC) support. The error rate for these types of lines must be less than one error in 500,000 bits transmitted.
- IBM Token Ring Local Area Network (LAN) adapter on the IBM 9370 Information System processor
- IEEE 802.3 (Ethernet[24]) LAN adapter on the IBM 9370 Information processor.

The TSAF virtual machine runs in a CMS virtual machine with at least 4MB of virtual storage. See the *VM/SP Connectivity Planning, Administration, and Operation* book for information on setting up the TSAF virtual machine.

Planning for Advanced Program-to-Program Communication/VM

APPC/VM (Advanced Program-to-Program Communication/VM) is an application program interface (API) for communicating between two virtual machines in a collection and with APPC programs in an SNA network. APPC (Advanced Program-to-Program Communication) is the inter-program communication service within Systems Network Architecture LU 6.2 (SNA LU 6.2) on which the APPC/VM interface is based. The Common Programming Interface (CPI) Communications is a set of program-to-program communications routines that allow applications written in REXX and high-level languages to access APPC/VM functions. These routines are part of IBM's Systems Application Architecture™. VM also has some of its own routines that are considered extensions of CPI Communications. In a VM environment, CPI Communications maps to APPC/VM.

Programs that connect to a resource and the virtual machine owning that resource must follow the rules of an APPC conversation. These rules are described under "APPC Mapped with APPC/VM" in the *VM/SP Connectivity Programming Guide and Reference.*

A VM resource is a program, a data file, a specific set of files, a device or any other entity or set of entities that you might want to identify for application program processing. A global resource is one known to all systems in the collection. A

[24] Trademark of the Xerox Corporation.

Systems Application Architecture is a trademark of the International Business Machines Corporation.

logical unit (LU) is an entity addressable within an SNA-defined network, similar to a node within a VM network. LUs are categorized by the types of communication they support: LU 6.2 is for application program-to-program communication. A TSAF collection in an SNA network is viewed as one or more LUs. A gateway is the logical unit name (LU name) of a TSAF collection that is a source for communications to an SNA-defined network or the target of communications from an SNA-defined network.

Collection Structure

A collection is a group of VM/SP systems. Each system in the collection has a TSAF virtual machine component installed and running. These systems must be connected, directly or indirectly, by 3088, CTC, BSC, or LAN links that TSAF owns and controls. Two systems do not have to be physically connected if a route exists through another processor or set of processors. The maximum for a TSAF collection is eight processors.

You must define each system within the collection to have a unique node ID. Also, you must ensure that no two users in a collection have the same user ID. In addition, each global resource or gateway name within a collection must be unique. See the *VM/SP Connectivity Planning, Administration, and Operation* book for more details about collections.

Merging TSAF Collections

When two TSAF collections merge to form one, one or more global resource names or gateway names may be duplicated in each collection. TSAF determines the collection that wins management responsibility of duplicate resources and gateways. In some cases, the largest TSAF collection wins management of the resource or gateway in the new merged collection. See the *VM/SP Connectivity Planning, Administration, and Operation* book for more details about merging TSAF collections.

Routing

Applications that use local paths perform faster than applications that use TSAF for remote APPC/VM paths. The reason is that to communicate with a remote resource, you will need at least two APPC/VM paths and one TSAF path. Communicating with a local resource requires only one APPC/VM path.

For communications between systems in a collection, remember that, a CTC link and a LAN link are much faster than a BSC line. So, using BSC lines can slow down global TSAF functions significantly. See the *VM/SP Connectivity Planning, Administration, and Operation* book for more information about routing as it relates to performance.

Resources

You should be aware of the types of resources (local, global, and private), their management, and how to set up virtual machines to manage resources. These topics are described in detail in the *VM/SP Connectivity Planning, Administration, and Operation* book.

User programs request access to server programs by an eight-character name. Generally, you, as the system administrator, control which user programs are allowed to request services of various server programs. See the *VM/SP Connectivity Programming Guide and Reference* for information on how to write a user program and the *VM/SP Connectivity Planning, Administration, and Operation* book for more detailed information on user and server programs.

Planning for APPC/VM VTAM Support (AVS)

What AVS Is

AVS is a component of VM/SP. You may choose to install or not install it. It is a VTAM application and runs in a GCS virtual machine. AVS may run in the same virtual machine as VTAM or in a separate virtual machine in the same GCS group.

AVS, along with VTAM, provides the functions necessary for APPC/VM programs within a TSAF collection or in a single VM/SP system to communicate with APPC programs anywhere in an SNA network utilizing the LU 6.2 protocol. AVS uses APPC/VTAM to communicate with general APPC programs. AVS and VTAM provide the SNA network with a view of the TSAF collection. The SNA network considers the TSAF collection as one or more LUs. The LUs representing the TSAF collection to VTAM are called gateways and are defined by the AVS operator or administrator. You should make sure that each gateway name is unique within a TSAF collection.

AVS lets users realize the full communications potential for global and private resources, extending communications (or connectivity) to the SNA network. This lets general APPC programs on VM and non-VM systems communicate with programs that use APPC/VM assembler language programming interface or the Common Programming Interface (CPI) for Communications. (See the *VM/SP Connectivity Programming Guide and Reference.*)

Connectivity is extended in these ways:

1. To communications between TSAF collections:

   AVS and VTAM allow communications from one TSAF collection to another TSAF collection. For example, one collection may be able to access an SQL data base or Shared File System (SFS) data in another collection.

2. To resources in a TSAF collection (inbound requests):

   AVS and VTAM let users (running the appropriate APPC programs), from outside the TSAF collection, allocate conversations with global and private resources in the TSAF collection without having to be logged on within the TSAF collection. AVS translates inbound requests from APPC/VTAM to APPC/VM.

3. From APPC/VM programs in the TSAF collection (outbound requests):

   AVS and VTAM let users run APPC/VM programs (in a virtual machine on a VM system) that can communicate with APPC programs outside of the TSAF collection on which the APPC/VM program is running. AVS translates outbound requests from APPC/VM to APPC/VTAM.

Both the system administrator and the AVS operator have responsibilities for global and private resources, AVS, and the SNA network. You should be familiar with the content of the *VM/SP Connectivity Planning, Administration, and Operation* book before setting up and activating AVS.

Machine Requirements

AVS runs on any VM/SP Release 6 supported processor.

Security Considerations

The *VM/SP Connectivity Planning, Administration, and Operation* book contains information on resource security.

AVS Tuning Parameters

IBM supplies a nonexecutable source module, AGWTUN ASSEMBLE, containing default tuning values. AGWTUN ASSEMBLE contains defined constants (DCs) that you, in doing system administration, can modify. Changing these values might improve performance in your installation. On the other hand, you could degrade performance. See the *VM/SP Connectivity Planning, Administration, and Operation* book for a description of these parameters. To change the IBM-supplied default values, simply edit the AGWTUN ASSEMBLE module and modify the selected values. For them to become effective, reassemble the module and relink the resulting AGWTUN TEXT file with the other AGWxxx text files.

AVS Accounting Exit

IBM also supplies a source module, AGWACI ASSEMBLE, as an accounting interface. You can modify this IBM-supplied module or replace it with your own assembler program to collect accounting information. The main function of AGWACI is to write accounting records to a VM/SP CP spool file by using the DIAGNOSE code X'4C', subcode X'0010' instruction. (See the *VM System Facilities for Programming* book for a description of the DIAGNOSE code X'4C' instruction.) AGWACI also contains DSECTs of the accounting data that is passed to it. You may wish to replace this module with your own routine to redirect or reformat accounting data.

This routine is written in assembler language and may be assembled with the CMS ASSEMBLE command or with VMFASM.

AVS Common Message Repository

AVS supplies two message repository files:

- AGWCMR REPOS (American English mixed case)
- AGWCMRB REPOS (American English uppercase only).

The source files are shipped with the product and should not require changing unless you want to change them to another national language. The AVS loadlibrary (AGW LOADLIB) is shipped prebuilt with the AGWCMR REPOS (mixed case English) file built in. You will not have a choice during the installation process. If you decide to rebuild, you can decide to put in a different REPOS file.

See the *VM/SP Connectivity Planning, Administration, and Operation* book for using the AVS message repository.

Planning for Installing and Running AVS

To install and use AVS, you must also install GCS and VTAM. You should be familiar with GCS considerations before installing AVS. (See "Planning for GCS" on page 89, the *VM/SP Group Control System Command and Macro Reference*, and the *VM/SP Installation Guide*.)

For AVS to function as a VTAM application, you must define it as such to VTAM through the APPL macro during system definition. Include APPC=YES for VTAM LU 6.2 support. The following is an example of what you might include on your VTAM application definition (see the *VM/VTAM Installation and Resource Definition* book for more information on defining VTAM resources):

```
AGWAPPL1 VBUILD TYPE=APPL                                   X
AGWGAT01 APPL  APPC=YES,                                    X
               AUTHEXIT=YES,                                X
               DSESLIM=100,                                 X
               DMINWNL=50,                                  X
               DMINWNR=50,                                  X
               MODETAB=AGWTAB,                              X
               PARSESS=YES
```

APPC=YES
: tells VTAM that this will be an LU 6.2 APPC application. You cannot use APPC without it.

AUTHEXIT=YES
: lets an application program such as AVS run with its exits authorized. If this is not coded, then VTAM and GCS will do validation checking on any data coming through and will severely impact processing time. This should be included.

DSESLIM=100
: maximum number of sessions between a local and remote LU per mode. It must be greater than or equal to DMINWNL + DMINWNL. This is also the limit for any given CNOS.

DMINWNL=50
: minimum number of contention winners guaranteed for a local LU. Actually negotiated on a CNOS between a remote and local LU for a particular mode.

DMINWNR=50
: same as DMINWNL except it is for a remote LU (minimum number of contention winners guaranteed for a remote LU).

MODETAB=AGWTAB
: identifies the default table for picking up mode entries (binds).

PARSESS=YES
: lets the application program have multiple LU-LU sessions with another application.

How to Bring Up AVS: Before you can run AVS, you must do one of the following:

1. Install it under its own virtual machine as a member of the GCS group.
2. Set it up in the same virtual machine as VTAM, with the VTAM profile containing the necessary GLOBAL, LOADCMD, and START entries.

We recommend that you run AVS in its own virtual machine (it has its own user ID) with automatic logon. After VTAM is autologged, the AVS virtual machine will be brought up automatically and started each time you bring up the system if you:

1. Make sure that AUTOLOG1 has an entry for the AVS virtual machine user ID (AVSVM):

   ```
   AUTOLOG AVSVM password
   ```

   This automatically logs on the AVS virtual machine, and CP will examine the VM/SP directory entry for AVSVM (see next step).

2. Make sure the VM/SP directory entry for user ID AVSVM has:

   ```
   IPL GCS PARM AUTOLOG
   ```

   This automatically IPLs GCS.

3. Make sure the following three statements are in the AVS virtual machine's PROFILE GCS (IBM ships these in the sample PROFILE GCS for AVS):

   ```
   'GLOBAL LOADLIB ... AGW ...'
   'LOADCMD AGW AGW'
   'AGW START'
   ```

   'GLOBAL LOADLIB ... AGW ...'
   : identifies the necessary load libraries (other parameters are included, but not shown). AGW identifies the AVS load library.

   'LOADCMD AGW AGW'
   : LOADCMD defines the command name and the CMS LOADLIB member; the first AGW defines the command named "AGW" to GCS. The second AGW is the module name for the AVS virtual machine and must remain AGW. It tells GCS what module to send the command to. For a complete description, refer to the *VM/SP Group Control System Command and Macro Reference.*

   'AGW START'
   : starts up AVS. (Any AVS command other than AGW START has to be in AGWPROF GCS or issued from the console after AVS initialization is complete.)

Your AVS virtual machine's PROFILE EXEC (AGWPROF GCS) is automatically executed. As part of your AGWPROF GCS, you can enter AVS commands, such as ACTIVATE in the following example:

```
'AGW ACTIVATE GATEWAY AGWBA12 GLOBAL'
```

Activates gateway AGWBA12 and indicates that it is an LU that may be used by programs outside of the TSAF collection to allocate conversations with TSAF collection global resources.

You could have AVS run in your VTAM virtual machine instead of in its own virtual machine (you will still want to have VTAM autologged). If you choose to do this, you would have to include the

```
GLOBAL LOADLIB VTAM AGW ...
LOADCMD AGW AGW
AGW START
```

as part of your VTAM profile (for example, VMVTAM GCS). (For information on the requirements on installing VTAM, see the VTAM product documentation.)

Planning for a Communications Directory

The CMS Communications Directory service is a means for easier access to target resources by providing symbolic names. A CMS communications directory is a special CMS NAMES file that assigns symbolic names to actual resources. This lets communications programs connect to a resource by a symbolic destination name rather than the *resource name* and locally known *LU name*. Using symbolic destination names, user programs can transparently access resources within the same system, within a TSAF collection, or across an SNA network.

Setting up a communications directory can enhance the security and usability of VM/SP communications. It supports *local*, *private*, and *global resource* connections and can cause the redirection of an APPC/VM connection request without changing current application programs. There are two levels of communications directories: the *system-level* and the *user-level*.

The system-level communications directory shipped with the system, SCOMDIR NAMES, is a general-purpose communications directory. The user-level communications directory shipped with the system, UCOMDIR NAMES, contains requester-specific communication information and is designed for file mode A usage.

Using the unchanged SYSPROF EXEC shipped with the system, when the system needs to resolve a symbolic destination name, CMS first checks the user-level directory, if it exists. If the user-level does not exist or does not contain the symbolic destination name, CMS searches the system-level directory for that name.

For details on setting up and using CMS communications directories (including examples), see the *VM/SP Connectivity Planning, Administration, and Operation* book.

Chapter 5. Extending VM/SP

Contents of Chapter 5

Planning for CMS VSAM and Access Method Services

Introduction

CMS supports interactive program development for Operating System (OS) and Virtual Storage Extended (VSE) programs using Virtual Storage Access Method (VSAM). It also supports VSAM macros used in CMS programs. All of the VSE/VSAM macros and their options and a subset of the OS/VSAM macros are supported by CMS.

Access method services to manipulate OS, VSE VSAM, and SAM data sets, and VSAM for use with DOS/VS SORT/MERGE are also supported by CMS.

Under CMS, VSAM data sets can span up to 25 DASD volumes. CMS does not support VSAM data set sharing. However, CMS does support the sharing of minidisks or full pack minidisks. Only one user may have write access to the VSAM master catalog, but many other users may read and reference the catalog.

VSAM data sets created in CMS are not in the CMS file format. Therefore, CMS commands currently used to manipulate CMS files cannot be used for VSAM data sets that are read or written in CMS. VSAM data sets cannot be stored in a CMS Shared File System (SFS) file pool.

Because VSAM data sets in CMS are not a part of the CMS file system, CMS file size, record length, and minidisk size restrictions do not apply. The VSAM data sets are manipulated with access method services programs running under CMS, instead of with the CMS file system commands. Also, all VSAM minidisks and full packs used in CMS must be initialized with the Device Support Facility or an appropriate VSE or OS/VS disk initialization program (if the minidisk is a full pack); the CMS FORMAT command must not be used.

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

Hardware Devices Supported by VSE

CMS support of VSAM data sets is based on VSE/VSAM. Disks supported by VSE/VSAM can be used for VSAM data sets in CMS. These disks are:

Device No.	Model No.
3310	Direct Access Storage
3330	Disk Storage, Models 1, 2, and 11
3340	Direct Access Storage
3344	Direct Access Storage
3350	Direct Access Storage
3370	Direct Access Storage Models A1, A2, B1, and B2
3375	Direct Access Storage
3380	Direct Access Storage
9313	Direct Access Storage
9332	Direct Access Storage Models A01, B01, 402, 600, and 602
9335	Direct Access Storage Models A01 and B01

When the VM/SP processor is attached to a Mass Storage System (MSS), the CMS disk may be defined as a 3330 Model 1 that is mapped by VM/SP to all or part of a 3330V volume.

CMS disk files used as input to or output from Access Method Services can reside in an SFS file pool or on any disk supported by CMS.

Data Set Compatibility Considerations

CMS can read and update VSAM data sets created under VSE/VSAM or OS/VS. VSAM data sets with physical record sizes .5K, 1K, 2K, or 4K created under CMS can be read and updated by OS/VS VSAM. See the *VSE/VSAM General Information Manual* for more information about VSE/VSAM and OS/VS VSAM.

If you perform allocation on a minidisk in CMS, you cannot use that minidisk in an OS virtual machine in any manner that causes further allocation. VSE/VSAM (and thus CMS) ignores the format-5, free space DSCB on VSAM disks when it allocates extents. If allocation later occurs in an OS machine, OS attempts to create an accurate format-5 DSCB. However, the format-5 DSCB created by OS does not correctly reflect the free space on the minidisk because OS expects it to be a full pack. In CMS, allocation occurs whenever data spaces or data sets are defined, and space is released whenever data spaces, catalogs, and data sets are erased.

ISAM Interface Program (IIP): CMS does not support the VSAM ISAM Interface Program (IIP). Thus, any program that creates and accesses ISAM (indexed sequential access method) data sets cannot access VSAM key sequential data sets.

There is one exception to this restriction. If you have (1) OS PL/I programs that have files declared as ENV(INDEXED) and (2) if the library routines detect that the data set being accessed is a VSAM data set, your programs will execute VSAM I/O requests.

Planning Considerations for Installing VSAM Under CMS

CMS support of VSAM and Access Method Services is based on the VSE/VSAM program product. You must order the supported level of the VSE/VSAM program and use the VSAMGEN EXEC to install VSAM under CMS.

Support of VSAM under CMS also requires that the CMSDOS and CMSBAM saved segments be generated. See the *VM/SP Installation Guide* for more information on installing VSAM under CMS.

Planning for CMS/DOS

Introduction

Those of you who use CMS/DOS must, in certain cases, have available a VSE SYSRES. If you wish to use either the DOS/VS COBOL or PL/I compilers under CMS/DOS you must first order and install a VSE system and install the compilers on this system.

Note: CMS/DOS support is based on the VSE/AF 1.3.5 program product and does not support the VSE/AF 2.1 librarian.

If you plan to use CMS/DOS, you must also generate the CMSDOS and CMSBAM saved segments. These segments contain simulated VSE services that are necessary for running VSAM and other VSE programs under CMS. Running VSAM under CMS is dependent on the generation of CMSDOS and CMSBAM.

See the *VM/SP Installation Guide* for more details on installing CMSDOS and CMSBAM saved segments.

VSE System Generation Considerations

CMS/DOS support in CMS uses a real VSE system disk in read-only mode. CMS/DOS provides the necessary interface, and then gets VSE logical transients and system routines directly from the VSE system libraries. Also, CMS/DOS gets the DOS/VS COBOL and DOS PL/I compilers directly from the VSE system or private core image libraries.

It is your responsibility to order the latest VSE system and then generate it. Also, if you plan to use VSE compilers, you must order the DOS/VS COBOL and DOS PL/I optimizing compilers and install them on this VSE system.

When you install the compilers on the VSE system, you must link-edit all the compiler relocatable modules using the linkage editor control statement:

```
ACTION REL
```

You can place link-edited phases in either the system or the private core image library.

When you later invoke compilers from CMS/DOS, the library (system or private) containing the compiler phases must be identified to CMS. You identify all system libraries to CMS using the file mode letter that corresponds to that VSE system disk. Do this by specifying the file mode letter on the SET DOS ON command when you invoke the CMS/DOS environment. You identify a private library by coding ASSGN and DLBL commands that describe it. These VSE system and private disks must be linked to your virtual machine and accessed before you enter the commands to identify them for CMS.

CMS/DOS has no effect on the update procedures for VSE, DOS/VS COBOL, DOS/VS RPG II, or DOS PL/I. You should follow the usual update procedure for applying IBM-distributed coding changes to them.

When the VSE System Must Be Online

Much of what you do in the CMS/DOS environment requires that the VSE system pack and/or the VSE private libraries be available to CMS/DOS. In general, you need these VSE volumes whenever you do the following:

- You use the DOS/VS COBOL or DOS PL/I compilers. These compilers are run from the system or private core image libraries.
- Your DOS/VS COBOL or DOS PL/I source programs contain COPY, LIBRARY, %INCLUDE, or CBL statements. These statements copy code from your system or private source library. This function requires that the CMSBAM shared segment be generated and available to CMS/DOS.
- You invoke one of the librarian programs: DSERV, RSERV, SSERV, PSERV, or ESERV.
- You link-edit VSE programs that use nondisk LIOCS modules. CMS/DOS link-edits LIOCS routines with the VSE program from VSE system or private relocatable libraries.
- You run VSE programs that get phases directly from VSE system or private core-image libraries.

A VSE system pack is usable when it is:

- Defined for your virtual machine
- Accessed
- Specified, by mode letter, on the SET DOS ON command.

A VSE private library is usable when it is:

- Defined for your virtual machine
- Accessed
- Identified by the ASSGN and DLBL commands.

The VSE system pack and private libraries may reside on full packs or minidisks.

CMS/DOS Tape Handling

You can use the CMS tape label processing features described in the *VM/SP CMS User's Guide* to process tapes defined with a DTFMT. The features described there let you define input and output tapes that have standard or nonstandard labels or are nonlabeled tapes. They also let you specify your own exits for processing user standard or nonstandard labels. Before CMS prepares your tape files for processing, it returns control to the tape label processing routines.

The CMS LABELDEF command, which is described in the *VM/SP CMS User's Guide*, is equivalent to the VSE TLB control statement for standard label tapes.

When a tape is defined as a work file, it is treated as nonlabeled and any labels encountered on the tape are written over.

Tape labels are not supported on tape files defined with DTFCP or DTFDI. Existing IBM standard header labels are bypassed on such tapes when they are used for input and any existing labels are written over when the tapes are used for output.

CMS/DOS Disk Label Information Area

CMS/DOS does not support a disk label information area. If the real VSE system pack used by CMS/DOS has a label information area, it is not used.

In CMS/DOS, ASSGN and DLBL commands provide functions similar to those provided by the VSE ASSGN, DLBL, and EXTENT control statements. In VSE, those control statements are in effect only for one job. Thus, it is convenient to place often used DLBL and EXTENT control statements on the label information area.

However, in CMS/DOS, there is no such thing as a job. Consequently, ASSGN and DLBL commands remain in effect for an entire CMS/DOS session, unless they are reset by another ASSGN or DLBL command. Also, in CMS, you can place all the commands you need to compile and run a program in an EXEC file and invoke that EXEC file by its file name.

Planning for Virtual Machine Operating Systems (Other than CMS)

Introduction

This section contains information about:

- VM/VS Handshaking feature
- Multiple Virtual Machines Using the Same Operating System
- VM/SP Using Channel switching
- Alternate path support
- Operating systems using reserve/release
- Virtual Machine Communication Facility.

VM/SP Handshaking Feature

The VM/VS Handshaking feature is a communication path between VM/SP and certain other system control programs (such as OS/VS1) that makes each system control program aware of certain capabilities and requirements of the other. The VM/VS Handshaking feature consists of:

- Closing VM/SP spool files when the system control program's output writer operation is complete

- Providing an optional nonpaging mode for operating systems running under the control of VM/SP

- Providing miscellaneous aids for an operating system's virtual machine running under the control of VM/SP.

Because no paging is done by the operating system using VM/VS Handshaking, ISAM programs are treated by VM/SP as if they are being run from fixed storage locations. Therefore, to run ISAM programs successfully, the virtual machine directory must include the ISAM option.

When the handshaking feature is active, the operating system using VM/VS Handshaking closes the CP spool files by issuing the CP CLOSE command when a task or job has completed. Once these spool files are closed, they can be processed by VM/SP without operator interruption.

Operating systems using VM/VS Handshaking can run in nonpaging mode. Nonpaging mode exists when (1) the Handshaking feature is active, and (2) the operating system's virtual storage size equals the virtual storage size of the VM/SP virtual machine. When the guest operating system runs in nonpaging mode, fewer privileged instructions are executed and duplicate paging is eliminated. Such a virtual machine may have a larger working set when it is in nonpaging mode rather than when it is in paging mode.

Also, there are some other aids for guest systems using VM/VS Handshaking while running under the control of VM/SP. With the handshaking feature, the guest system avoids some of the instructions and procedures that would be inefficient under VM/SP.

When the VM/VS Handshaking feature is active, the operation of a system control program closely resembles the stand-alone operation because much repetition of function between VM/SP and the operating system is eliminated.

Multiple Virtual Machines Using the Same Operating System

In general, an operating system that runs in a virtual machine should have as few options generated as possible. This is also true when many virtual machines share a system residence volume. Very often, options that improve performance on a real machine have no effect (or possibly a negative effect) in a virtual machine. For example, seek separation, which improves performance on the real machine, is not needed in a virtual machine. CP itself issues a stand-alone seek for all count-key-data disk I/O.

Sharing the system residence volume makes it unnecessary to keep more than one copy of the operating system online. The shared system residence volume should be read-only so it can be shared among virtual machines. CMS saved segments can also be shared among all virtual machines. CMS/DOS simulates VSE/AF supervisor and I/O functions, thus allowing the running of many DOS programs. DOS and OS systems can be shared among users if all data sets with write access are removed from the system residence volume. See the *VM/SP Application Development Reference for CMS* for more details.

VM/SP Using Channel Switching

The two or four-channel switch can be used in the following cases:

- Two processors, one running VM/SP, the other running an operating system that supports channel switching.
- Two virtual machines running under VM/SP. Each virtual machine operating system must support the channel switch feature (CMS does not).
- A single virtual machine running under VM/SP. The virtual machine operating system must support the channel switch feature.
- A processor running VM/SP and managing multiple paths to devices through VM/SP alternate path support. See "Alternate Path Support" on page 149.

You can use the two or four-channel switch for devices attached to two processors. For example, one processor could be running VM/SP and the other could be running OS, as shown in Figure 22.

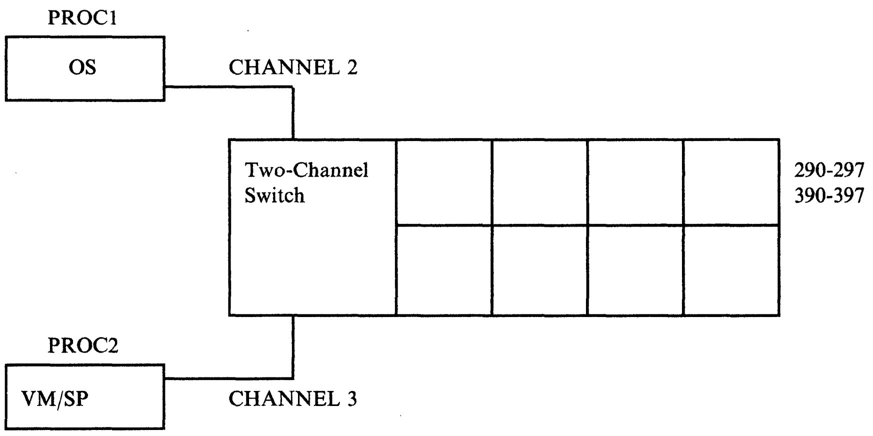


Figure 22. Channel Switching between Two Processors

VM/SP requires the following RDEVICE and RCTLUNIT macro instructions to support the following configuration:

```
RDEVICE  ADDRESS=(290,8),DEVTYPE=3330
RDEVICE  ADDRESS=(390,8),DEVTYPE=3330
RCTLUNIT ADDRESS=290,CUTYPE=3830
RCTLUNIT ADDRESS=390,CUTYPE=3830
```

These macro instructions make it possible for you to run VM/SP on PROC1 or PROC2. If you are always going to run VM/SP on PROC2, you can eliminate one path (eliminate one set of RDEVICE and RCTLUNIT macro instructions).

If any I/O devices controlled by CP for its own exclusive use are attached to a control unit by a two or four-channel switch, the processor 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 disks are CP-owned volumes (such as CP system residence and CP paging and spooling volumes), the OS system running on PROC1 must vary the CP-owned volumes offline. This procedure protects volumes that CP needs.

You can also use the two or four-channel switch for devices attached to one processor running VM/SP. For example, one processor could be running VM/SP with OS running in a virtual machine as shown in Figure 23. In this case, the virtual machine operating system supports channel switching.

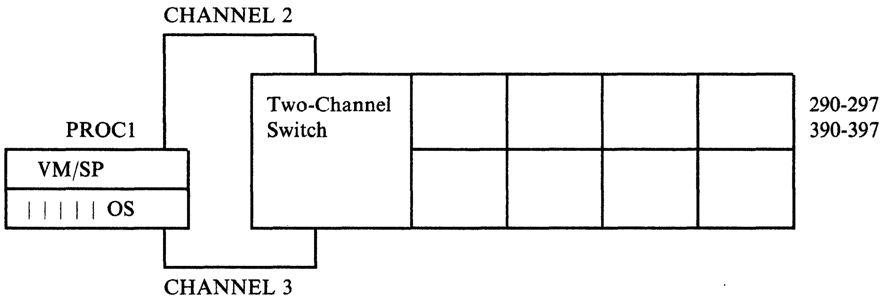


Figure 23. Channel Switching on One Processor

VM/SP requires the following RDEVICE and RCTLUNIT macro instructions to support this configuration:

```
RDEVICE  ADDRESS=(290,8),DEVTYPE=3350
RDEVICE  ADDRESS=(390,8),DEVTYPE=3350
RCTLUNIT ADDRESS=290,CUTYPE=3880
RCTLUNIT ADDRESS=390,CUTYPE=3880
```

For this example, you should have all devices associated with one path offline when you load VM/SP. Otherwise, the following message is displayed:

```
DMKCPI954E DASD raddr VOLID volid NOT MOUNTED, DUPLICATE OF DASD raddr
```

The 3880 Storage Subsystem contains two storage directors. Each storage director acts as a control unit providing I/O operations to a string of devices. Because each storage director can be addressed separately, you must code one RCTLUNIT macro statement for each module. The optional ALTCH operand of the RCTLUNIT macro allows specification of up to three alternate channel interfaces to a single storage director. The two or four-channel switch feature allows up to four channels to have access to a storage director. VM/SP supports a maximum of four channel paths to a single storage director.

DASDs can be used by OS running in a virtual machine if they are dedicated to that virtual machine by the ATTACH command or the DEDICATE control statement in the virtual machine directory entry. Device addresses generated for the virtual machine operating system need not be the same as those defined for the real machine.

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)
- Enter the CP ATTACH command for both device addresses (580 and 680) whenever the real device is to be attached to the virtual machine.

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 directory entry).

Alternate Path Support

Alternate path logic provides support for the two-channel switch, the two-channel switch additional feature, and the string switch feature. This support allows up to four channels on one control unit to be attached to VM/SP and/or one device to be attached to two logical control units. This allows the control program up to eight paths to a given device when the maximum number of alternate channels and alternate control units are specified. When an I/O request is received for a device, CP can select a free path from any of the available paths to the device. With this support, even though the primary path to a device is busy, there may exist an alternate path(s) that is available. Instead of the I/O request being queued, it can be started immediately on an alternate path. In the case where no available path to the device exists, alternate path I/O scheduling is implemented in such a way that the request is queued off multiple busy and/or scheduled paths. The first path to become available will be the path the I/O is started on. This approach has some distinct advantages over approaches used by other operating systems:

1. The I/O starts on the first available path to the device. This eliminates the arbitrary choice of queuing based on number of IOBLOKs already queued, primary path, last busy scheduled path encountered, and others.
2. No user is penalized more than any other user.
3. The first in, first out (FIFO) principle is maintained.

The goal of alternate path support is to define alternate paths to a device on the VM/SP processor. The virtual operating system does not define alternate paths. Instead, VM/SP defines alternate paths to the device with RCTLUNIT and REDEVICE macros. VM/SP then performs the alternate path I/O scheduling. If you wanted VM/SP, rather than the virtual operating system, to perform alternate path I/O scheduling, the following RDEVICE and RCTLUNIT macros would be required:

```
RDEVICE ADDRESS=(290,8),DEVTYPE=2314
RCTLUNIT ADDRESS=290,CUTYPE=IFA,ALTCH=(3)
RCHANNEL ADDRESS=3
RCHANNEL ADDRESS=2
```

To specify an alternate control unit on the RDEVICE macro, code:

```
RDEVICE ADDRESS=ccuu,DEVTYPE=nnnn,MODEL=n,ALTCU=ccuu
```

For example:

```
RDEVICE ADDRESS=(340,32),DEVTYPE=3330,MODEL=1,ALTCU=260
RCTLUNIT ADDRESS=340,CUTYPE=3830,FEATURE=32-DEVICE
RCTLUNIT ADDRESS=260,CUTYPE=3830,FEATURE=32-DEVICE
```

Figure 24 shows how the real I/O control block structure is coded, and how it logically appears when an alternate control unit is specified.

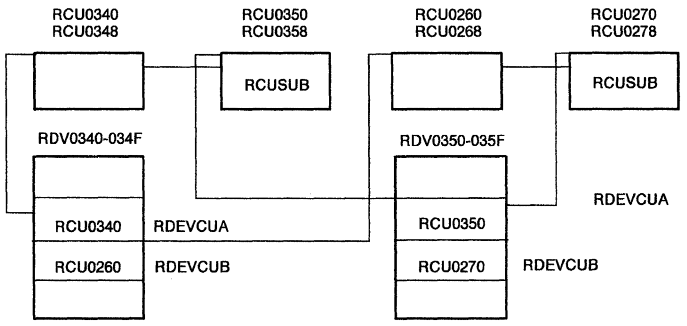


Figure 24. Real I/O Control Block Structure for Alternate Control Unit Specification

To specify alternate channel addresses on the RCTLUNIT macro, code:

```
RCTLUNIT ADDRESS=ccuu,CUTYPE=nnnn,FEATURE=xxx-DEVICE,    X
         ALTCH=(c,c,c)
```

Example:

```
RCTLUNIT ADDRESS=340,CUTYPE=3830,FEATURE=32-DEVICE,     X
         ALTCH=(1,2,4)
RCHANNEL ADDRESS=1,CHTYPE=MULTIPLEXOR
RCHANNEL ADDRESS=2,CHTYPE=MULTIPLEXOR
RCHANNEL ADDRESS=3,CHTYPE=MULTIPLEXOR
RCHANNEL ADDRESS=4,CHTYPE=MULTIPLEXOR
```

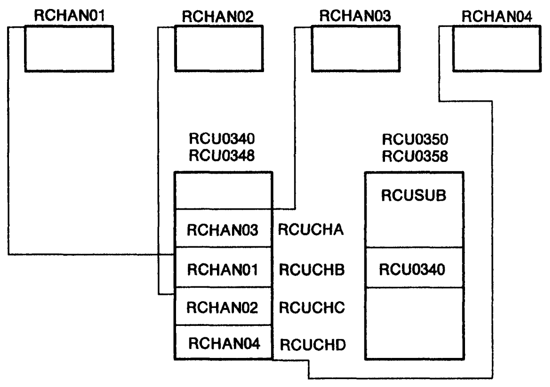


Figure 25. Real I/O Control Block Structure for Alternate Channel Specification

Figure 25 shows how the real I/O control block structure would be coded and logically appear when alternate channels are specified. Note that the subordinate control unit blocks do not contain pointers to the alternate channel blocks. Only the prime control unit block contains pointers to the alternate RCHBLOKS. This is consistent with the current CP block structure.

Restrictions

The following restrictions apply directly to alternate path processing:

- VM/SP does not support alternate paths for devices that issue attention interruptions to invoke a read response from the host; for example, the 3851 Mass Storage Control (MSC) unit.
- All devices on one physical control unit must be defined as either alternate path or no alternate path. There can be no splitting of control units when dealing with alternate paths.
- Only one alternate channel can be specified for MP configured systems.

Channel-Set Switching Facility

The channel-set switching facility is available on the 3033 attached processor and multiprocessor systems and the 308x and 3090 Processor Complexes. (It is not available on the 4381 Model 3 or Model 14 Processor.) This feature permits a set of channels to be switched from one processor to another in a multiprocessor or attached processor environment. A channel-set is the collection of channels that are switched as a group. On a 3033 attached processor system, all online channels comprise the channel-set.

VM/SP, when generated for AP operation, uses the channel-set switching facility. The switching operation directs the execution of I/O instructions and I/O interruptions from the main processor to an attached processor, thus permitting an operator to vary the main processor offline. The switching operation does not control other channel activity, such as data-transfer operations and chaining.

In 3033 and 308x attached processor environments, channel-set switching continues system operation in uniprocessor mode when the main (I/O) processor is taken offline as the result of a VARY OFFLINE PROCESSOR command or a main processor failure. This support switches the channel-set from the main processor to the attached processor.

There are no required system generation macro instructions to support channel-set switching. In the event of a failure on the main (I/O) processor, the automatic

There are no required system generation macro instructions to support channel-set switching. In the event of a failure on the main (I/O) processor, the automatic processor recovery routine determines if channel-set switching capability exists. If there is no channel-set switching capability in the system, CP enters the wait state with a code of X'0001'. If the error is TOD clock damage or a malfunction alert on the main (I/O) processor and the processor is in problem state, the failing processor is taken offline, provided that the attached processor is equipped with the channel-set switching facility. The channel-set switching feature disconnects the channel-set from the failing processor and to reconnect the channel-set to the attached processor. Processing continues on the attached processor in uniprocessor mode. This message is issued:

```
DMKxxx623I CHANNEL-SET CONNECTED TO PROCESSOR nn
```

Operating Systems Using Reserve/Release

Shared DASD is the capability of accessing direct access devices from two or more systems. The systems can be in virtual machines, on the same real processor, or on different real processors. Device access by the sharing systems is sequential.

Sharing of DASD volumes can occur when:

- A two or four-channel switch attaches a device's control unit to two or four channels
- String switching is used and the control units to which the DASD volumes are switched are on channels of two different systems.

With shared DASD, an I/O operation may be started to a shared device from any of the systems able to access the device using the switch. Each sharing system waits for the programmable switch to gain device access. The first requesting system gets the switch set to its interface so that it may perform I/O operations to a shared device. When the switch returns to neutral, any other system, or the same one, may select the shared device and have the switch set to its interface.

Note that none of the sharing systems is aware of what the other is doing with data on the shared devices. Data integrity is the responsibility of the using program. For this reason, a program can issue the RESERVE hardware command to retain exclusive use of a shared device while a critical update to data is being performed. Device RELEASE is issued to terminate exclusive reservation. If a shared device has been reserved for exclusive use, the system channel through which RESERVE was issued will lock out any other channel, on the same or different system, from accessing the device.

There are many reasons why you would elect to share devices between systems:

- Scheduling of jobs is simplified, and operator intervention is minimized. Instead of being moved from one system to another, the volume remains mounted and available to each system able to access the data by means of the two or four-channel switch or string switch.
- Updating of data is reduced. One update to a shared data set is needed, instead of the multiple updates required if each of many systems had its own copy of the data set.

- Backup and switchover in the event of hardware failure is made easier in a multisystem environment if the needed data is accessible to surviving systems without moving it.
- Direct access storage space may be saved because one copy of the data is required instead of multiple copies.

Two assembler language macros, RESERVE and DEQ, are available for reserving and releasing of a device. Data integrity of shared devices can be maintained by application program use of the RESERVE macro, or by operating system components that automatically issue the RESERVE macro if the target of their update operation is to a shared device. CMS does not make use of these macros in its CMS file system. In addition, CMS does not support these macros in OS simulation or CMS/DOS. The SHAREOPTIONS operand on the Access Method Services control statement has no function in CMS. No attempt is made by CMS VSAM to reserve or release system resources. Use of shared DASD by virtual machines should be limited to guest operating systems that will maintain the integrity of shared data, such as catalogs, VTOCS, program libraries, and so on. These guest operating systems should also support the use of the RESERVE and DEQ macros used by application programs running under these systems. The only other alternative is use of hardware reserve or release CCWs by an application program running under CMS. In this case, the application program issues hardware reserve and release CCWs in a SIO or DIAGNOSE operation to the shared device.

The reserve/release support can be addressed in two forms:

- Hardware-supported reserve/release
- Virtual reserve/release.

Hardware-supported reserve/release is the use of reserve/release CCW strings by virtual machine or processor operating systems to preserve data integrity. Data integrity is preserved by the hardware on a device basis during the interval of time between the reserve and release CCWs by not allowing access to the reserved device through any other path.

Virtual reserve/release is software simulation of reserve/release CCWs for minidisks. Because virtual devices associated with a minidisk all map to the same real channel interface to the device, hardware protection is lost, and a software locking structure is required to maintain data integrity during reserve/release sequences.

CP and CMS do not issue reserve CCWs. The use of reserve/release is the responsibility of the virtual machine operating system. The VM/SP initialization routine issues a release CCW to tape and DASD volumes to determine if the two or four-channel switch feature is installed.

Shared DASD

Operating systems with hardware-supported reserve/release use reserve/release CCWs to preserve data integrity in the following cases:

- Two virtual machines with each operating system having a separate channel path to the device to be shared. Each virtual operating system uses reserve/release CCWs to preserve data integrity.

  The reserve/release CCWs are recognized by the CP CCW translation routine and are executed by the hardware to preserve data integrity. In this case devices should be generated, at system generation time in DMKRIO, as separate devices. Each device should be dedicated to a virtual machine by means of the ATTACH command or DEDICATE control statement in the directory.

- A virtual machine runs under VM/SP and shares a device with another processor. The operating system in the virtual machine uses reserve/release CCWs to preserve data integrity. The operating system running on the other processor can be VM/SP, in which case the virtual machine operating system uses reserve/release CCWs to maintain data integrity. It can also be a non-VM/SP operating system with reserve/release capability.

  To support this environment, the device should be dedicated to the virtual machine by means of the ATTACH command or DEDICATE control statement in the directory.

  In the above shared DASD environments, the use of reserve/release by virtual machine operating systems and alternate path support are mutually exclusive. CP changes a reserve CCW to a sense CCW when an alternate path is defined for the device. In a multiprocessor (MP) system, an alternate path is defined if both processors have a channel path to the device. The protection offered by hardware reserve is lost. A single path should be defined in VM/SP for devices that will be dedicated to virtual machines, and then shared between other virtual machines or processors.

The device can be defined as a minidisk on the VM/SP processor, which begins at real cylinder 0. Do not use virtual reserve/release support in this environment. The volume being shared should not contain more than one minidisk or be used for CP paging, spooling, and others, because reservation by the other processor could lock out virtual machine users or VM/SP system I/O requests to the same device.

When an I/O error occurs on a device, the processor maintains a contingent connection for that device until a SENSE CHANNEL command is executed and sense data is recorded. No other I/O activity can take place on the device during this time. Under VM/SP, the contingent connection is maintained until the SENSE command is executed, but I/O activity from other virtual machines or another processor can begin on the device while the sense data is being reflected to the virtual machine. The user should therefore be aware that, on a shared disk, the access mechanism may have moved during this time.

The performance of virtual machine operating systems may be degraded when sharing DASD. If not running in single processor mode, when a device is being used by another virtual machine or system, I/O requests may be queued and the virtual machine left in an I/O wait state. If running in single processor mode, a device busy state is reflected up to the V=R guest when a device is in use and a device end interrupt is reflected when an interruption occurs. Depending on contention for devices, the V=R guest may get multiple device busy states and device end interrupts before the device is available to it.

Virtual Reserve/Release

The reserve/release software simulation in VM/SP provides reserve/release protection at the minidisk level, including full volume minidisks. Virtual reserve/release is used by virtual machines that support shared DASD (not CMS) running on the VM/SP processor. Virtual reserve/release simulation is requested by appending a character "V" to the mode operand on the MDISK directory statement. All future links to this minidisk are subject to virtual reserve/release processing. A software locking structure is created to manage the reservation status by minidisk. CP then examines virtual machine channel programs and manages reserve/release CCWs presented by sharing virtual machines. CP simulates hardware reserve by reflecting a *device busy* condition in response to a virtual machine SIO when the minidisk is already reserved by another virtual machine. When the minidisk is released, a *device end* interrupt is reflected to all virtual machine users who received a *device busy* indication. DIAGNOSE users can also issue reserve/release CCWs. No *device busy* or *device end* status, however, is reflected to the virtual machine. If a minidisk is already reserved, a subsequent DIAGNOSE request for another virtual machine is queued until the minidisk is released, when the DIAGNOSE request will be redriven.

Control Program Handling of a Reserve CCW

The reserve/release support and alternate path support are mutually exclusive. The CCW translation routine changes a reserve CCW to a sense CCW when alternate paths have been defined to the device from the VM/SP processor. Data integrity is not preserved when sharing a device between processors or virtual machines if alternate paths are defined. In an MP system, an alternate path is defined if both processors have a symmetrical channel path to the device. When using virtual reserve/release to share a minidisk between virtual machines on the processor, CP still changes a reserve CCW to a sense CCW when alternate paths are defined to the real device; but because hardware reserve/release is simulated when virtual reserve/release is being used, data integrity is preserved when alternate paths are defined. Figure 26 identifies those cases when CP changes a reserve CCW to a sense CCW.

Type of Device	Alternate Path Support	Reserve/Release Executes in the Hardware (2-4 Channel Switch)	Virtual Reserve Release Requested (V Added to Mode in MDISK)	Sent by CP to Device	CCW Comnd Note
Dedicated DASD or Tape	Not defined	Not applicable	Not applicable	Reserve	1
	Defined	Not applicable	Not applicable	Sense	2, 6
Minidisk	Not defined	Yes	No	Reserve	1
	Not defined	Yes	Yes	Reserve	1
	Not defined	No	No	Reserve	3
	Not defined	No	Yes	Sense	4
	Defined	Not applicable	Not applicable	Sense	5

[1]Normal Operation. The command is passed unchanged to the hardware.
[2]When the VM/SP system has been generated with alternate path support for those devices, it prevents the devices from being reserved. This action causes CP to avoid a possible channel lockout. CP does not return any indication that the device was not reserved to the operating system issuing the CCW command.
[3]Without the two-channel switch special feature, CP sends the reserve/release CCW command unchanged to the hardware. However, the hardware rejects the command and does not reserve the device.
[4]Before sending the command to the hardware, CP changes the reserve CCW command to a SENSE CCW command, and places a virtual reserve on the minidisk. The real device is not reserved. The virtual reserve prevents other operating systems running under the same VM/SP system from accessing the minidisk. However, these same virtual operating systems may virtually reserve other minidisks located on the same real volume. Because the two-channel switch feature is not installed on the channels, only one address path goes to the device from the VM/SP processor. This path allows virtual reserve/release processing to send a SENSE CCW to the device, although the reserve CCW command is rejected by the hardware.
[5]When alternate paths to a device have been defined (by the ALTCU operand on the RDEVICE macro instruction and the ALTCH operand on the RCTLUNIT macro instruction), CP changes reserve/release CCW commands to SENSE CCW commands to prevent a possible channel lockout.
[6]In an MP system, an alternate path is defined if both processors have a symmetrical channel path to the device.

Figure 26. Summary of Reserve/Release Support

Restrictions

Device Sharing between Real Processors

- When a device is shared between processors and at least one of the processors is running VM/SP, the shared volume cannot contain more than one minidisk. The single minidisk may encompass the entire volume or a small portion of the volume. The remainder of the volume must not be referenced by CP for use as paging, spooling, and so forth, or by any virtual machine.

  Note: This restriction applies only when both virtual and hardware-supported reserve/release are used. Assume that two virtual machines use separate minidisks on the same real volume and that both minidisks are defined for virtual reserve/release.

  1. Virtual machine A issues a reserve to minidisk A, resulting in a RESERVE CCW to the real volume.
  2. Virtual machine B issues a release to minidisk B, resulting in a RELEASE CCW to the real volume.
  3. Another real machine can now write to that volume, including the minidisk A area.

- Devices shared between processors must not be generated in DMKRIO as having alternate paths. If there are multiple paths from the VM/SP processor to the shared devices, as well as a path from the same devices to another processor, the paths from the VM/SP processor cannot be generated in DMKRIO as alternate paths by way of the ALTCH or ALTCU macro operands. *This means that the definition of alternate paths in DMKRIO and the use of hardware-supported reserve/release are mutually exclusive.*

Device/Minidisk Sharing on a Single Processor

- If more than one path to a volume exists, DMKRIO may be generated so that each path is defined as a separate path, not as an alternate path. When this is done, each path can be attached or dedicated to a different user, and reserve/release CCWs entered by such users preserve data integrity. In this case, integrity is preserved by the hardware, not by the software reserve/release support. *Again, the definition of alternate paths in DMKRIO and the use of hardware-supported reserve/release are mutually exclusive.*

- A volume may be defined through the directory to contain one or more minidisks. Such minidisks must be identified through the MDISK statement as requesting virtual reserve/release support. These minidisks may then be shared between virtual machines that support shared DASD and data integrity is preserved by the use of reserve/release CCWs in the virtual machine channel program. Alternate paths may be defined to the device when using virtual reserve/release. The reserve CCW is still changed to a sense CCW, but data integrity is preserved by the virtual reserve/release code.

Mixed Mode Access to a Shared Device

The preferred way of sharing the same physical space with two different minidisk addresses is by the directory LINK statement. However, if two separate MDISK statements define the same physical DASD space, they *must* be defined with the same virtual reserve release capabilities and they should be equal in size.

Inter-User Communication Vehicle

The Inter-User Communication Vehicle (IUCV) provides a communication capability between virtual machines. The facility also supports communication within the same virtual machine and between a virtual machine and CP.

IUCV communication takes place between two communicators. Every communication has a source communicator and a target communicator. A communication occurs over a predefined linkage called a path. Messages are created, transmitted over the path, and then eliminated by IUCV. IUCV functions include the following:

- Communications paths and messages are initiated by either CP or a virtual machine
- A communicator can selectively establish and terminate communication paths
- Two communicators can establish more than one communication path between them
- More than one message can be transmitted in either direction at the same time, using the same path
- All IUCV functions are privileged
- All IUCV functions are invoked with the IUCV macro
- Directory authorizations allow an installation to control the establishment of IUCV communication paths between virtual machines and CP system services.

For a detailed description of IUCV functions and the IUCV macro instruction, see the *VM System Facilities for Programming* book.

The IUCV directory control statement defines authorizations for establishment of IUCV communication paths. The MAXCONN keyword of the OPTION directory control statement defines the maximum number of IUCV connections allowed for a virtual machine. For more information, see Chapter 7, “Creating Your VM/SP Directory” on page 215.

Although IUCV is similar to the virtual machine communication facility, IUCV does not replace VMCF. Both communication facilities are available and can coexist. IUCV should be considered for new applications requiring inter-user communication. IUCV is used for communication between System Network Architecture Console Communications Services (SNA CCS) and VCNA.

Virtual Machine Communication Facility

The Virtual Machine Communication Facility (VMCF) allows one virtual machine to communicate and exchange data with any other virtual machine operating under the same VM/SP system. The VMCF external interruption masking is controlled by PSW bit 7 and CR0 bit 31. It is to your advantage to always have CR0 bit 31 set to 1 (while VMCF is in use) and to control the interruptions with PSW bit 7 only. This reduces the number of LCTL instructions.

Messages and data directed to other virtual machines are logically identified by the virtual machine's user ID. Data is transferred in 2048-byte blocks from the sending virtual machine's storage to the receiving virtual machine's storage. The amount of data that can be moved in a single transfer is limited only by the storage sizes of the respective virtual machines.

Use of real storage is small. Only one real storage page need be locked during data transfer. A special interrupt notifies one virtual machine of a pending transfer of data. This interrupt also synchronizes the sending and receiving of data.

Under the special message facility, CP acts as a virtual machine for a virtual machine that issues SMSG. The receiving virtual machine, properly programmed to accept and process special messages, authorizes itself to CP. Data (message) transfer is from CP, through the message and VMCF modules.

Performance Guidelines

Introduction

The performance characteristics of an operating system running in a virtual machine are difficult to predict. Some of the factors are:

- Processor model
- System type (uniprocessor, attached processor, or multiprocessor)
- Total number of virtual machines running
- Type of work each virtual machine is doing
- Speed, capacity, and number of the paging devices
- Fixed-head cylinders for preferred paging
- Amount of real storage available
- Degree of channel and control unit contention, as well as arm contention, affecting the paging device
- Type and number of VM/SP performance options in use by one or more virtual machines
- Whether hardware assist exists
- Favored priority and V=R options in effect.

Also, the virtual machine's channel mode, block multiplexer or selector, has an effect on the virtual machine's performance.

Note: The performance of an MSS (Mass Storage System) being used by the operating system and shared with other systems depends on the total MSS usage and contention.

Performance Measurement and Analysis

VM/SP has two commands that tell you how VM/SP is performing. These commands are the MONITOR and INDICATE commands.

The MONITOR command controls the collection of performance data and the writing of the data to spool files or tapes. Both summary and trace data can be collected. You may specify classes of data to be collected using either the operands of the MONITOR command or the SYSMON macro. Which classes you select depends on what kind of analysis you want to do. You can use the IBM Field Developed Program (FDP) VM/370: Performance/Monitor Analysis Program (VMMAP) to reduce the amount of data you collect. The guidelines for using this program help you determine the overall load and performance profile of your system. The VM/370 Performance/Monitor Analysis Program enables you to analyze usage of and contention for major resources such as the processor, storage, and I/O paging subsystems.

The INDICATE command displays, at a terminal, performance information. INDICATE shows the use of the major resources (processor and storage) and contention of these resources. This includes attached processor (AP) and multiprocessor (MP) usage when operating an AP or MP system. If, after using the INDICATE command, you want more or less data, use the MONITOR command.

Specify automatic data collection with the SYSMON macro in DMKSYS. Coding considerations are in Chapter 9, "Preparing the CP System Control File (DMKSYS)" on page 327. See the *VM/SP Administration* book for the following:

- Directions on using the MONITOR command to collect performance data on a dedicated tape drive or spool file
- Format and contents of the various classes of data collection available with the MONITOR command
- Details of the INDICATE command options.

Note: The VM Real Time Monitor (SMART), program number 5796-PNA, is another program that helps you monitor and tune your system.

Improving Performance

You can improve the performance of a virtual machine by assigning it one or more performance options. These include the following:

- Favored execution
- Priority
- Reserved page frames
- Locked pages
- Virtual = real
- Queue drop elimination.

A System Product Editor SET FULLREAD ON has performance implications. For local channel attached display controllers with many users taking advantage of the SET FULLREAD usability improvement, there is a potential for a significant increase in system response time. For remote attached display controllers, setting FULLREAD ON will result in a noticeable increase in response time. Additionally, due to increased line traffic, the maximum number of terminals that can be supported on a link may be significantly reduced.

An alternative to using FULLREAD is the 3274 Entry Assist option. See the *VM/SP Terminal Reference* for more information. You can improve the performance of VM/SP running virtual storage operating systems if you use virtual machine assist or Extended Control-Program Support. The following chart shows how these and MVS/System Extensions Support are supported by various VM/SP processors.

Virtual Machine Assist			MVS/System Extensions and MVS/SP Support		Extended Control-Program Support (Note 3)	
Standard	RPQ	Not Available	System/370 Extended Facility	System/370 Extended Feature	Standard (Note 2)	Not Available
135	168	155	3031UP	158(Note 1)	135-3	135
135-3	168-3	155 II	3031AP	158-3(Note 1)	138	145
138	168AP	165	3032	158AP(Note 1)	145-3	155
145	168MP	165-3	3033UP	158MP(Note 1)	148	155 II
145-3	3032		3033AP	168	3031UP	158
148	3033UP		3033MP	168-3	3031AP	158-3
158(Note 1)	3033AP		3081	168AP	4321	158AP
158-3(Note 1)	3033MP		9375"60	168MP	4331(Note 4)	158MP
158AP(Note 1)					4331-2(Note 4)	165
158MP(Note 1)					4341	165-3
3031UP			ECPS:MVS(Note5)		4361	168
3031AP					4381	168-3
4321			4341		9370	168MP
4331(Note 4)			4361			3032
4331-2(Note 4)			4381			3033UP
4341			9377			3033AP
4361						3033MP
4381						3081
3081-D16						
9370						

1 Virtual machine assist and the System/370 Extended Feature are mutually exclusive on a Model 158 processor except for Model 3 with RPQ No. MK3272 installed. However, in a Model 158 attached processor complex, virtual machine assist can be installed on one processor while the System/370 Extended Feature is installed on the other, or both may be installed on the attached processor (not the I/O processor).
2 Users running VM/SP on a 135-3, 138, 145-3, or 148 with ECPS:VM/370 may not realize the full benefit of ECPS:VM/370 because shadow table maintenance algorithms may be used in preference to some ECPS:VM/370 algorithms.
3 Compatibility must be established when using the functions contained in VM/SP on systems with ECPS:VM/370. To establish compatibility, make sure that the service level is compatible with the latest functional update to the hardware.
If compatibility is not established, an error message is issued and ECPS:VM/370 is not established, an error message is issued and ECPS:VM/370 is nullified
4 No charge special feature if ordered with the processor.
5 ECPS:MVS and ECPS:VM/370 are mutually exclusive on the 4341 processors.
On the 4341-2 and 4341-12 processors, ECPS:MVS and ECPS:VM/370 may be used concurrently if the ECPS Expansion Feature is installed.

Additional planning is needed to support the virtual=real option and virtual machine assist, as well as ECPS:VM/370. The *VM/SP Administration* book describes all of these performance options in detail.

Specifying a Virtual = Real Machine

Although the virtual=real option eliminates paging for the affected virtual machine, its main function is to bypass 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 reflect the real storage addresses.

The only exception is virtual page 0. Virtual page 0 does not exist as real page 0. It is relocated to the first real page after the virtual=real area. Because of the relocation of page 0, CCW translation must remain on if the virtual machine performs I/O to page 0.

When CP loads an operating system into a virtual=real area, it turns on CCW translation. Once CP has loaded the operating system, the operator of the virtual machine may issue a CP command to turn CCW translation off.

When the virtual machine is operating with CCW translation off, it must not perform I/O into virtual page 0. Violation of this restriction may cause damage to VM/SP. You can generate most operating systems so they do not use virtual page 0 for I/O.

You specify the size of the virtual=real area when you generate CP. It must be large enough to contain the entire address space of the largest virtual machine that you run in the virtual=real area. Note that you can define only one virtual=real area and only one virtual machine at a time can occupy the virtual=real area.

Because the virtual=real option removes pages from the dynamic paging area, it affects the performance of the other virtual machines. The virtual=real area is set up at VM/SP initial program load (IPL). The primary system operator can release it to be used as part of the dynamic paging area. Once released, it cannot be reclaimed except by reloading VM/SP. The operator must release the virtual=real area in total. That is, unused pages of the area cannot be selected for release.

Each virtual machine logged on by CP requires some of CP's fixed free storage. If a very large virtual=real area is released after VM/SP initialization, system performance may degrade as more and more users logon and use the released space. This is because the number of pages allocated for CP fixed free storage during VM/SP initialization is based on real machine size minus virtual=real size. Therefore, the number of fixed free pages allocated for a system with a virtual=real area may not be enough to accommodate the larger number of users of the released space. Also, system overhead may increase as CP extends to get dynamic free storage pages.

You can solve this problem by using the FREE operand of the SYSCOR macro in the system control (DMKSYS) file when you generate VM/SP.
Chapter 9, "Preparing the CP System Control File (DMKSYS)" on page 327 describes the SYSCOR macro. The examples used in the following discussions assume that you are allowing VM/SP to determine the number of free storage pages to allocate.

To use the virtual=real option effectively on a multipoint teleprocessing system with no CCW translation (SET NOTRANS ON), you must dedicate lines to that system. You do this by way of the ATTACH command or in the VM/SP directory. Conversely, on a multipoint teleprocessing 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. If you enter the DIAL command while SET NOTRANS ON is in effect, CCW translation is done for I/O involving that line.

You cannot run programs with dynamic or self-modifying channel programs in a virtual=real area if you also use the DIAL command. Also, you cannot load (IPL) a shared system into a virtual machine running in the virtual=real area. For a virtual=real machine, you must enter the IPL command with either a device address or the name of a nonshared system.

To generate CP so that it properly supports a virtual = real area you must do the following:

- Change the CP build list (loadlist) specified in the VM/SP Product Parameter File (5664167E $PPF). For more information about the CP build lists and the VM/SP Product Parameter File, see the *VM/SP Service Guide*
- Reserve enough DASD space for the CP nucleus
- Reserve enough real storage space to contain the CP nucleus, virtual = real area, and other virtual machine requirements. Real storage space considerations are critical. If storage space requirements for the nucleus and virtual = real area exceed the size of real storage, the real IPL process on a VM/SP system supporting virtual storage preservation will result in a hardware load error.
- Specify the amount of storage you want reserved for a virtual = real area.

Chapter 7, "Creating Your VM/SP Directory" on page 215 describes the Directory program, including information about the VIRT = REAL operand of the OPTION control statement.

Note: With VM/SP you do not need extra DASD space for a virtual = real system.

Real Storage Validation

VM/SP automatically validates real storage on the 3081 processor using the 3081 hardware instruction TEST BLOCK. TEST BLOCK (TB) is an RRE format instruction (four bytes long) that has an operation code of X'B22C'. When VM/SP is loaded or initialized on a 3081 processor, VM/SP issues TB instructions to validate 4K blocks of real storage. This ensures that all page frames to be occupied by system modules are valid.

The 3081 real storage is not necessarily contiguous. One or more storage frames is used as a hardware system area to contain channel microcode, control blocks, and usage information. Because the hardware system area (HSA) is not addressable by the control program, an attempt to access the HSA causes an addressing exception. Thus, VM/SP uses TB on the 3081 processor to detect noncontiguous blocks of real storage and recognize unusable storage frames.

At VM/SP load, the loader uses the TB instruction as it relocates itself to the high-end of storage and while loading the system modules into storage. The VM/SP nucleus must reside in contiguous storage. If an unusable or nonaddressable frame is detected within the area reserved for the nucleus, the system load is stopped with a disabled wait state code X'AAAAAA'. There is one exception. *Nonaddressable frames and frames having errors encountered in the virtual = real area do not cause a disabled wait state at VM/SP load.* Instead, informational messages are sent to the system operator and the load continues. For this reason, virtual machine operating systems that run in the V = R area on a real 3081 should use the TB instruction to validate their storage. When the V = R area is unlocked, VM/SP automatically validates the area using TEST BLOCK.

When VM/SP is initialized on a 3081 processor, those modules involved in the IPL process issue TB instructions to determine the status of every frame of real storage. If a nonaddressable frame or a frame containing errors is detected within the area reserved for the nucleus (excluding the V = R area), system initialization is stopped with a disabled wait state code X'14'. Storage frames reserved for the V = R area are not validated at VM/SP initialization. V = R frames are validated only at VM/SP load time as described earlier. In both cases, VM/SP load and VM/SP initialization, nonaddressable or invalid frames encountered outside the nucleus area are identified to the system operator by a series of informational messages.

VM/SP simulates TB for any virtual machine with EC mode capability regardless of real processor type. However, VM/SP and MVS/SP are the only operating systems that may issue TB. When a virtual machine is running V=V on a 3081, or on any other processor regardless of virtual machine mode, CP simulates all requested TB instructions by setting the storage block and storage key to zero and returning a condition code zero. For a V=R virtual machine running on a real 3081, CP performs a real TEST BLOCK and reflects the result to the virtual machine. If the storage block is usable, the storage block and storage key are set to zero and condition code zero is returned. If the storage block is unusable, the storage block and storage key remain unchanged. A condition code one is returned. A protection exception is reflected to a virtual machine that attempts to issue a TB instruction to a shared page.

Virtual Storage Requirements

When generating VM/SP you have three limitations on the maximum virtual=real size you can specify; real storage, virtual storage, and the size of your nucleus.

Before you load the CP nucleus, be sure the virtual machine you are using has enough virtual storage to contain the loader and CP nucleus (including the virtual=real area).

loader + nucleus being loaded + V=R area = total storage requirement

You must have an area larger than this total storage requirement to use the loader. If your virtual machine does not have enough virtual storage, redefine storage and IPL again before continuing.

Specifying the Amount of Virtual=Real Space

If you are generating a VM/SP system to include a virtual=real machine, during the system generation procedure you respond "yes" to the system message:

```
VIRTUAL=REAL OPTION REQUIRED (YES,NO) :
```

You are then prompted to enter the size of the virtual=real machine size:

```
STORAGE SIZE OF VIRT=REAL (MINIMUM IS 32K) :
```

Usually, you would not want to specify the largest virtual=real machine possible, because that would leave few page frames available for other virtual machines.

At IPL time, the virtual=real area is locked in storage immediately following CP page 0. The system operator can issue the UNLOCK command with the VIRT=REAL option to free the virtual=real area for additional dynamic paging space for other virtual machines. The area cannot be relocked; it remains unlocked until another system IPL.

Calculate the maximum amount of virtual=real storage available on your processor as follows:

- Use Formula 1 to calculate the amount of real storage above the minimum required by CP at IPL time. If available real storage (ARS) is negative or zero, CP will not IPL.
- Use Formula 2 to calculate the maximum virtual=real size (VRS) for any real machine size. If VRS is negative or zero, a virtual=real area is not supported.

Calculating Available Real Storage (Formula 1): Calculate available real storage (ARS) by subtracting the amount of storage required by CP from the real machine size. Formula 1 is:

$$\mathrm{ARS} = \mathrm{RM} - \left[\mathrm{I} + \mathrm{T} + 12\mathrm{K} + 4\mathrm{K} \left[\frac{\mathrm{RM} - 256\mathrm{K}}{64\mathrm{K}} \right] \right]$$

where:

RM

is the real machine storage size.

I

is the storage needed to IPL CP. Refer to the load map produced when the CP nucleus is generated. The amount of storage needed to IPL CP is all of storage up to, and including, the module DMKSAV.

T

is the storage allocated for the CP internal trace table. CP allocates 4K of storage for each 256K of real storage for the CP internal trace table:

$$4\mathrm{K} \left[\frac{\mathrm{RM}}{256\mathrm{K}} \right]$$

If the calculation RM/256K results in a fraction, the result should be rounded upward to the next higher integer.

12K

is the fixed free storage allocated for the first 256K of real storage.

$$4K \left[\frac{RM - 256K}{64K} \right]$$

is the fixed free storage allocated for real storage beyond the first 256K (if there is no virtual = real area). If the calculation enclosed in brackets results in a negative value, replace it with zero.

If the same calculation results in a fractional number, disregard the fraction.

The result obtained from Formula 1 is the available real storage (ARS) for a particular real machine size. This result is needed to calculate the maximum size of a virtual = real area in Formula 2.

Calculating the Maximum Size of the Virtual = Real Area (Formula 2): Calculate the maximum size of the virtual = real area for a particular real machine size by recalculating the real storage required by CP and subtracting that value from the real machine size. When you calculate the real storage required by CP this time, you do not permanently allocate free storage for the portion of storage that is available for the virtual = real area (according to Formula 1). The result of Formula 2 is the maximum size virtual = real area (VRS) you can specify for a particular real machine size. Formula 2 is:

$$VRS = RM - \left[I + T + 12K + 4K \left[\frac{RM - 256K - ARS}{64K} \right] + 16K \right]$$

Use the same value for RM, I, and T as you used in Formula 1. ARS (available real storage) is the result calculated from Formula 1. If the calculation

$$\frac{RM - 256K - ARS}{64K}$$

results in a negative value, replace it with zero. If the same calculation results in a fractional number, disregard the fraction (see Examples 1 and 2). 16K is the storage needed at IPL time for the dynamic paging area. After VM/SP is loaded (via IPL), the size of the dynamic paging area is the number of pages from DMKCPE to DMKSAV plus 16K.

The following table shows the maximum size virtual = real area you can specify for some real machine sizes.

Real Machine Size	Maximum VIRT=REAL Size
768K	332K
1M	580K
2M	1582K

The values in this table assume the value of I is equivalent to 388K[25].

Example 1: Determine the maximum size of the virtual=real area for a real machine with 768K of storage running in a VM/SP system that requires 388K to IPL.

Formula 1

$$ARS = 768K - \left[388K + 4K \left[\frac{768K}{256K} \right] + 12K + 4K \left[\frac{768K - 256K}{64K} \right] \right]$$

$$ARS = 768K - \left[388K + 12K + 12K + 32K \right]$$

$$ARS = 768K - 444K$$

$$ARS = 324K$$

25 Because the amount of storage required to IPL VM/SP varies with the inclusion of optional features and the number of devices in DMKRIO, this figure is used in the following examples for illustrative purposes only.

Formula 2

$$\text{VRS} = 768\text{K} - \left[388\text{K} + 12\text{K} + 12\text{K} + 4\text{K} \left[\frac{768\text{K} - 256\text{K} - 324\text{K}}{64\text{K}} \right] + 16\text{K} \right]$$

$$\text{VRS} = 768\text{K} - \left[412\text{K} + 4\text{K} \left[\frac{188\text{K}}{64\text{K}} \right] + 16\text{K} \right]$$

$$\text{VRS} = 768\text{K} - \left[412\text{K} + 4\text{K} \left[2 \right] + 16\text{K} \right]$$

$$\text{VRS} = 332\text{K}$$

Note that the fraction (188/64) resulting from the

$$\frac{\text{RM} - 256\text{K} - \text{ARS}}{64\text{K}}$$

calculation in Formula 2 is rounded to the next lower integer, 2.

Example 2: Determine the maximum size virtual=real area for a real machine with 2048K of real storage. The VM/SP system requires 388K to IPL.

Formula 1

$$\text{ARS} = 2048\text{K} - \left[388\text{K} + 4\text{K} \left[\frac{2048\text{K}}{256\text{K}} \right] + 12\text{K} + 4\text{K} \left[\frac{2048\text{K} - 256\text{K}}{64\text{K}} \right] \right]$$

$$\text{ARS} = 2048\text{K} - \left[388\text{K} + 4\text{K} \left[8 \right] + 12\text{K} + 4\text{K} \left[28 \right] \right]$$

$$\text{ARS} = 2048\text{K} - \left[388\text{K} + 32\text{K} + 12\text{K} + 112\text{K} \right]$$

$$\text{ARS} = 2048\text{K} - \left[544\text{K} \right]$$

$$\text{ARS} = 1504\text{K}$$

Formula 2

$$\text{VRS} = 2048\text{K} - \left[388\text{K} + 32\text{K} + 12\text{K} + 4\text{K} \left[\frac{2048\text{K} - 256\text{K} - 1504\text{K}}{64\text{K}} \right] + 16\text{K} \right]$$

$$\text{VRS} = 2048\text{K} - \left[432\text{K} + 4\text{K} \left[\frac{288}{64} \right] + 16\text{K} \right]$$

$$\text{VRS} = 2048\text{K} - \left[464\text{K} \right]$$

$$\text{VRS} = 1584\text{K}$$

Note that the fraction (288/64) resulting from the

$$\frac{\mathrm{RM - 256K - ARS}}{\mathrm{64K}}$$

calculation in Formula 2 is rounded to the lower integer, 4.

Example 3: Determine the maximum size virtual=real area for a real machine with 400K of real storage. The VM/SP system requires 388K to IPL.

Formula 1

$$\mathrm{ARS = 400K -} \left[\mathrm{388K + 4K} \left[\frac{\mathrm{400K}}{\mathrm{256K}} \right] \mathrm{+ 12K + 4K} \left[\frac{\mathrm{400K\text{-}256K}}{\mathrm{64K}} \right] \right]$$

$$\mathrm{ARS = 400K -} \left[\mathrm{388K + 4K} \left[2 \right] \mathrm{+ 12K + 4K} \left[2 \right] \right]$$

$$\mathrm{ARS = 400K -} \left[\mathrm{416K} \right]$$

$$\mathrm{ARS = -16K}$$

Because ARS is a negative number, CP cannot IPL. Therefore, to tell you of this condition, a disabled wait state PSW with a wait state code of D is then loaded.

Calculating DASD Space: To evaluate the relationship of DASD requirements to real storage space for saved systems on DASD, use the following formulas:

(program size/4)/57	= number of 3330/3333 cylinders
(program size/4)/24	= number of 2305/3340/3344 cylinders
(program size/4)/120	= number of 3350 cylinders
(program size/4)/96	= number of 3375 cylinders
(program size/4)/150	= number of 3380 cylinders
(program size/4)	= number of FB-512 pages

Program size is the real storage size (in K bytes). K represents 1024 bytes.

Virtual Machine Assist

Virtual machine assist is both a processor feature and VM/SP programming. It improves the performance of VM/SP. Virtual storage operating systems that run in problem state under control of VM/SP use many privileged instructions and SVCs that cause interrupts that VM/SP must handle. When you use virtual machine assist, the processor intercepts and handles many of these interrupts. "Using Performance Options" earlier in this section describes how various VM/SP processors support virtual machine assist and Extended Control-Program Support (ECPS:VM/370).

VM/SP must handle certain interrupts. Therefore, virtual machine assist is not available if it:

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

Because an SVC interrupt recognizes an address stop, VM/SP must handle SVC interrupts while address stops are set. Whenever you enter the ADSTOP command, VM/SP turns off the SVC handling part of the assist feature for your virtual machine. VM/SP turns on the assist feature again after it encounters the instruction and removes the address stop.

Whenever a virtual machine issues a TRACE command with SVC, PRIV, BRANCH, INSTRUCT, or ALL operands, VM/SP turns off the virtual machine assist feature for that virtual machine. VM/SP turns on the assist feature again when tracing is completed.

If virtual machine assist is available on a processor, the operator, using the CP SET command, can turn the function off and on again for the entire VM/SP system. Also, if virtual machine assist is available to VM/SP, virtual machine operators can turn the function off and on again for their own virtual machines. When you create your VM/SP directory, you can set off the SVC-handling part of virtual machine assist for various virtual machines. Do this by specifying SVCOFF on the OPTION control statement.

VM/370 Extended Control-Program Support

VM/370 Extended Control-Program Support (ECPS:VM/370) is a hardware assist function that provides support beyond that provided by the virtual machine assist feature described above. Therefore, it reduces VM/SP's real supervisor state time needed to support virtual machines. ECPS:VM/370 provides the following functions:

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

Whenever you load VM/SP on one of the supported processors, all three hardware assist functions plus virtual machine assist become active unless the system operator turns them off.

Expanded virtual machine assist includes a more extensive emulation of the SSM, LPSW, STNSM, and STOSM privileged instructions. Additional privileged instructions are also emulated.

CP assist provides a hardware assist for the high-use portions of the following CP functions:

- Virtual machine I/O

- Storage management
- Page management
- SVC handler
- Privileged instruction handler
- Dispatcher.

The appropriate CP software routine is used if:

- CP assist is turned off
- Hardware assist does not support the specific service required
- An error condition occurs.

Virtual interval timer assist provides for hardware updating of the location 80 interval timer for each virtual machine that has virtual timer assist turned on. This timer assist provides an accurate and repeatable interval timer value for virtual machines.

CP turns off both virtual machine assist and expanded virtual machine assist if you start certain TRACE functions. Also, CP turns off virtual interval timer assist if external interrupts are traced. When the tracing function is stopped, CP restarts these hardware assist functions.

See the *VM/SP Administration* book for more details on VM/370 Extended Control-Program Support.

Queue Drop Elimination

VM/SP attempts to optimize throughput by monitoring the execution status of virtual machines. When a virtual machine becomes idle, VM/SP drops it from the active queue, invalidating the virtual machine's resident page and segment tables. In certain special cases, VM/SP finds a virtual machine idle and drops it from the queue. However, the virtual machine becomes active again sooner than expected. If this cycle of queue dropping and reactivating repeats, the overhead involved in invalidating and revalidating the virtual machine's pages may degrade the system.

CP SET QDROP lets you control this. See the *VM/SP Administration* book for more details.

MVS/System Product and MVS/System Extensions Support

VM/SP enables MVS, running in a virtual machine, to use the MVS/System Product or MVS/System Extensions program product. "Using Performance Options," earlier in this section, details VM/SP processors that have this support available.

The following conditions are necessary to use the MVS/System Product or MVS/System Extensions support on your virtual machine:

- Hardware is available on the real machine
- The operator has entered a SET S370E ON for your VM/SP system
- 370E appears on the directory OPTION statement for your virtual machine, or you have entered the SET 370E ON command for your virtual machine.

Note: The SET S370E command is invalid when running VM under VM. Therefore, if you are attempting to run MVS on VM under VM, do not set S370E on.

When this support is enabled, an operating system running in your virtual machine can use these functions of System/370 Extended Facility:

- Low address protection
- Common segment support
- Invalidate page table entry (IPTE) instruction
- Test protection (TPROT) instruction
- Virtual-machine extended-facility assist.

Alternate CP Nucleus

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

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

The system operator can also use the REIPL option of the SHUTDOWN command to IPL an alternate nucleus by specifying the real device address. For example, if the system is 140 the operator can enter the command:

```
shutdown reipl 141
```

to shutdown the system on 140 and immediately IPL the system on 141.

In addition to maintaining an alternate nucleus, you can also improve your system generation and IPL procedures by:

- Sharing the same warm start data, checkpoint data, and error recording data between two or more compatible versions of CP
- Protecting spool files from hardware failures by defining the warm start area and the checkpoint area on two different disk volumes
- Defining a length in the SYSNUC operand of the SYSRES macro instruction to prevent a nucleus area overflow condition from occurring when the system is built
- Using the same copy of the DMKSYS ASSEMBLE file for more than one nucleus under the following conditions:
  - SYSCKP, SYSERR, and SYSWRM areas are shared
  - The nucleus volumes are compatible with the specified SYSTYPE
  - The nucleus area is defined in the same location on all nucleus volumes
  - A SYSRES device address is selected that will not be used for any other disk (because CP would no longer require the target volume to have a specific label like VMSRES).

You can define an alternate nucleus and the other IPL improvements described earlier by using the options available in the SYSRES macro instruction. See Chapter 9, “Preparing the CP System Control File (DMKSYS)” on page 327 and the *VM/SP Operator's Guide* for more information.

A Sample Alternate Nucleus Configuration

This section shows how to plan, define, and use a general-purpose alternate nucleus configuration. For this example we will assume the following real and virtual devices exist:

Real devices:

140 - a 3380 disk labelled VMSRES containing the primary nucleus
141 - a 3380 disk labelled VMPK01 containing minidisks

Virtual devices defined for MAINT:

123 - MDISK definition for VMSRES
124 - MDISK definition for VMPK01

Planning for the Alternate CP Nucleus

For each nucleus, you will need:

- PERM space on a SYSOWN volume. No more than one nucleus can be installed on a single disk volume.
- PERM space on a SYSOWN volume for any checkpoint, error recording, or warm start area that is moved or required to be a unique area for this nucleus.
- A copy of DMKSYS for each nucleus (unless you are able to configure your space to take advantage of SYSVOL=*).

To plan an alternate nucleus configuration, you should:

1. Decide how much protection you want.
   - A stand-alone CP system used by MAINT to recover from failures. This level of protection is available through the VM/SP Starter System.
   - A production system that the operator can IPL after the primary system abends or exhibits a serious problem.
   - A production system that the operator can IPL after replacing a complete SYSOWN disk pack.

     This example produces an alternate nucleus image that is ready to IPL from a SYSOWN disk volume. If you want more complete protection, you should put the checkpoint and warm start areas on two separate volumes, and ensure that neither volume contains TEMP space (for spool files).
2. Find PERM space for each area that you want to replicate or move.

   Note that the nucleus, checkpoint, error recording, and warm start areas **do not** have to be adjacent when they appear on the same disk. One way to find space for these areas is to run DISKMAP to find gaps between the areas that are defined by MDISK statements in the directory. However, this information is only accurate if you have maintained directory MDISK statements to map all areas that are in use (for example, saved systems defined by DMKSNT).

   When you find space to use for these areas, remember to create MDISK statements in the directory to document the use of each area. This will make it easier for you to find unused gaps in the future using DISKMAP.

   Use the FMT utility to allocate all of these areas as PERM space (if they are not already defined as such).

   If you have to add a disk volume to your SYSOWN list, **do not** change the position of any volume containing TEMP space. This would destroy any spool file with pages on the affected volumes. New volumes should be added to the END of the SYSOWN list. Of course if you have a new system or you plan to IPL without restoring spool files you can make any changes you want in the SYSOWN list.

For this example, we need to find five cylinders on VMPK01 for the nucleus. We will assume that a gap of five cylinders was available from cylinder 1 to 5. The other areas are neither moved nor replicated for this example.

3. Define a version of DMKSYS for each nucleus.

   In general, DMKSYS should have different values for each nucleus. You may update DMKSYS ASSEMBLE directly, or use the UPDATE facilities of CMS to maintain different versions of DMKSYS.

   It is possible to use a single version of DMKSYS under certain conditions, but this practice requires you to DEFINE or ATTACH the appropriate volume to the SYSRES address each time you save the nucleus image. The system will not detect a problem if you are saving the nucleus to the wrong volume.

   For this example, the two copies of CP use the same collection of spool files. The checkpoint and warm start areas are both on the primary nucleus.

```
*
*       DMKSYS FOR THE NUCLEUS ON VMSRES
*
DMKSYS CSECT
       SYSOWN  VMSRES,                                         X
               VMPK01
       SYSRES  SYSVOL=VMSRES,                                  X
               SYSRES=(123,140),                               X
               SYSTYPE=3380,                                   X
               SYSCLR=NO,                                      X
               SYSNUC=(1,5),                                   X
               SYSWRM=(17,2,VMSRES),                           X
               SYSERR=(19,2,VMSRES),                           X
               SYSCKP=(265,1,VMSRES)
       .
       .       (SYSMON, SYSJRL, SYSCOR, SYSOPR, SYSACNT, SYSTIME,
       .       SYSFORM, SYSPCLAS, SYSID, SYSORD, SYSMIH, SYSFCN,
       .       SYSLOCS)
       .
       END
```

Figure 27. Sample DMKSYS for the Primary Nucleus. This example highlights the parameters that are significant with respect to the alternate nucleus support.

```
*
*       DMKSYS FOR THE NUCLEUS ON VMPK01
*
DMKSYS CSECT
       SYSOWN  VMSRES,                                                X
               VMPK01
       SYSRES  SYSVOL=VMPK01,                                         X
               SYSRES=(124,141),                                      X
               SYSTYPE=3380,                                          X
               SYSCLR=NO,                                             X
               SYSNUC=(1,5),                                          X
               SYSWRM=(17,2,VMSRES),                                  X
               SYSERR=(19,2,VMSRES),                                  X
               SYSCKP=(265,1,VMSRES)
       .
       .       (SYSMON, SYSJRL, SYSCOR, SYSOPR, SYSACNT, SYSTIME,
       .       SYSFORM, SYSPCLAS, SYSID, SYSORD, SYSMIH, SYSFCN,
       .       SYSLOCS)
       .
       END
```

Figure 28. Sample DMKSYS for an Alternate Nucleus. This example highlights the parameters that are significant for an alternate nucleus.

Defining the Alternate Nucleus

This discussion is based on the basic system configuration found in the VM/SP starter system package. A simple alternate nucleus configuration is developed using VMPK01 as the alternate IPL volume. Note that for this example, separate DMKSYS files are required. These can be established by using an update file, separate source files, or by modifying DMKSYS before each assembly.

To build an alternate nucleus:

1. Run DISKMAP to find space for the alternate nucleus. You should reserve the same amount of space that is needed for the primary nucleus. If you want to move SYSWRM or SYSCKP to the alternate pack, realize that these areas **do not** have to be adjacent to the nucleus area, or to each other.

   If the warm start area or checkpoint area is moved, only move one at a time, and start the system the next time using the area that has NOT moved (for example, move the checkpoint area and IPL the system with a WARM start the next time).

2. Update the directory to reserve the space under the $SYSNUC$ user ID. This is not required, but it is a convenient way to document the use of these areas.

3. Run FMT to allocate the area as PERM space (if it is not already allocated as such).

4. Update DMKSYS to build a system on the alternate volume. The following changes are needed:

   - SYSVOL=VMPK01

     Specify the alternate nucleus volume. This is a volume in the SYSOWN list.

   - SYSRES=(124,141)

If you maintain an MDISK for the alternate nucleus you should update SYSRES to point to the correct virtual device address (for example, 124).

- SYSNUC=(1,5) The second parameter (the nucleus length) is optional, but could save the contents of the area that follows in the event of a CP nucleus that is unusually large.
- SYSERR=(19,2,VMSRES)

  The volume label is added to point to the original error recording area.
- SYSCKP=(265,1,VMSRES)

  The checkpoint area is on the primary nucleus volume.
- SYSWRM=(17,2,VMSRES)

  The warm start area is on the primary nucleus volume.

After updates, the first part of DMKSYS would contain:

```
DMKSYS CSECT
       SYSOWN  VMSRES,                                          X
               VMPK01
       SYSRES  SYSVOL=VMPK01,           *changed                X
               SYSRES=(124,141),        *changed                X
               SYSTYPE=3380,                                    X
               SYSCLR=NO,                                       X
               SYSNUC=(1,5),            *need not be the same   X
               SYSWRM=(17,2,VMSRES),    *changed                X
               SYSERR=(19,2,VMSRES),    *changed                X
               SYSCKP=(265,1,VMSRES)    *changed
```

5. Use VMFASM to assemble DMKSYS. Then use VMFBLD to include DMKSYS and any other changes planned for this nucleus. For example, you might want to apply service to CP at this time and build the new level of CP on the alternate volume. Then, if you IPL the alternate nucleus (at 141) and detect an error of some kind, it is simple for the operator to IPL the primary nucleus (at 140) and restore the production system. For more information on VMFASM and VMFBLD see the *VM/SP Service Guide*.
6. IPL the load deck from the reader (or tape if you selected this option). The nucleus image is now stored on the alternate volume.

Variations

1. A single DMKSYS file can be used by specifying SYSVOL = *. The sample DMKSYS could be converted as follows:

```
DMKSYS CSECT
       SYSOWN  VMSRES,                                   X
               VMPK01
       SYSRES  SYSVOL=*,             *changed            X
               SYSRES=120,           *changed            X
               SYSTYPE=3380,                             X
               SYSCLR=NO,                                X
               SYSNUC=(1,5),                             X
               SYSWRM=(17,2,VMSRES), *changed            X
               SYSERR=(19,2,VMSRES), *changed            X
               SYSCKP=(265,1,VMSRES), *changed
```

   Note that the SYSRES address (120) does not match either volume owned by MAINT. This requires MAINT to DEFINE 123 or 124 as 120 before saving a new nucleus. The nucleus will be saved on any 3380 found at device address 120 when SYSVOL = * has been specified.

2. SYSCKP and SYSWRM can be placed on separate volumes. If neither disk contains TEMP space, then the loss of one disk would leave you with the ability to recover spool files using a WARM start or CKPT start. The system data areas could be defined like this:

```
               SYSWRM=(17,2,VMSRES),                     X
               SYSERR=(19,2,VMSRES),                     X
               SYSCKP=(265,1,VMPK01),
```

Potential Problems to Avoid

- All nucleus volumes must have unique volume labels. This support will not work if you attempt to SHUTDOWN REIPL a different volume with a label that matches the current IPL volume.
- In general, **do not** share warm start or checkpoint areas between two releases of CP. The size of the SFBLOK has been increased for nearly every release. When the SFBLOK size changes, the warm start data of one release cannot be read by the other release of the system.
- In general, **do not use** SHUTDOWN REIPL *cuu* to switch between two copies of CP that contain different CKPLIST definitions. If CKPLIST is changed, you should clear storage before you IPL the other system.

Using the Alternate Nucleus

Usually, the operator should IPL real device 140 from the system console. System operation will not be affected by the presence of an alternate nucleus on another disk. However, if a problem occurs while running the primary nucleus, the operator can:

1. Enter the SHUTDOWN command.

2. IPL device 141 (the alternate nucleus). This can be done through SHUTDOWN REIPL 141 if the two systems have compatible CKPLIST and SFBLOK definitions.
3. Notify the system programmer to get the primary nucleus rebuilt as quickly as possible.

Protecting Spool Files

In the earlier example, SYSCKP and SYSWRM were both on volume VMSRES. The loss of VMSRES would mean the loss of all spool files.

This danger can be minimized by saving checkpoint data on one disk and warm start data on another disk. If neither disk contains TEMP space it would be possible to recover all spool files by replacing the faulty disk and starting the system with the data on the other disk.

For example, we could place warm start data on VMDSK1 and checkpoint data on VMDSK2. Neither disk contains TEMP space so it is impossible for part of a spool file to reside on one of these disks. Let us assume VMDSK1 is destroyed. All spool files should be recovered by the following procedure:

1. Run FMT to format, label, and allocate a replacement for VMDSK1.
2. IPL the system using a CKPT or FORCE start (using checkpoint data on VMDSK2).

Maintaining Backup Directories and Override Files

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

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

Use the DIRECT command to maintain directory files.

If you establish backup directories and your installation overrides the IBM-defined classes, an override file must be established on each volume that contains a CP directory. This is because CP uses the override file found on the same volume as the directory. Therefore, if CP is forced to use a backup directory, the override file is loaded from the same volume as the backup directory.

Use the OVERRIDE command to maintain override files. See the *VM/SP Administration* book for information on the OVERRIDE command.

Generating Attached Processor and Multiprocessor Systems

Introduction

To generate an attached processor (AP) system during the system generation procedure you need to use a CP control file and build list (loadlist) specific for the type of system build you are doing. The default entries in the VM/SP Product Parameter File (5664167E $PPF) are for a uniprocessor (UP) system. Note that if you plan to generate an AP or MP system, you must change the SYSCOR macro in DMKSYS. See Chapter 9, "Preparing the CP System Control File (DMKSYS)" on page 327.

The CP control file and build list (loadlist) required to generate an AP system are specified in an override area called CPAP in the VM/SP product parameter file. For information about using an override area to build a system and building the CP nucleus, see the *VM/SP Service Guide*.

Use DMKSPA CNTRL and APLOAD (or AVLOAD for a system with a virtual=real area) to be used in place of DMKSP CNTRL and CPLOAD (or VRLOAD) EXECs.

DMKSPA MACLIB

The OPTIONS COPY member of DMKSPA MACLIB is identical to OPTIONS COPY in DMKSP MACLIB, except that the variable "&AP" is set to 1, causing AP support to be included in the module you are assembling. DMKSPA CNTRL uses this MACLIB to create a TXTAP rather than the usual TEXT, if the module is affected by attached processor support.

Modules Providing AP Support

Seven modules exist exclusively for attached processor (AP) and multiprocessor (MP) support. Nucleus-resident modules are DMKLOK and DMKMCT. Pageable modules are DMKAPI, DMKCLK, DMKCPO, DMKCPP, and DMKCPU. These modules have only an 'AP' text file and their names are contained only in the AP and MP loadlists (APLOAD and AVLOAD).

The modules that have TXTAP decks and are for AP support can be found using the following steps:

1. List all the TXTAP decks off the VM/SP base tape.
2. List all the TXTAP decks off the latest VM/SP PUT tape.
3. Combine the two lists to produce a complete list of all TXTAP decks (all AP versioned modules).

Use DMKSPM CNTRL and APLOAD (or AVLOAD for a system with a virtual=real area) to be used in place of DMKSP CNTRL and CPLOAD (or VRLOAD) EXECs.

DMKSPM MACLIB

The OPTIONS COPY member of DMKSPM MACLIB is identical to OPTIONS COPY in DMKMAC MACLIB except that the variable "&MP" is set to 1, causing MP support to be included in the module you are assembling. DMKSPM CNTRL uses this MACLIB to create a TXTMP rather than the usual TEXT, if the module is affected by multiprocessor support.

Modules Providing MP Support

DMKIOS and DMKIOQ are two modules used in MP support that is different from AP. DMKSPM MACLIB contains OPTIONS COPY with the "&MP" variable set on. The DMKRIO sample supplied with the VM/SP starter system contains a COPY OPTIONS statement following the CSECT. If you are using your own version of DMKRIO, be sure to include the COPY OPTIONS statement before assembly. DMKRIO must be assembled using the DMKSPM control file, which will pick up the DMKSPM MACLIB. To assemble, use the command:

```
vmfasm dmkrio dmkspm
```

Planning for SNA Console Communication Services

What SNA CCS Does

The Systems Network Architecture Console Communications Services (SNA CCS) provides a total data communication structure for transmitting information by way of a communications network. SNA communication products perform functions traditionally handled by the main processor. For example, management of communications lines, device dependent characteristics and control, and data formatting.

SNA CCS provides full VM/SP console capabilities to operators on SNA terminals. You can use SNA terminals as virtual machine consoles. The specific communication services and facilities used in exchanging information are transparent. If you are planning to use SNA CCS processing, you must consider the following topics.

Structure of the SNA Environment

Three major components contribute to SNA console support:

- SNA Console Communications Services (SNA CCS)
- Inter-User Communication Vehicle (IUCV)
- Either:
  - VTAM SNA Console Support (VSCS)
  - VTAM Communications Network Application (VCNA) program product.

IUCV and SNA CCS are part of VM/SP.

SNA virtual console support is provided through a virtual machine. The VTAM service machine (VSM) is the virtual machine that acts as an interface between SNA CCS and the SNA network.

The VTAM service machine is a machine that runs either VSCS or VCNA. VSCS and VCNA both work with the Advanced Communications Function/Virtual Telecommunications Access Method (ACF/VTAM) in doing their job.

Usually, ACF/VTAM is also in the VTAM service machine. VSCS, however, may be in a different virtual machine. ACF/VTAM manages the SNA network, while VSCS or VCNA acts as the interface to CCS.

A VCNA VTAM service machine also requires one of the following operating systems with External Interrupt Support (EIS):

- VSE/Advanced Functions (latest level)
- OS/VS1 with Basic Programming Extensions program product.

VSCS does not require a guest operating system.

NCP and PEP Sharing

CP supports version 2.1 of the Network Control Program (NCP) only. VTAM loads ACF/NCP. You must prevent CP from loading a back level of the NCP at initialization or restart. This can be done in many ways. All methods depend on your procedures.

One method is to initialize CP, but not enable lines until the VTAM service machine (VSM) has been initialized. A similar technique could be used when you wish to load the NCP before initializing the VSM. You can initialize CP without enabling lines. Load the NCP using your own CMS EXEC and then enable the lines. When the VSM is initialized, neither CP nor VTAM reloads.

Another method is to alter your system generation slightly by not specifying CPNAME= in the RDEVICE macro statement for the 370x device. This prevents automatic loading of the 370x at initialization by way of an IPL. The VTAM service machine would then load the ACF/NCP.

In all cases, the 370x must be dedicated to the VSM. Loading and reloading of the 370x is controlled by the VSM.

The Partitioned Emulation Program (PEP) is shared by CP and the VTAM service machine. The 370x must be dedicated to the VSM. PEP is loaded by VTAM. Once the load is complete, EP lines are disabled. The lines can then be enabled for CP. If the 370x is reloaded under control of the VSM, EP lines must be enabled for CP users.

Tracing for SNA Console Communications Services

SNA CCS places a trace entry in the CP trace table for each inbound and outbound work transaction. The trace entry identifies the type of IUCV transmission, the SNA user, and the important characteristics of the transaction. The transaction can be tracked throughout the system by use of the SNA CCS work element block, WEBLOK (passed to VCNA), the SNA CCS and VCNA path IDs, and the IUCV message ID. These fields can be matched with corresponding or similar fields in the IUCV trace elements in CP and VCNA trace elements in VTAM.

Normal Trace: SNA CCS creates trace table entries in the CP trace table, leaving an audit trail of its activities. This trace is started automatically. If you want to turn it off, set TRACE(9) to (0) in the LOCAL COPY control statement and reassemble the SNA modules.

Error Trace: In addition to the regular trace function described earlier, SNA CCS includes an error trace function. You can include the error trace independent of the normal trace. For error trace processing, SNA CCS places an entry in the CP trace table for logical errors and unexpected return codes from IUCV transmissions. If the WEBLOK that is passed between SNA CCS and VCNA is invalid, data in the trace element pertains to the invalid WEBLOK.

The error trace function is started automatically. If you do not want to use it, set the SNA CCS error trace bit off (X'40') in the CP trace flags, TRACFLG3, of the PSA (hex location 402).

Excluding SNA CCS Modules

If you do not want to use support provided by SNA CCS, you can eliminate the SNA CCS modules to save storage. You should delete the five SNA CCS modules (DMKVCP, DMKVCR, DMKVCT, DMKVCV, and DMKVCX) by excluding them from the loadlist. See Chapter 3, "Estimating VM/SP Storage Requirements" on page 41 for more information on excluding SNA CCS support modules. The size of the CP nucleus can be decreased further by eliminating the SNA routines in modules DMKQCN and DMKCPV. To eliminate the SNA routines in modules DMKQCN and DMKCPV, reassemble them with the LOCAL COPY control statement for SNA set off.

The format of the LOCAL COPY control statement to exclude SNA CCS processing is:

```
&SNAVCS SETB 0
```

Chapter 6. Planning for Hardware Devices

Contents of Chapter 6

Planning for the 3081 D16 Processor

What the 3081 D16 Processor Is

The 3081 D16 Processor is a powerful and versatile general-purpose computer designed for commercial, scientific, data acquisition, and data communication applications. This processor makes extensive use of highly integrated logic circuitry to achieve faster internal speeds and reliability, while significantly reducing power, space, and cooling requirements.

The 3081 D16 Processor complex uses the 3082 processor controller, a service processor that handles central communications for the processor complex. The processor controller does the following:

- Validates storage when the system is initialized
- Configures channel groups
- Detects and corrects recoverable channel errors
- Monitors power and coolant levels.

Monitoring and Service Support Facility

The Monitoring and Service Support Facility (MSSF) is another name for the processor controller. The MSSF gives the 3081 D16 Processor complex information about I/O devices and storage.

Virtual machine operating systems can use the SCPINFO command word to find out the processor and storage configuration of the complex. If you are running MVS under VM/SP in V=V mode and you enter SCPINFO, VM/SP simulates the MSSF response by giving you formatted data. If you are running MVS under VM/SP in V=R mode and enter SCPINFO, VM/SP gives you real data.

The MSSF also processes VARY PROCESSOR commands. When you enter the CP command VARY PROC ONLINE or VARY PROC OFFLINE VPHY on a 308x processor, VM/SP calls MSSF. The MSSF then brings the processor online or takes the processor offline. Then the MSSF returns a completion status code.

VM/SP uses the MSSFCALL DIAGNOSE instruction and MSSF commands to communicate with the MSSF. The *VM System Facilities for Programming* book describes the MSSFCALL DIAGNOSE instruction (function code X'80',) the MSSF command words, and completion status codes. The *VM/SP CP Diagnosis Reference* describes the response codes that VM/SP simulates for a V=V virtual machine.

You do not need to define the MSSF when you generate VM/SP. You must, however, define your *real* I/O configuration to the processor controller using the Input/Output Configuration Program (IOCP).

Input/Output Configuration Program

The Input/Output Configuration Program (IOCP) processes macro statements that describe the I/O devices attached to the 308x processor complex. The IOCP creates the Input/Output Configuration Data Set (IOCDS), which is stored in the processor controller. The IOCDS describes the following:

- Channel paths to the processor

- Control units assigned to the channel paths
- I/O devices assigned to the control units.

Note: The device addresses (channel, control unit, and device) you define in the IOCP macro statements should match the addresses you define in the Real I/O configuration file (DMKRIO).

A 3084 processor complex has two sides, A and B. Each side has four I/O configuration data sets. The data sets on side A are A0, A1, A2, and A3. On side B are B0, B1, B2, and B3. Except for the 3081 Model D processor complex, all of the other 308x processor complexes have only an A side and, therefore, four I/O configuration data sets. The 3081 Model D processor has only two configuration data sets - LVL0 and LVL1 IOCDS.

The IOCP has three versions; Stand-alone, VM, and MVS.

Stand-alone Version Input/Output Configuration Program

If you are installing a 308x processor for the first time, you may need to run the stand-alone version IOCP. The stand-alone version of IOCP defines the devices attached initially to the processor complex. Using the 3081 service support console or the system console, you can run the stand-alone version of IOCP to do the following:

- Read the input/output data set (IOCDS)
- Display and change the IOCDS
- Print reports from the IOCDS
- Write a new or updated IOCDS.

The stand-alone version of IOCP is shipped with the processor complex as a software program. It resides in a partitioned data set on the integrated processor controller file in the processor controller. You should check the starter IOCDS to see if there are enough I/O devices, control units, and channels defined and whether they match your devices. The *Input/Output Configuration Program User's Guide and Reference* lists the starter I/O Configuration Data Sets.

If there are enough entries in the starter IOCDS for you to generate VM/SP, you do not need to run the stand-alone version of IOCP. Simply power-on reset the processor controller and generate your VM/SP system. The processor controller uses the starter IOCDS defined in the level 0 IOCDS of the processor controller. After you generate VM/SP, and CMS is operating, use the VM/SP IOCP to define all your I/O devices to the processor controller.

If the starter IOCDS does not meet your needs, run the stand-alone version of IOCP.

VM Input/Output Configuration Program

The VM version runs under all releases of the following operating systems: VM/System Product (VM/SP) (5664-167), VM/System Product High Performance Option (VM/SP HPO) (5664-173), the VM/Extended Architecture for Systems Facility (VM/XA Systems Facility) (5664-169), and the Conversational Monitor System (CMS). Using the IOCP command with the needed options, you can change the IOCDS or print reports from the IOCDS. If you have an operating VM/SP system (that is, you have defined enough devices to the processor controller to generate VM/SP), you can use the IOCP command to define all your VM/SP devices.

The IOCP command writes the new input/output configuration to the IOCDS specified on the IOCP command. You can write-protect each IOCDS individually by using the IOCDMS(SYS021) frame on the system console. To test the new IOCDS you must do the following:

1. Shutdown VM/SP
2. Power-on reset the processor controller using the IOCDS level previously written
3. Start VM/SP.

MVS Input/Output Configuration Program

The MVS version of IOCP runs as a job under control of the OS/VS2 MVS system control program with the OS/VS2 MVS/System Product. You use JCL statements to run IOCP. By coding options on the PARM=parameter of the EXEC statement, you can create a new IOCDS or print reports from the IOCDS in storage.

You can run the MVS version of IOCP in a virtual machine running under VM/SP.

Reference

See Chapter 8, "Preparing the Real I/O Configuration File (DMKRIO)" on page 279 for more information about the IOCP source file and matching the Real I/O configuration file (DMKRIO).

For details about IOCP macro statements, the CMS IOCP command, and the starter IOCDS, see the *Input/Output Configuration Program User's Guide and Reference.*

Planning for 3270s

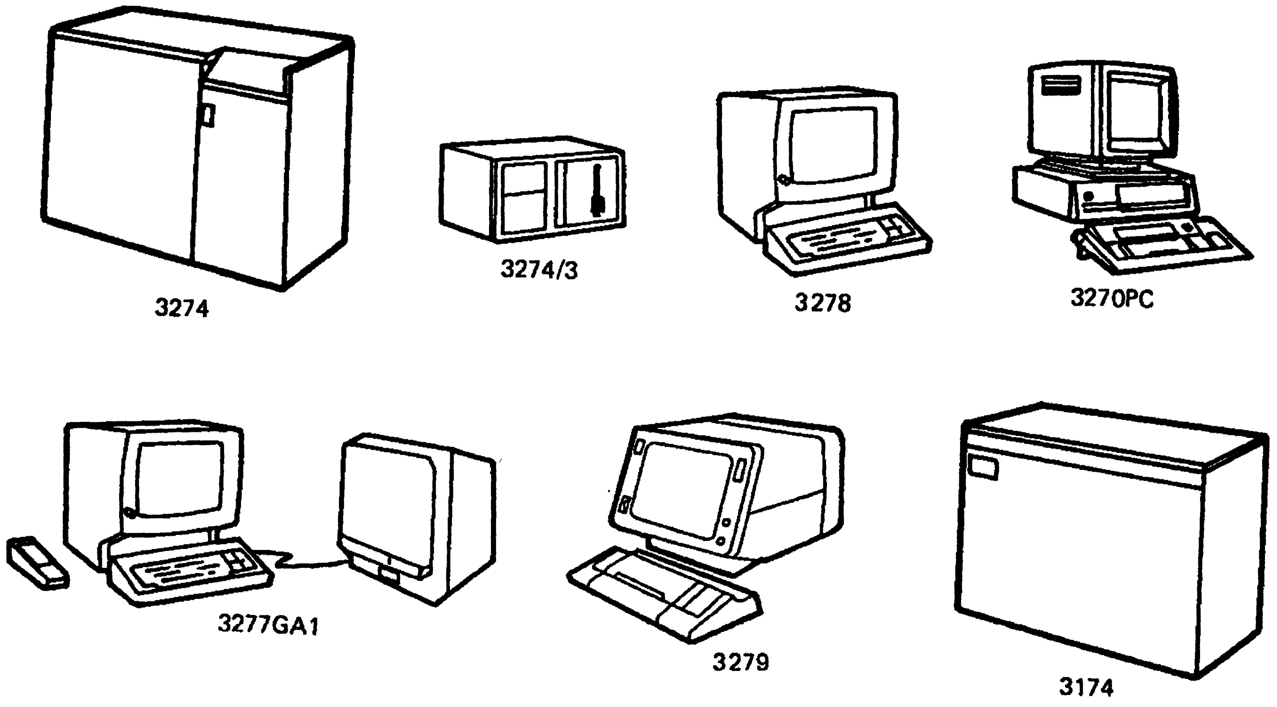


Figure 29. 3270 Hardware Devices

The VM/SP 3270 attachments can be either local or remote. You do not need telecommunication lines for local attachments. Many devices that you attach locally, can also be attached remotely. Remote attachments are attached to binary synchronous lines. For remote attachments, you usually need the following:

- A channel
- A communication controller or transmission control unit
- The device or control unit (for terminal attachment and/or RJE systems).

"Planning for SNA Console Communication Services" on page 182 talks about planning for VTAM supported terminals attached to a VTAM service machine.

Remote Attachments

VM/SP supports the 3270 in remote cluster and stand-alone configurations. This support includes:

- Nonswitched point-to-point binary synchronous transmission
- Switched binary synchronous transmission for 3275 terminals equipped with the dial feature only
- Cluster configurations of up to 32 display stations and/or printers
- The local 3270 copy function
- EBCDIC (Extended Binary Coded Decimal Interchange Code) transmission code only
- 3270s supported as virtual machine operator consoles

- CP commands allowing the operator to start and stop the teleprocessing lines, display stations and printers
- CMS Editor and System Product Editor (XEDIT)
- The recording of the Miscellaneous Data Recorder (MDR) records and Outboard Recording (OBR) records on the VM/SP error recording cylinder. The MDR records are for the station and the OBR records are for the line. The CPEREP program edits and prints these records.

The 3270 copy support lets you assign a screen copy function to a 3270 program function key. Pressing that key transfers all of the current display image to a printer attached to the same control unit. If the printer is busy or otherwise not available when you press the key, you receive a NOT ACCEPTED message on the screen.

VM/SP supports remote 3270s with the following restrictions:

- You cannot use remote 3270 terminals as primary or alternate VM/SP system consoles.
- The number of binary synchronous lines you can use for 3270s is 256 minus the number of 3704/3705/3725 communication controllers.
- You cannot use connections with multipoint clusters.

3270 Support on Binary Synchronous Lines

Display devices on binary synchronous lines are as flexible and useful as locally attached 3270 devices, except for these limitations:

- *Display Information Inquiry and Retrieval Speed* -- 3270 remote stations are subject to slow teleprocessing transmission speeds. Therefore, polling, screen display, and data entry are not as rapid for remote 3270s as they are for locally attached 3270s.
- *Hard Copy of 3270 Screen Image* -- Users of locally attached 3270s can spool their virtual console input and output to the system printer. Users of remote 3270s can also do this. But remote 3270 users and local 3270 users whose terminals are far from the printer can use a limited VM/SP hard-copy function. The RSCS Networking program product lets users print spooled output.
- *TEST REQUEST and SYSTEM REQUEST Keys* -- VM/SP does not support these keys on the 3270 terminal. The TEST REQUEST key supports the Test Request function on locally attached 3277s. The SYSTEM REQUEST key supports the Test Request function on locally attached 3278s.

  Note: If you press the SYSTEM REQUEST key on a remote terminal, the terminal session ends.

Remote Hardware Configurations Supported

To use remote 3270s you need the following:

- A binary synchronous line
- A transmission control unit
- Terminal devices (display stations and/or printers) and associated control units.

The binary synchronous line must be in 2701/2703 mode. "Configurations Supported by CMS" on page 15 describes transmission control units supporting remote 3270s on binary synchronous lines and cluster and stand-alone control units supporting remote 3270s.

Figure 30 shows a typical 3270 configuration.

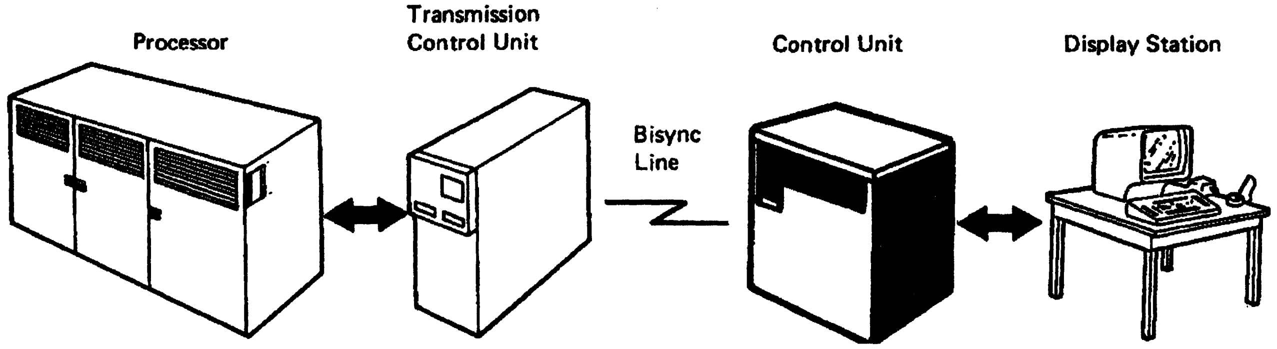


Figure 30. Remote 3270 Configuration

System Generation Requirements for Remotely Attached Display Systems

For CP to use 3270s as virtual machine consoles, you must code CLUSTER, TERMINAL, and RDEVICE macros. Then assemble them as part of the Real I/O configuration file (DMKRIO). You should do this when you generate VM/SP. After DMKRIO assembles successfully, you should make a list of resource identification codes of all the remote 3270 lines and terminals. Give the list to your operators. They need this information when they enter CP commands that control operation of remote 3270 lines and devices.

The RDEVICE macros must be in the same order as the CLUSTER macros. (See the DMKRIO example of a remote 3270 configuration on page 193.)

Chapter 8, "Preparing the Real I/O Configuration File (DMKRIO)" on page 279 describes the CLUSTER, TERMINAL, and RDEVICE macros.

Resource Identification Code

The resource identification code is a four-digit hexadecimal code. The low-order two digits of the resource identification code is the resource address and the high-order two digits is the line code. VM/SP generates the resource address. The order in which TERMINAL macros appear in DMKRIO determines the resource addresses of the terminals defined. Each CLUSTER macro defines a 3270 control unit with a resource address of X'00' through X'FF'. The device defined by the first TERMINAL macro after the CLUSTER macro (in DMKRIO) has a resource address of X'01', the second has a resource address of X'02', up to the maximum of X'20' (because X'20' represents the decimal maximum of 32 devices on one control unit).

VM/SP also generates the line code. See the assembly listing for DMKRIO to determine the line code. Find the label DMKRIORN near the end of the DMKRIO assembly listing. This label identifies a list of all lines used by remote 3270s and by 3704/3705/3725 communication controllers in EP mode. The high-order two digits is the line code that is assigned in the order that line addresses appear in the list. The first line address has a line code of 0 to complete its resource identification code, the second has 1, and so on up to the last line.

VM/SP supports a maximum of 256 binary synchronous lines for remote 3270s. Thus, the maximum value of the two high-order digits is F. Table 15 on page 192 shows you a sample DMKRIO assembly listing and the corresponding line codes. Note that the example shows four lines were generated. This simply means that the line codes for these four lines will be 00 to 03. If eight lines had been generated, the line codes would have been 00 to 07.

Table 15. Example of Determining Line Code for Remote 3270 Resource Identification Codes

Sample of DMKRIO Assembly Listing	Line Code (in hexadecimal)
DMKRIORN DC F'4'	
DC AL2((RDV078-DMKRIODV)/8)	
DC XL2'078'	00
DC AL2((RDV07A-DMKRIODV)/8)	
DC XL2'07A'	01
DC AL2((RDV079-DMKRIODV)/8)	
DC XL2'079'	02
DC AL2((RDV07B-DMKRIODV)/8)	
DC XL2'07B'	03

Once you determine the resource identification codes for devices in your remote 3270 configuration, generate a list for operations. The list should include the following:

- Line address
- Line code
- Resource address
- Label of plug on control unit panel
- Resource identification code
- Device type.

Note: The plug panels of the 3271 control unit and 3274 control unit Model 1C have up to 32 ports where you can attach terminals and printers. The 3276 has up to eight ports where you attach the 3276 integrated display and where you can attach up to seven additional terminals or printers.

An Example of a Remote 3270 Configuration

The example on page 193 shows the contents of DMKRIO that define the following:

- A clustered 3271 control unit with eight ports
- A stand-alone 3275 display station
- A second clustered 3271 control unit with eight ports.

Macros are coded so that the 3271 clustered control unit can support eight display devices, or six display devices and two printers. To define this, you must code two CLUSTER, 16 TERMINAL, and two RDEVICE macros defining the two separate clusters. The macros also support a 3275 stand-alone control unit with one display and one printer. To define it, you must code one CLUSTER, one TERMINAL, and one RDEVICE macro.

The DMKRIO file for this example is:

```
DMKRIO   CSECT
CLUST078 CLUSTER  CUTYPE=3271,GPOLL=407F,LINE=078,DIAL=NO
         TERMINAL TERM=3277,SELECT=6040,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C1,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C2,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C3,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C4,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C5,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3286,SELECT=60C6,MODEL=2
         TERMINAL TERM=3284,SELECT=60C7,MODEL=2
CLUST07A CLUSTER  CUTYPE=3275,GPOLL=407F,LINE=07A
         TERMINAL TERM=3275,SELECT=6040,FEATURE=OPRDR,MODEL=3
CLUST079 CLUSTER  CUTYPE=3271,GPOLL=407F,LINE=079,DIAL=NO
         TERMINAL TERM=3277,SELECT=6040,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C1,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C2,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C3,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C4,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C5,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C6,FEATURE=OPRDR,MODEL=2
         TERMINAL TERM=3277,SELECT=60C7,FEATURE=OPRDR,MODEL=2
         RDEVICE  ADDRESS=078,DEVTYPE=3705,ADAPTER=BSCA,      X
                  BASEADD=0B0,CLUSTER=CLUST078
         RDEVICE  ADDRESS=07A,DEVTYPE=3705,ADAPTER=BSCA,      X
                  BASEADD=0B0,CLUSTER=CLUST07A
         RDEVICE  ADDRESS=079,DEVTYPE=3705,ADAPTER=BSCA,      X
                  BASEADD=0B0,CLUSTER=CLUST079
         RDEVICE  ADDRESS=0B0,DEVTYPE=3705,ADAPTER=TYPE4,     X
                  MODEL=F8,CPTYPE=EP
```

In this configuration, if the 3271 cluster control unit is on line 078, there are six display devices and two printers supported. If the 3271 cluster control unit is on line 079, eight display devices and no printers are supported. You can interchange display devices among the resource addresses assigned to display devices. Likewise, you can interchange printers among the resource addresses assigned to printers. You cannot attach a printer at an address defined for a display device and you cannot attach a display device at an address defined for a printer.

Note: You do not need to code CLUSTER macros in ascending numerical order. However, you must code the RDEVICE macros in the same order as their corresponding CLUSTER macros. In the preceding example the order of the CLUSTER macros is CLUST078, CLUST07A, and then CLUST079. The RDEVICE macros are in the same order: RDEVICE 078, RDEVICE 07A, and then RDEVICE 079.

Table 16. Sample List of Resource Identification Codes for Operations

Line Address	Line Code	Resource Address	Label of Plug in Control Unit Panel	Resource Identification Code	Device Type
078	00	00	--	0000	Cluster
		01	0	0001	Display
		02	1	0002	Display
		03	2	0003	Display
		04	3	0004	Display
		05	4	0005	Display
		06	5	0006	Display
		07	6	0007	Printer
		08	7	0008	Printer
079	02	00	--	0200	Cluster
		01	0	0201	Display
		02	1	0202	Display
		03	2	0203	Display
		04	3	0204	Display
		05	4	0205	Display
		06	5	0206	Display
		07	6	0207	Display
		08	7	0208	Display
07A	01	00	--	0100	Cluster
		01	--	0101	Display
		02	--	0102	Printer

Note: Table 15 on page 192 shows the line codes to this sample.

Local Hardware Configurations Supported

"Configurations Supported by CMS" on page 15 describes the control units attached directly to the processor channels and the display stations and printers that attach directly to the control units.

System Generation Requirements for Locally Supported Display Systems

System generation requirements for locally supported terminals and control units are the same as the requirements for DASD or unit record devices. The RCHANNEL, RCTLUNIT, and RDEVICE macros handle the channel, control unit, and devices. See Chapter 8, "Preparing the Real I/O Configuration File (DMKRIO)" on page 279 for further details.

Planning for the 3704/3705/3725 Control Program

What 3704 and 3705 Communication Controllers Are

The 3704 and 3705 communication controllers are modular, programmable units that provide a high degree of versatility for tailoring to a teleprocessing system's requirements.

Both communication controllers provide attachment to processors of local and remote I/O devices. These devices communicate over various common carrier or equivalent customer owned communication facilities.

3704 and 3705 Support

The 3704/3705 communication controllers can support the following:

- Up to 352 low-speed start-stop lines
- Up to 60 medium-speed synchronous lines
- Line speeds from 45.2 to 56.0K baud
- Modem capability within the 3704/3705
- Limited-distance *hard-wire* capability
- 16 to 256K internal storage
- Remote 3275, 3276, 3277, 3278, and 3279 terminals with optional 3284, 3286, 3287, 3288, and 3289 printers (EP mode only)
- Remote 2780 terminals (EP mode only)
- Emulator Program (EP) Version 3.0.

Note:

VM/SP's support of the 3704/3705 does not include remote 3704/3705 communication controllers.

Note that the book *Introduction to the IBM 3704 and 3705 Communication Controllers* is an essential publication for generating a 3704/3705 control program under VM/SP. Additional 3704/3705 publications are listed in the Related Publications section on page 449.

3704 and 3705 Support Package

Before you can generate a 3704/3705 control program, you must have the following OS/VS Emulation Program Support Package. This is the only 3704/3705 support package that contains the CMS files required for generating and loading the 3704/3705 control program under VM/SP.

The support package is:

- IBM 3704/3705 Emulation Support and System Support Package (EP/VS SCP) for OS/VS (Order No. 5744-AN1). VM/SP supports this package in emulation mode only.

This package contains the following basic material:

- A program directory
- Related 3704/3705 publications
- A magnetic tape containing the macros and modules of the 3704/3705 control program and the OS/VS system support programs.

What a 3725 Communication Controller Is

The 3725 communication controller is a modular, programmable, communication controller. It runs under control of the Advanced Communications Function for Network Control Program (ACF/NCP) or Emulation Program (EP/3725). ACF/NCP communicates with an SNA access method located in one or more host processors.

The 3275 controls data communication between modem-attached or direct-attached terminal (or processor) devices, and one or more host systems. By enhancements to the storage size and internal processing speed, the 3275 accommodates large networks with high-speed lines.

VM/SP Support of the 3704, 3705, and 3725

VM/SP supports all models of 3704, 3705, and 3725 communication controllers. Three terminals are supported on start-stop lines: 1050, 2741, and CPT-TWX 33/35. The 3767 terminal (operating as a 2741) is supported by lines in EP mode. The 3101 and 3232 display terminals are supported as CPT-TWX 33/35.

The minimum internal storage required by an EP control program is 16K.

When planning for the installation of the 3704, 3705, and 3725 communication controllers, be sure that you are familiar with device characteristics, have the appropriate publications and support package, and have a VM/SP system that supports the 3704/3705/3725 control program.

CP Support of 3705 and 3725

The 3705 and 3725 communication controllers can be either one of the following:

- Connected to a single processor running VM
- Shared between two or more processors, one of which is running VM.

In both environments, the loading of the 3705 and 3725 can be done by various methods:

- The Control Program (CP) component of VM/SP can load an Emulation Program (EP) into a 3705

  OR

- A virtual machine can load any of the following into a 3705 or a 3725:

  Emulation Program (EP)
  Network Control Program (NCP)
  Partitioned Emulation Program (PEP).

Note: A virtual machine can be a guest operating system (for example, VSE, VS1, or MVS) or CMS with the Advanced Communication Functions/System Support Programs (ACF/SSP) program product.

These loading methods are described separately under the following sections:

Single 3705 or 3725 and one processor

Single 3705 or 3725 and two processors.

Single 3705 or 3725 and One Processor: In this situation, a single 3705 or 3725 is connected to a channel on a single processor running VM. The 3705 or 3725 may be loaded with an EP, NCP, or PEP. (See Figure 31.)

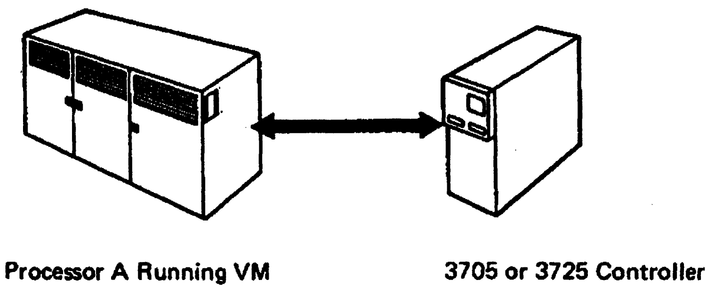


Figure 31. 3705 and 3725 Connected to a Single Processor Running VM

CP Loads the EP

In this situation, VM owns the 3705 and loads the EP into it. The lines on the 3705 may be used by VM as remote consoles or may be attached to a virtual machine, for example, to a VSE or VS1 running CICS.

When VM goes down, the 3705 may have to be reloaded, all lines must be re-enabled and, if necessary, reattached.

Virtual Machine Loads the EP

In this situation, a virtual machine owns the 3705 or 3725 and loads an EP into it. The virtual machine may own *all* the lines or the virtual machine operator may vary off one or more of the lines. The CP operator can enable these lines and they can be used for remote VM consoles.

When CP goes down, the virtual machine also goes down and must be restarted. At this time, the 3705 should be reloaded and the appropriate attaches and enables done.

A Note about Twin-tailed 3705 or 3725...

It is possible to have a 3705 or a 3725 attached to a single processor on *two* channels simultaneously. This is known as *twin-tailing*. CP can load an EP into the 3705 or a virtual machine can load an EP into a 3705 or 3725 that recognizes this configuration. The EP is loaded once over one of the channel addresses and each line could have two addresses. CP would only use a single address; that is, there is no alternate path support. Distributing lines between virtual machines and CP would be done by varying off lines in the virtual machine and enabling them in CP if the EP was loaded by the virtual machine or attaching them to the virtual machine if CP loads the EP.

Virtual Machine Loads An NCP

In this situation, a virtual machine loads an NCP into a 3705 or 3725. This is done using one of the VTAM programs (NCP Load or SSP). CP itself does not own any of the communication lines.

If you want to use some of the terminals as VM consoles, VTAM Communications Network Application (VCNA) program product must be installed in the virtual machine (VSE or VS1 only) or VM/VTAM Version 3 must be installed under VM/SP Release 4 or later.

Virtual Machine Loads PEP

In this situation, a virtual machine loads a PEP into the 3705 or 3725. In a PEP, some lines are controlled by an NCP and the rest are controlled by an EP. The lines controlled by the EP can be varied off by the virtual machine operator and enabled by the CP operator. These lines can then be used for remote VM consoles.

It is also possible to generate a twin-tailed 3705 or 3725 in this environment. The considerations are the same as stated earlier: it is loaded over one channel address and, while the lines may each have two addresses, CP will only use one of the addresses for any given line.

Note that if CP goes down, the virtual machine also goes down and has to be restarted. The PEP is then reloaded by the virtual machine and the steps taken to move lines from the virtual machine to CP have to be repeated.

There is a significant difference between this configuration and one in which the twin-tails go to different processors: VM is the sole owner of the 3705 or 3725 and can control failures and restarts.

Single 3705 or 3725 and Two processors: In this situation, a twin-tailed 3705 or 3725 is connected to a channel on processor A and to another channel on processor B. (It is assumed processor A is running VM and processor B is running VSE, VS1, MVS, or VM.) The 3705 or 3725 may be loaded with an EP, NCP, or PEP. (See Figure 32.)

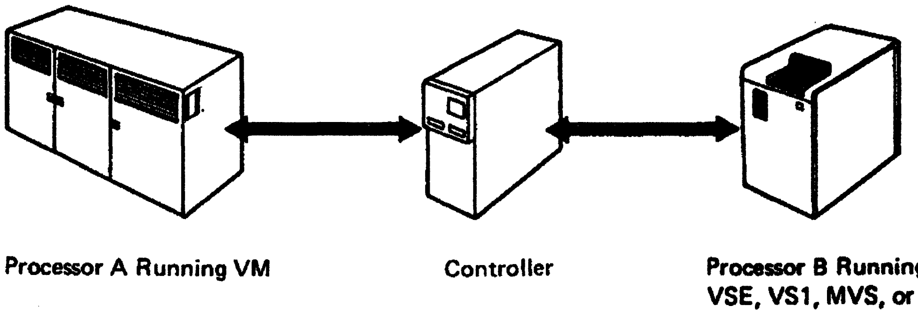


Figure 32. 3705 and 3725 Shared Between Two or More Processors (Running VM)

Be aware that...

- Effective management of both processors is essential to optimal availability when twin-tailing a 3705 or 3725.
- Events that occur in processor A can effect processor B. For example, when processor A does a system reset, processor B could be adversely affected if processor A had loaded the 3705 by way of CP.
- Any reference to twin-tailing can also be applied to multitailing a 3705 or 3725.

CP Loads an EP in a Twin-tailed Environment

The CPNAME operand in the RDEVICE macro should never be coded when twin-tailing a 3705.

The same line cannot be shared between processors because of unpredictable results. Line addresses must be mutually exclusive between processors.

If processor A and processor B are both active to an EP, and processor A or processor B reloads the 3705 or 3725, the other active processor in the twin-tail will experience loss of I/O.

Virtual Machine Loads an EP in a Twin-tailed Environment

If the EP in the 3705 or 3725 is active when attempting a load after a system restart, respond NO to any load questions to avoid reloading the EP.

If the virtual machine that owns the 3705 or 3725 is logged off, the EP lines are still active but error recording may be impacted. If CP does not load the 3705, the virtual machine is responsible for error recording.

When sharing a 3705, the CPNAME operand in the RDEVICE macro should not be coded. This will avoid automatic reloading of the 3705 which could impact the other processor if it has outstanding I/O.

Virtual Machine Loads a NCP in a Twin-tailed Environment

A virtual machine running a guest operating system can load a 3705 or 3725 with an NCP by way of VTAM, or VM can load a 3705 or 3725 with an NCP by way of ACF/SSP. To activate NCP lines in the 3705 or 3725, regular VTAM procedures should be followed. Initially, processor A loaded the 3705 or 3725 (LOAD=YES). During a restart situation, use the default load option (LOAD=NO) for both processor A and processor B.

Virtual Machine Loads a PEP in a Twin-tailed Environment

In this situation, a virtual machine in processor A loads a PEP into the 3705 or 3725 and activates the NCP while processor B activates the EP lines. If processor B does not own the 3705 or 3725 the EP lines are always there. If processor B crashes, it just activates the EP lines (that is, no reload of the 3705 or 3725 is necessary).

If processor A crashes, the virtual machine that originally loaded the PEP into the 3705 or 3725 can simply activate the NCP lines. The EP lines in both processors are still active.

EP with VM/SP

The 3704 and 3705 communication controllers are programmable units. EP can be generated to run in 3704/3705 storage.

There exist different versions of EP. VM/SP can load EP (order number 5744-AN1) into a 3704 or 3705 communication controller. A special EP (order number 5735-XXB) was created for a 3705 so that ACF/NCP-SSP can provide loading and dumping facilities for this communication controller. The same EP (order number 5735-XXB) can be used for the 3725. VM/SP provides no loading or dumping facilities for a 3725. These facilities are provided by ACF/NCP-SSP.

EP lets existing teleprocessing systems, including VM/SP, which use the 2701, 2702, or 2703 transmission control units, the 2703 compatible communications adapter of the 4331 processor, or the Integrated Communications Adapter (ICA) of the System/370 Models 135, 135-3, and 138 to run without change on the 3704/3705.

In this book, the term *3704/3705 control program* refers to the EP control program.

The EP 3704/3705 control program under VM/SP does the following:

- Emulates 2701, 2702, and 2703 operations
- Attaches to a byte-multiplexer channel
- Supports up to 255 start-stop lines for 1050, 2741, and CPT-TWX (33/35) terminals
- Supports up to 50 medium-speed synchronous lines for 3270 and 2780 terminals
- Supports service programs and special CMS commands that let you generate the EP control program in a CMS virtual machine
- Supports the CP NETWORK command that lets you load or dump the 3704/3705 and provides for automatic dumping and reloading if a fatal error occurs.

Planning Considerations

The generation of a 3704 or 3705 communication controller control program that runs under VM/SP is usually done after VM/SP system generation is complete. However, when a 3704 or 3705 is generated, the following preparations must be made:

- An RDEVICE macro instruction for the 3704, 3705, or 3725 must be included in the Real I/O configuration file (DMKRIO)
- The 3704/3705 control programs, used by VM/SP, must be stored on a CP-owned volume in the page format currently used for saved virtual machine systems (that is, those created by the SAVESYS command). Each 3704/3705 control image saved must be defined by a NAMENCP macro instruction in the system name table (DMKSNT), and saved with the SAVENCP command. This is not true for a 3725 communication controller or for a 3705 loaded by ACF/NCP-SSP.
- Enough space to contain the 3704/3705 control program image must be allocated on the CP-owned volume specified in the NAMENCP macro instruction. (Not applicable to 3725.)

Note: The alternate console for VM/SP must not be on a telecommunication line on a real 3704/3705/3725, unless the 3704/3705/3725 is loaded by another operating system (OS/VS1, OS/VS2, or DOS/VS) before VM/SP is loaded.

The *VM/SP Installation Guide* discusses how to generate a 3704 or 3705 control program. It describes support provided with EP and tells you how to generate the 3704/3705 control program, step by step.

Coding the RDEVICE Macro: The RDEVICE macro is described in Chapter 8, "Preparing the Real I/O Configuration File (DMKRIO)" on page 279.

Creating an Entry in the System Name Table

You must create an entry in the system name table (DMKSNT) for each different 3704/3705 control program that you generate. If you can foresee generating many versions of the 3704/3705 control program, define extra entries in the system name table when you generate VM/SP. In this way, you need not regenerate the VM/SP system just to update the system name table. If you should have to regenerate the VM/SP system to add a new entry to the system name table, see the discussion about the SPGEN EXEC procedure in the *VM/SP Installation Guide*.

The NAMENCP macro is described in "Preparing the System Name Table File (DMKSNT)" on page 377.

Reserving DASD Space for the 3704/3705 Control Program Image

DASD space that contains the 3704/3705 control program image must be reserved on a CP-owned volume. The DASD space reserved should be enough to contain the number of pages specified in the SYSPGCT operand of the NAMENCP macro, plus one or more for system use.

If CPTYPE=EP, allow only one extra page.

These additional pages are used to store reference table information provided by the SAVENCP program.

See the *Related Publications* section on page 449 for more information on the 3704 and 3705 communication controllers.

Planning for the 3800 Printing Subsystem

What the 3800 Is

The 3800 is a general-purpose, high-performance, nonimpact printer subsystem that uses laser and electrophotographic technology.

The 3800 Model 1 subsystem can be channel-attached to the System/370 Model 145 and larger, 303x, 308x, 4331, 4341, 4361, and the 4381 processors. The Model 135 and 138 can attach to the 3800 Model 1 by an RPQ.

The 3800 Model 3 attaches to System/370 Models 158 and 168, and the 303x, 308x, 3090, 4341, 4361, and 4381 processors. Advanced function printing is provided with the 3800 Model 3 to expand the print functions performed. Data Streaming Channel Adapter (standard) reduces channel utilization when compared to running the 3800 Model 3 in DC Interlock (DCI) mode and allows either channel mode.

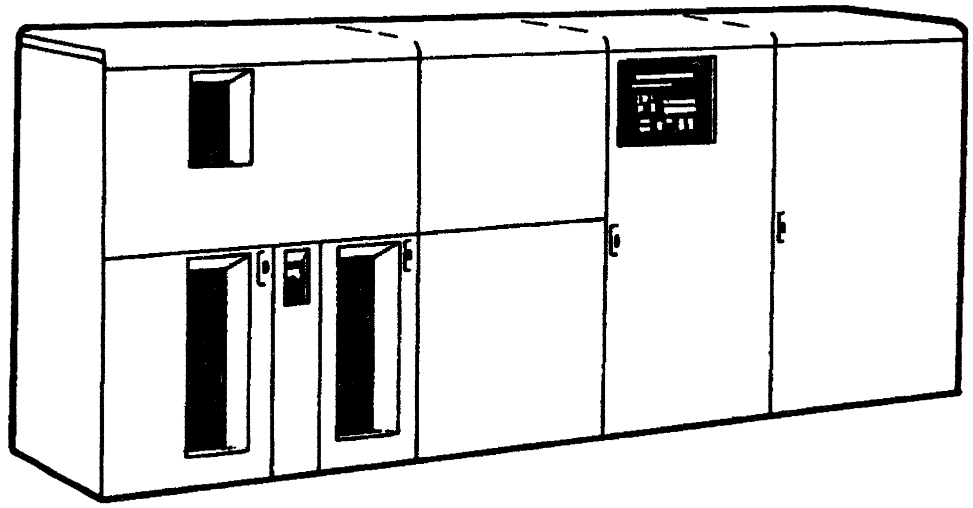


Figure 33. 3800 Printer

Planning Considerations

When planning for the 3800 subsystem, you must include an RDEVICE macro for the 3800 printer in the Real I/O configuration file (DMKRIO).
Chapter 8, "Preparing the Real I/O Configuration File (DMKRIO)" on page 279 describes the RDEVICE macro.

Allocate enough space for each 3800 image library on a CP-owned volume and create an entry in the system name table (DMKSNT) for each 3800 image library. Use the NAME3800 macro, which is described in "Preparing the System Name Table File (DMKSNT)" on page 377.

The 3800 image library contains a character arrangement table (CAT) for each character set that you want to use. The image library also contains any library character sets (LCSs) and graphic character modification modules (GRAPHMODs) referenced by these CATs. Also, the image library may contain forms control buffers (FCBs) and copy modification modules (COPYMODs). VM/SP loads these modules into the 3800 printer before printing files.

Note: 3800 Model 3 LCSs must be included in the image library. 3800 Model 1 LCSs must be included only if an odd (bit=7) CGMID is specified in the CAT.

To generate a 3800 image library:

1. Use the GENIMAGE command to create 3800 modules. The *VM/SP Administration* describes the GENIMAGE command.
2. Store the 3800 modules in the 3800 image library using the IMAGELIB and/or IMAGEMOD commands. See the *VM/SP Administration* for more information on these commands.

Store any 3800 modules that users are to have access to on a CMS disk. Users can then access this disk and embed the modules in print files with the SETPRT command.

See the *Related Publications* section on page 449 for more information on the 3800 Printing Subsystem.

Planning for the 3850 Mass Storage System

What the 3850 Mass Storage System Is

At many installations, the total volume of data collected, maintained, and/or saved for historical purposes or backup cannot fit onto the available direct access storage media. It is beneficial at many user installations to have another alternative for data storage - a *Mass Storage System.* Such a system should have:

- Capacity equivalent to a tape library
- Availability and mounting of volumes under control of the system itself (rather than by human operators)
- Data organizable in the variety of methods available on DASDs
- Cost per megabyte of storage significantly lower than DASD storage cost.

The 3850 Mass Storage System (MSS) addresses these problems. It satisfies user requirements for an economical, large-capacity storage device. This system extends the concepts of virtual storage beyond processor main storage to the I/O components of a computer system.

Generating a VM/SP System That Supports a 3850

The 3850 MSS supplies large amounts of online data under system control. Up to 472 billion bytes of data space is available, allowing you to place significant amounts of tape and DASD shelf data under direct system control. Up to four virtual machines concurrently running OS/VS1, MVS, or SVS operating systems with MSS support can each control an interface to a common 3850 MSS.

Hardware Supported

Support for the 3850 is available on the following processors supported by VM/SP:

System/370 Models 145, 145-3, 148, 155II, 158UP/AP/MP, 165II, 168UP/AP/MP, 3031UP/AP, 3032, 3033UP/AP/MP, 3033-N, 3033-S, 3042AP-2, 3081 and the 4300 processors.

Major hardware components of MSS are:

- 3851 Mass Storage Facility (MSF) (See Figure 34.)
- 3830 Model 3 Storage Control for System/370 Models 145, 145-3, 148, 155II, 158, 165II, and 168 or the Integrated Storage Control for the System/370 Models 158 and 168
- 3333 Disk Storage and Control (Models 1 or 11)
- 3330 Disk Storage Drives (Models 1, 2, or 11)
- 3350 Disk Storage Drives (Real Only).

Note: Figure 35 on page 205 shows a functional configuration of an MSS.

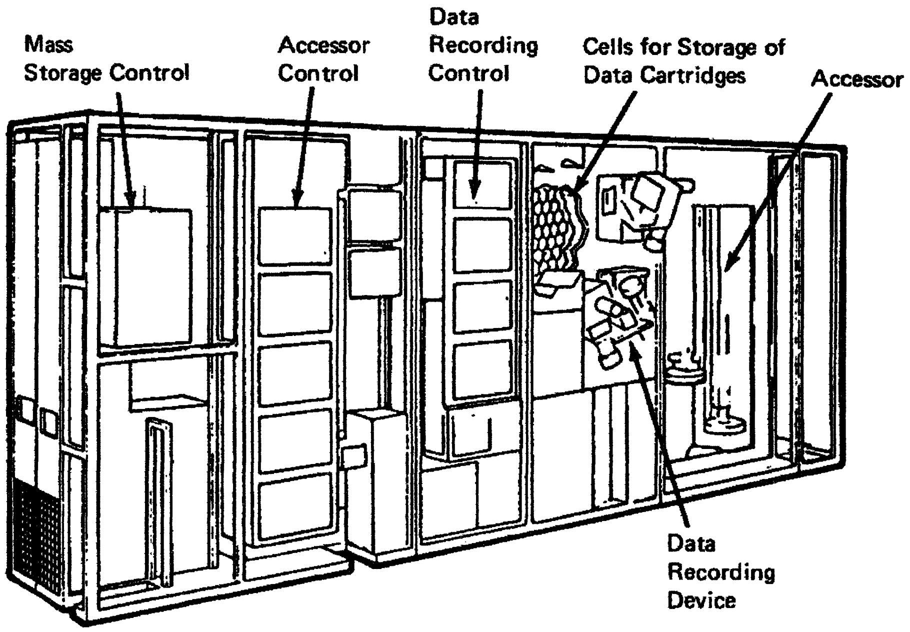


Figure 34. 3851 Mass Storage Facility Components

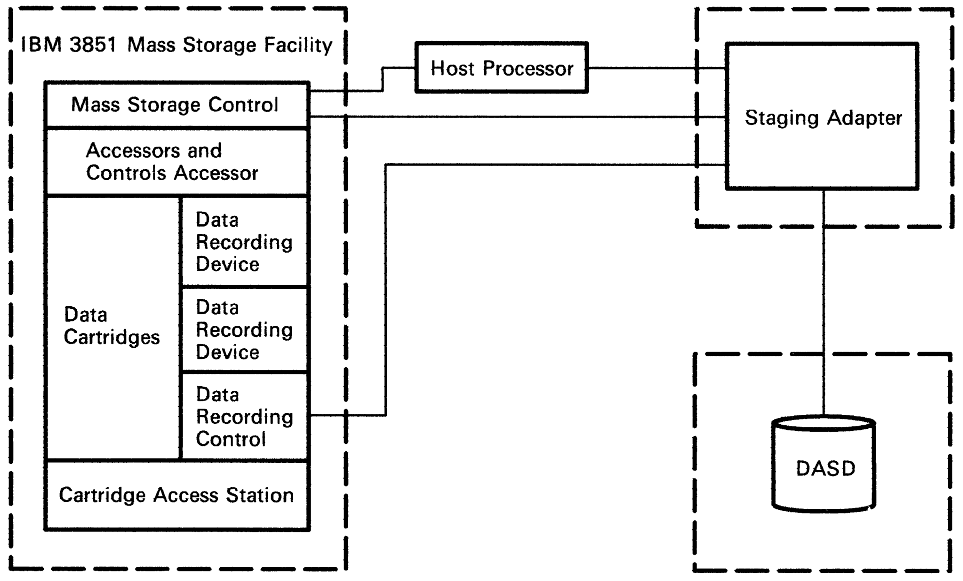


Figure 35. An MSS Configuration

Mass Storage Control

The Mass Storage Control (MSC) is a microprogrammed processor that provides operational control for components of the Mass Storage System. It is housed in the 3851 Mass Storage Facility. MSC may have four channel interface positions; A, B, C, and D. A host system attaches to one of these through a control unit position of the byte multiplexer channel or block multiplexer channel operating in burst mode. The MSC channel interface is used for transfer of orders, commands, control information, and status messages between the host system and MSC. It does not carry user application data.

Up to four operating systems containing MSS support (OS/VS1, SVS, or MVS) may be connected to MSC. These operating systems may be running in a virtual machine under VM/SP, or in a real processor, connected to the same MSC as VM/SP. One of the four MSC interfaces is dedicated to each virtual machine. Each virtual machine using an MSC port reduces by one the number of other real processors that may be connected to the MSS.

The MSS uses the 3333 control unit and the 3330 Model 1, 2, or 11 for staging data and for holding tables it requires for its operation. These units connect to the Mass Storage Facility and to the processor through a staging adapter. Many models of the 3330 may be mixed on the staging adapter. The 3330 disk drives can be one of three types; real, staging, or convertible.

Real DASD drives are not available to the MSS for any activity. They are part of the system because they have a data and control path through a staging adapter, but real drives are not logically connected to the MSS.

Staging drives are used to hold data staged from mass storage volumes to be available for processing. Staging packs are divided into pages of storage. Each page consists of 8 cylinders. The term virtual volume refers to pages of space and the data staged to that space. Each virtual volume is assigned a virtual unit address. Staging drives are logically divided into staging drive groups to assist in the management of online space. Each staging drive must belong to one and only one staging drive group. There can be no more than two staging drive groups for each Staging Adapter. Each staging drive group can have a maximum of eight logical staging drives; a logical drive being the equivalent of one 3330 Model 1. One 3330 Model 11 counts as two logical staging drives.

Convertible drives can be real or staging drives, but not both at the same time. If the drive is to be real, the real path between the drive and the operating system must be available. When the drive is a staging drive, this real path must be offline.

Note: Information describing MSS hardware can be found in *Introduction to the IBM 3850 Mass Storage System (MSS).*

On a 3850 Mass Storage System the Mass Storage Control can contain at most four channel interfaces to a single processor. The 3830 Model 3 Staging Adapter can have a maximum of four channel interfaces. The first channel interface on the 3830 Model 3 must be attached to a lower control unit position of the 3851 MSC. This control unit position does not conflict with the previously mentioned MSC port addresses. The remaining three channel interfaces of the 3830 may be attached to one or more host systems. Only the channels attached to the system being generated should be defined as primary or alternate channels.

For each of the three remaining (available) channel interface positions of a staging adapter, there are 64 possible device addresses. Thus, for each 3830 Model 3 control unit, or Integrated Storage Control with the staging adapter feature, there are 192 possible device addresses. Each device address corresponds to pages of staging space on the staging DASD. The staging space, which represents a volume, is allocated by MSC. Transfer of data between the staging space and the Mass Storage Facility, is also under control of MSC, which maintains the logical connection between a device address known to the host processor, the staging space allocated to the device, and the MSS volume mounted on the device.

When an MSS is connected to a VM/SP system, the addresses known to VM/SP are the MSC's channel interfaces and the device addresses to the channel interface positions on the Staging Adapter. MSC is supported in VM/SP only as a dedicated device. For a virtual machine to access MSC, at least one of the MSC channel interfaces must be dedicated to the virtual machine.

In this publication, the device addresses corresponding to the channel interface positions on the staging adapter are known as 3330V device addresses. There are sixty four 3330V devices per channel interface position, or 192 3330Vs per staging adapter. There may be volumes mounted on all of these devices concurrently. These 3330V volumes represent 3330-1 volumes. With the proper programming support, they may be used for all purposes that a 3330-1 volume is used except VM/SP system residence, paging, and spooling.

3330V devices may be used in three ways in VM/SP:

- Mounted and used as VM/SP system volumes (excluding system residence, paging and spooling) under the control of CP
- Dedicated to a virtual machine as a 3330-1 and accessed from the virtual machine using standard 3330-1 support
- Dedicated to a virtual machine as a 3330V, in which case the virtual machine must contain MSS support.

A 3330V device address is not manually available to the VM/SP system operator. Instead, it is an accumulation of pages of staging space on MSS staging DASD.

Volumes are mounted on, and demounted from, 3330V devices only through orders passed to MSC. MSC is supported as a dedicated device under VM/SP, and full MSC support is in OS/VS1 and MVS. Therefore, to mount and demount 3330V volumes for VM/SP use, CP communicates with an OS/VS system to which an MSC channel interface is dedicated.

Any programming in a virtual machine that accesses a real 3330-1 can access a 3330V without modification. CMS users may access CMS minidisks on MSS volumes. One MSS 3330V volume may contain minidisks for one or many CMS users. At the same time, virtual volumes may also be used as system residence packs for a VS system, and the VS system can be IPLed from the virtual volume.

The mounting and demounting of 3330V volumes used as VM/SP system volumes is done by CP communicating with an OS/VS system in a virtual machine. There is an MSS communication program named DMKMSS that is part of the VM/SP system, but runs in problem program state in an OS/VS1 or MVS system. This DMKMSS program is the interface between CP and MSC support in OS/VS.

Obtaining the MSS Communicator Program

When there is an MSS attached to your VM/SP system, you can use the DMKMSS program to communicate between the VM/SP control program and the MSC. This enables VM/SP to dynamically mount and demount MSS volumes. In this case, you should obtain the file that will install the DMKMSS program in a VS system. The required file is distributed with the VM/SP control program object code, which resides on user ID MAINT's 194 minidisk. The first step is to ensure that MAINT has access to its 194. Logon the MAINT virtual machine and enter the CMS command:

```
access 194 d/a
```

For a cardless system, before punching any files, spool the virtual punch to yourself:

```
sp pu *
```

Next, punch the file; this will install DMKMSS in your VS system. If your VS system is OS/VS1, enter the command:

```
punch mssvs1 jcl
```

If your VS system is OS/VS2, enter the command:

```
punch mssvs2 jcl
```

The punched output you receive is a series of OS/VS jobs. This file must be saved. When you run the jobs in your OS/VS system, they will install the DMKMSS program and create a VS operator procedure called DMKMSS. This procedure is later used to start the program in the communicator virtual machine.

OS/VS1 Jobs: There are four OS/VS1 jobs. They are:

- LINKDMK - This job linkedits the object code for DMKMSS into the SYS1.LINKLIB data set; the load module name is DMKMSS. The DMKMSS program must be located in SYS1.LINKLIB; this is one of the requirements of APF (Authorized Program Facility).
- DUMPT - This job prints two lists (named IEFSD161 and IEF161SD) in the system program properties table. These lists are used in the next job.

- APFZAP - This job, as distributed with VM/SP, replaces the module IEHATLAS with DMKMSS in the program properties table; this adds DMKMSS as an authorized program and removes IEHATLAS. If you wish to retain IEHATLAS as an authorized program, examine the lists produced in job DUMPT above. Change the control statement provided in APFZAP to add DMKMSS rather than replace IEHATLAS.
- LINKPROC - This job adds the procedure DMKMSS to the SYS1.PROCLIB data set. You must place the communicator device address on the COMM control statement before running this job. After the job has completed, the OS/VS1 system operator may start the DMKMSS program by issuing the command 'START DMKMSS.P*' where * is the number of the partition in which DMKMSS is to run.

OS/VS2 Jobs: There are two OS/VS2 jobs. They are:

- LINKDMK - This job linkedits the object code for DMKMSS into the SYS1.LINKLIB data set; the load module name is DMKMSS. In OS/VS2, this linkedit provides the necessary APF authorization.
- LINKPROC - This job adds the procedure DMKMSS to the SYS1.PROCLIB data set. After this job completes, the OS/VS2 system operator may start the DMKMSS program by issuing the OS/VS2 operator command 'START DMKMSS'. Before you run job LINKPROC, you must place the communicator device address on the COMM control statement.

It is not necessary to generate a VS operating system specifically for the virtual machine environment. Any OS/VS1 or MVS system that supports MSS can use VM/SP MSS support, and can act as host for the communicator program. There is, however, a requirement for MSS I/O devices in the VS system to match the definition of the virtual machine.

When OS/VS is IPLed, the system tests for any 3330Vs not online. When one is found, an order is issued to MSC for demount. The 3330V address is passed to MSC. The order tells MSC to demount any volumes currently mounted on that 3330V.

A 3330V may be offline to a virtual machine because none of VM/SP's 3330Vs were allocated to the virtual machine at that virtual address. However, the 3330V may be a valid address to MSC. If the virtual machine issues a demount order to one of these 3330V devices, a volume in use by VM/SP or another virtual machine MSC can be demounted.

The following rule must be used when defining (by way of IOGEN) 3330V devices in a VS system to run in a virtual machine to which an MSC interface is dedicated.

For each 3330V defined in the VS system there must be a corresponding 3330V defined to VM/SP and allocated to the virtual machine.

For example, if you wish to dedicate real 3330Vs 240 through 27F to virtual CPUID 22222 as virtual devices 140 through 17F, then only 3330Vs 140-17F can be defined (by way of IOGEN) in the OS/VS system running in CPUID 22222.

Defining the MSS Communication Device

CP issues an MSS mount or demount request by generating an attention interruption on a specified device. This device must be specified in the directory of the virtual machine as a unit record output device, for example:

```
SPOOL 017 2540 PUNCH
```

The same device address must be specified on the job control language used to start DMKMSS in VS, for example:

```
//MSSCOMM  DD UNIT=017
```

This device address must be constructed in VS at the same time as the IOGEN for the 3330Vs. The address chosen must not correspond to an actual device that VS will attempt to use for any other purpose. This is done by specifying the device as a DUMMY in the VS IOGEN, for example:

```
IODEVICE ADDRESS=017,UNIT=DUMMY,DEVTYPE=nnnnnnnn
```

The value of nnnnnnnn is any valid hexadecimal code. It is a VS requirement to provide a UNITNAME statement for this device, for example:

```
UNITNAME NAME=017,UNIT=017
```

Mass Storage Control Tables

This topic is for those who intend to run VS systems in a virtual machine, and access the MSS (under control of VS) from those systems. If you run only one VS virtual machine that has MSS support, and that virtual machine will access MSS only upon request from VM/SP, then this section does not apply. However, you must follow the guidelines in this topic if you have a virtual machine that has 3330Vs dedicated to it (that is, you plan to run more than one MSS virtual machine or to run VS MSS jobs in the MSS communication virtual machine).

MSC is controlled by tables that reside on DASD. These tables are used, among other things, to define the MSS configuration. This configuration includes such items as addresses to be used for all components of the system, and available paths from all connected hosts to all these component devices. The MSC tables define the allowable paths from any host (as defined by that host's CPUID) to a 3330V where the 3330V is defined in terms of the staging adapter address and the specific processor channel attachment to the staging adapter.

When a virtual machine is given access to MSS, one interface to MSC is dedicated to that virtual machine. To MSC, this is the same as having that interface connected to a native processor. The MSC tables must be constructed so that MSC can process requests from the virtual machine. MSC must treat the requests as if they came from a native processor, controlling other components of MSS such that MSS activity, as seen by VM/SP and the virtual machines, occurs on the correct 3330V device address.

Consider the example of a virtual machine that is given a virtual CPUID of 12345. This processor also has one of the MSC upper interfaces dedicated to it. Suppose that VM/SP's 3330V 250 is dedicated to the virtual machine as virtual device address 150. When virtual CPUID 12345 issues an order to MSC, the 3330V placed in the order will be 150. When interruptions are generated for this 3330V they will be sent from the Staging Adapter on the interface that corresponds to virtual CPUID 12345's 150. Because that device is known by VM/SP as 250, the MSC tables must have been constructed such that the definition of 3330V 150 for virtual CPUID 12345 corresponds to the physical connection known to VM/SP as 250.

Each 3330V in the MSC tables must map to a specific channel attachment on a specific Staging Adapter. In this case, the MSC table was constructed so that the definition for 3330V 150 on virtual CPUID 12345 corresponds to the physical connection from the real processor. This connection is through channel 2 to the same upper interface on the Staging Adapter. Interruptions received from the virtual machine's 150 are received on VM/SP's 250 as long as it is dedicated to the virtual machine corresponding to virtual CPUID 12345. Similarly, when the virtual machine issues an MSC order such as demount, the volume on VM/SP's 250 is the volume demounted.

Two different virtual machines, having the same virtual device addresses can run concurrently under VM/SP. If there are two virtual machines, each of which has defined a 3330V at the virtual machine's device address 150, then the MSC tables and the physical MSS configuration can be set so that each virtual machine can have a 3330V at address 150.

Example:

One configuration has a native processor with two block multiplexer channels, channel 1 and 2, and one Staging Adapter. Channel 1 is connected to the B interface of the Staging Adapter and channel 2 is connected to the C interface of the Staging Adapter. The VM/SP system has 3330Vs generated as 140 through 17F and 240 through 27F. Two virtual machines are defined as CPUID 11111 and CPUID 22222. Each of these machines can support an operating system in which the 3330Vs are generated at addresses 140 through 17F. The MSC tables for this configuration must show CPUID 11111 with its 3330Vs 140-17F mapped to the Staging Adapter interface B and CPUID 22222 with its 3330Vs 140-17F mapped to the Staging Adapter interface C.

Creating MSS Volumes

Before a pair of MSS data cartridges can be treated as a volume (see Figure 36 on page 211) or accessed as VM/SP system volumes, they must be initialized as the image of a 3330-1 disk pack. This initialization is done by an OS/VS Access Method Services command called CREATEV. CREATEV is one of many commands that are part of the MSS component of Access Method Services, which in turn is a standard component of OS/VS1 and OS/VS2. CREATEV can run either under VS on a native processor, or under VS running in a virtual machine to which an MSC port has been dedicated. In either case, once CREATEV has completed, the volume is known to the MSS and may be referenced in MSC mount and demount orders.

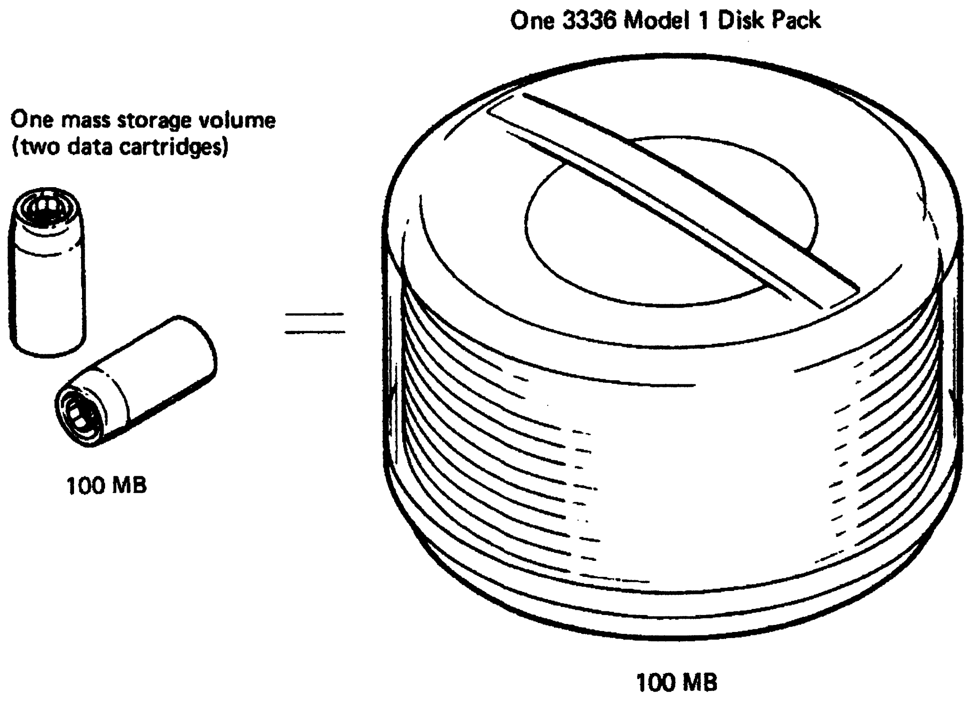

One 3336 Model 1 Disk Pack
One mass storage volume (two data cartridges)
100 MB
100 MB

Figure 36. Capacity of a Mass Storage Volume

Copying 3330-1 Volumes to 3330V Volumes

A full or partial 3330-1 volume may be copied to 3330V volumes. Once the MSS volumes have been initialized as described previously, with CREATEV, either of the following may be done:

- The access method services command CONVERTV can be issued from either a native processor or a VS virtual machine. CONVERTV will make a bit by bit copy of the 3330-1 on the MSS 3330V.
- All or part of the 3330-1 volume and the 3330V volume can be allocated to a virtual machine using the directory MDISK or DEDICATE statements, or the operator ATTACH command. Standard CMS, OS, DOS, OS/VS and stand-alone utilities can then copy data to the MSS volume.

Using 3330V Volumes for VS System Residence

A VS system can be loaded in a virtual machine from a 3330V volume because VM/SP can make the virtual IPL device appear to be a 3330-1. The following steps describe one way this can be done:

- Use the CREATEV command to create an MSS volume with a volume serial number of VOL001.
- Define a directory entry for a virtual machine (VS2VM) with an MDISK statement, describing a minidisk spanning cylinders 1 through 401 on volume VOL001.
- VM/SP mounts VOL001 and allocates the minidisk when VS2VM logs on. The operator can then attach a 3330-1 containing a VS2 system to VS2VM.
- Copy cylinders 0-400 of the 3330-1 to the minidisk within VS2VM.
- IPL the virtual device address corresponding to the minidisk as a VS2 system residence device.

See the *Related Publications* section on page 449 for more information on the 3850 Mass Storage System.

Part II. Defining Your VM/SP System

Part II describes macros and control statements that you need for defining your VM/SP system.

The following topics are discussed in Part II:

- Creating your VM/SP Directory
- Preparing the Real I/O Configuration File (DMKRIO)
- Preparing the CP System Control File (DMKSYS)
- Preparing the System Name Table File (DMKSNT)
- Additional System Definition Files.

Chapter 7. Creating Your VM/SP Directory

Contents of Chapter 7

Directory Components

The VM/SP directory contains the entries of all potential virtual machines permitted to logon the VM/SP system. Without the proper directory entry, a user cannot logon to VM/SP. Entries in the directory contain:

- User identification and password
- Virtual machine I/O configuration
- Associated virtual and real addresses
- Disk usage values
- Virtual processor storage size
- Other options.

There must be at least one CONSOLE or SPOOL directory entry for each user, except those whose password is NOLOG. These options are discussed in the directory program control statement descriptions.

The directory usually resides on the VM/SP system residence disk, and is pointed to by the VOL1 label. On a count-key-data DASD, this is at cylinder 0, track 0, record 3. On an FB-512 DASD, it is at absolute block 5. The size of the directory was formerly limited to 10,816 users. It is now limited only by the space available. The VM/SP directory program (module DMKDIR, started by the DIRECT command, or run stand-alone) processes your control statements and writes the directory on disk. You describe your real configuration when you create the real I/O configuration file (DMKRIO). You describe your many virtual configurations with directory program control statements.

To create a VM/SP directory, you must:

1. Prepare directory program control statements.
2. Sort the directory by user ID.
3. Format and allocate DASD space to contain the directory. (See "VM/SP Directory DASD Requirements" on page 50 for more information.)
4. Change restricted passwords to installation-specific passwords.
5. Run the directory program.
6. Correct any errors.
7. Rerun the directory program if necessary.

At this time, prepare the directory program control statements and sort the directory by user ID. Though not necessary, sorting the directory is recommended. Sorting the directory permits a search to go to the correct page immediately instead of reading every page until the desired user ID is found.

During system generation you must do the following:

- Format and allocate DASD space for the primary and alternate directories
- Generate the directory.

Considerations for Preparing the Directory Control Statements

First, prepare a directory control statement that defines the device on which the VM/SP directory is to be written. This statement (DIRECTORY) must be the first control statement in the input to the directory program. It is followed by the sets of statements describing your virtual machines.

Next, prepare directory program control statements describing each virtual machine. The descriptions contain accounting data, options, and virtual machine configurations for each virtual machine that appears in the VM/SP directory.

When preparing directory control statements, see Table 17 for the following information about virtual machines.

- Description
- Privilege classes
- Address of the user's primary minidisk space
- Size of the user ID directory (blksz).

The following table lists directory information that can help you plan for the products in your directory.

Table 17 (Page 1 of 3). System Directory Planning Input Information (System IDs)

System User ID	Description	Privilege Class	Minidisk Address	Size in Blocks (blksz = 512 Bytes)
AUTOLOG1	Automatically does operations every time the system is IPLed.	ABCDEG	191	192
CMSBATCH	Better utilizes the system and personnel resources.	G	195	960
CMSUSER	A virtual machine example required by the typical CMS user.	G	191	2280
EREP	Program that stores errors produced by failed hardware components.	FG	191	912
GCS	A virtual machine supervisor that lets group members share VM/SP systems.	G	191	4600
IVPM1	Directory entry for the IVP virtual machine.	G	191	256

Table 17 (Page 2 of 3). System Directory Planning Input Information (System IDs)

System User ID	Description	Privilege Class	Minidisk Address	Size in Blocks (blksz = 512 Bytes)
IVPM2	Directory entry for the IVP virtual machine.	G	191	256
MAINT	Performs system maintenance activities. MAINT owns all system files.	ABCDEFG	096 123, 124 127, 129 130 190 191 192 193 194 195 196 19D 201 293 294 295 296 319 392 393 394 395 396 59E 595 596	5,064 FULLPACK (DASD Dependent) 72,000 8,936 4,600 40,000 30,000 5,848 13,952 38,000 20,520 38,464 19,232 25,088 16,744 5,000 5,848 70,000 70,000 4,600 30,696 9,000 28,000 5,064
OLTSEP	Provides online diagnosis of I/O errors for devices attached to System/370.	FG	5FF	FULLPACK
OPERATNS	Handles IPCS extended problems and CP system dumps.	BCEG	191,193	456; 7,296
SYSDUMP1	Handles DASD volume backups and restores a user's CMS files from backup tapes.	BG	123, 124, 127 129, 130	FULLPACK DASD Dependent
TSAFVM	User ID for TSAF	G	191	1,024
AVSVM	Provides SNA LU 6.2 interface to other systems.	G	191	848

Table 17 (Page 3 of 3). System Directory Planning Input Information (System IDs)

System User ID	Description	Privilege Class	Minidisk Address	Size in Blocks (blksz = 512 Bytes)
VMSERVS	Manages the IBM-supplied file pool VMSYS.	B	191 301 302 303 304 305	2,280 1,024 2,280 2,280 2,280 32,616
VMSERVU	Manages the IBM-supplied file pool VMSYSU.	B	191 301 302 303 304 305	2,280 8,936 14,824 14,824 2,280 5,848

If you are defining your own privilege classes for CP commands, see the *VM/SP Administration* book.

VM/SP does not check for overlapping extents. Therefore, you must ensure that minidisk extents defined in the VM/SP directory do not overlap each other and (in the case of 3330, 3340, and 3350 disks) do not overlap the "alternate track" cylinders. If overlap conditions exist, file data damage is inevitable. You must define one or more virtual machines for the operator. You should define virtual machines for the system analyst or system programmer.

Note: As the directory source file is processed, VM/SP checks those statements that describe virtual devices for a subchannel protocol conflict. If a conflict is detected, CP sends an error message to the user. Only the effects of the T-DISK option of the MDISK statement and the CONSOLE, SPECIAL, and SPOOL statements are considered. The effects of the DEDICATE, LINK, and MDISK statements depends on the real device configuration at LOGON time.

The operator's virtual machines control the following:

- VM/SP sessions
- Allocation of machine resources
- Spooling activity
- Online disk areas.

Also, define virtual machines for system analysts that do the following:

- Perform system analysis
- Modify certain VM/SP functions.

and virtual machines to update or operate the following:

- CP system
- CMS system
- Hardware

- Other operating systems that run in the virtual machine environment.

System Reserved Areas

The following user IDs in the directory do not represent virtual machines. Instead, they indicate areas on the physical DASD volumes (real storage) that are reserved for system functions. You should not define user minidisks in these areas.

Table 18. User IDs Reserved for System Functions

System User ID	Description
$ALLOC$	Reserves the first cylinder (or first 16 FBA blocks) of each DASD volume for the allocation map, volume label, and other volume-related information
$TEMP$[1]	Indicates the real storage location of TEMP space
$TDISK$[1]	Indicates the real storage location of T-DISK space
$CPNUC$[2]	Indicates the real storage location of the CP nucleus
$DIRECT$[1]	Indicates the real storage location of the directory
$SAVSYS$[3]	Indicates the real storage location of saved systems
$SYSCKP$[2]	Indicates where CHECKPOINT START information can be saved
$SYSERR$[2]	Indicates where system error recovery information can be saved
$SYSWRM$[2]	Indicates where WARM START information can be saved.

Notes:

1. This should match the areas defined when you allocate the volumes.
2. This should match the entries in DMKSYS.
3. This should correspond to the area designated in DMKSNT.

Passwords That Do Not Work

For every virtual machine you create, you must define a password on the USER control statement. When the directory program is run, it checks all these passwords against a list of restricted passwords in a file called RPWLIST DATA. If any password is found in the RPWLIST DATA file, it is changed to NOLOG. The user will not be able to log on until you change the password that was found in the RPWLIST DATA file to a password that is not restricted.

Note: The stand-alone directory program does not check for restricted passwords. Also, on MDISK Control Statements, RPWLIST DATA does not affect the link passwords.

The following list of restricted passwords, which is supplied by IBM, includes sample passwords published in IBM documents:

ACNT	AUTOLOG	AUTOLOG1
BATCH	CE	CMS1
CMS2	CMS3	ECMODE
GCS	IBMCE	IPCS
ISMAINT	ITPS	IVPASS
LEV2VM	MAINT	MASTER
MDVR	OPASS	OPERATNS
OPERATOR	OSVS1	PASSWORD
PSR	ROUTER	SYSADMIN
SYSDUMP	VMAP	VSEIP
PROMAIL	SFBATCH	CMSUSER
SFCAL	CPCMS	DIRM
VSEIPO	SQLDBAPW	PRODBM
RSCS	VSEMAINT	SQLUSER

You can add your own passwords to this list, or delete passwords from it. You will probably want to restrict obvious passwords such as your company's name or any password that you think unauthorized persons may know.

To change the RPWLIST DATA file, use the System Product Editor. Enter one restricted password per record. The file must have a fixed record length. Each record must be in the following format:

Column	Contents
1 - 8	Password (only one)
9	Blank
10 - EOR	Comments

If you rename or erase the RPWLIST DATA file, passwords are not checked and a warning message is issued. However, the directory will be updated.

System Support Virtual Machines

At system generation, two additional virtual machines should be created beyond those needed by VM/SP users. These machines include one each for hardware and software support. IBM ISG programming support should be consulted when you establish these virtual machines.

Hardware Support

The hardware support is for:

- The processor, which must be supported in a dedicated environment. This is because there is no method available that allows concurrent support of the processor, real storage, and channels when running problem programs.
- I/O equipment, which can be supported using online test (OLT) under OLTSEP. The OLTSEP program can be run in its own virtual machine.

Any offline testing capabilities of system devices can be used on inactive units while the system is operating.

To perform online hardware support, a virtual machine must be defined in the VM/SP directory for the IBM service representative. The virtual machine should have enough virtual storage defined to run OLTSEP. Usually, the device being tested is dedicated to the service representative's virtual machine. The system operator can dedicate devices to a virtual machine by entering the ATTACH command.

The virtual machine for hardware support should have the minimum configuration required to run online tests, and provide access to CMS with a read/write minidisk. Privilege class F should be assigned to allow the hardware diagnostics to be run, and error recording and retrieval facilities to be used.

The hardware service representative's virtual machine should also have access to CMS and to the error recording area of the system residence volume. An EREP program (CPEREP) runs under CMS allowing editing and printing of all VM/SP recorded machine check and channel check errors. This directory entry is included in the VM/SP directory provided with the starter system.

Software Support

The virtual machine for software support should have sufficient system resources allocated to recreate problems (virtually) that occur on the real machine, and to perform software service activities. ECMODE ON must be specified for this machine.

Using Directory Profiles

A directory profile is a set of statements that defines users' virtual machines. Instead of coding the same statements over and over in many user entries, you can define one directory profile and include it in each user entry that needs the set of statements.

To define a directory profile, you must code a profile entry in the directory. A profile entry begins with a PROFILE control statement and can include any directory control statements except DIRECTORY, PROFILE, INCLUDE, USER, and MDISK. (See "PROFILE Control Statement" on page 232.)

To include a directory profile in a user's directory entry, code an INCLUDE control statement. (See "INCLUDE Control Statement" on page 237.) This has the same effect as coding all the statements in the directory profile.

Suppose that all the users in Department G63 need many of the same control statements in their directory entries. You code the following profile entry to define a directory profile called DEPTG63:

```
PROFILE DEPTG63
    OPTION REALTIMER
    ACCOUNT G6305K G63X0253
    ACIGROUP DEPTG63
    IPL CMS PARM AUTOCR
    CONSOLE 009 3215
    SPOOL   00C 2540 READER *
    SPOOL   00D 2540 PUNCH B
    SPOOL   00E 1403 6 PRINT25
    SPECIAL OFF TIMER
    LINK    ANNSMITH    190 190 RR
    LINK    ANNSMITH    19A 19A RR
    LINK    ANNSMITH    49B 49B RR
    LINK    SYSTOOLS    395 399 RR
```

When you code the directory entries for each user in Department G63, you do not have to repeat the 13 control statements in the directory profile. Each user's directory entry can be quite short, for example:

```
USER SKYE1 XXXXXXX 02048K 4M G
    INCLUDE DEPTG63
    MDISK 191 3350 430 005 MADRD3 M

USER SKYE2 XXXXXXX 02048K 4M G
    INCLUDE DEPTG63
    OPTION ECMODE
    ACCOUNT G6305K BLDG482
    MDISK 191 3350 435 005 MADRD3 M
```

By using a directory profile, you have saved time and storage space.

Notice that the OPTION and ACCOUNT statements in SKYE2's directory entry contradict statements in the directory profile. Control statements in an individual user entry override the corresponding statements in a directory profile. SKYE2 has the ECMODE option but not the REALTIMER option. When he or she logs on, a message will be issued telling him or her that the directory entry has a statement that does not agree with the profile it includes.

Security Considerations

Transparent Services Access Facility

If your installation uses the Transparent Services Access Facility (TSAF), you must ensure that all user IDs within your TSAF collection are unique. For more information, see the *VM/SP Connectivity Planning, Administration, and Operation* book.

APPC/VM VTAM Support

For security considerations when setting up AVS, see the *VM/SP Connectivity Planning, Administration, and Operation* and *VM/SP Introduction to Security* books.

Access Control Groups

If your installation has the Resource Access Control Facility (RACF)/VM 1.7.1 (Program Number 5740-XXH), you can use the ACIGROUP control statement to determine which users are allowed to:

- Log on to the system
- Link to a given minidisk
- Transfer or spool data to a given user ID
- Use the TAG command to send data.

When a user enters the LOGON or AUTOLOG command, a password verification routine checks to see if that user belongs to an authorized access control group. When a user enters the LINK, SPOOL, TRANSFER, or TAG command, an access verification routine does the same thing. If the user's directory entry does not contain an ACIGROUP statement with the right groupname, the control fails.

The LOGON, AUTOLOG, and LINK commands fail even if the unauthorized user knows the logon or minidisk password.

You must modify RACF's password and access verification routines to indicate which access control groups are authorized. See the *VM/SP Administration* book to do this.

See "ACIGROUP Control Statement (Optional)" on page 247 for more instructions on coding the ACIGROUP directory control statement.

Directory Program

You can run the VM/SP directory program (DIRECT) under CMS (using the DIRECT command) or stand-alone. The stand-alone version of the directory program is provided in object deck form (a three card loader, followed by the DMKDIR text deck), and may be loaded directly from either a real or virtual card reader.

If you run the directory program under CMS, input records must be in a CMS file with a default file ID of USER DIRECT. The file can reside on a minidisk or in a Shared File System (SFS) directory. The DIRECT command loads the directory creation module. If no file name is specified, the program looks for a file named USER DIRECT. Otherwise, it looks for a file named DIRECT.

If the file is not found, or if an error occurs during processing, the directory is not created and the old directory remains unaltered.

Normal completion writes the DASD address of the new VM/SP directory in the VOL1 label. If the active system directory is the directory being updated, it is placed in use at this time. You can print the new directory by entering the CMS command PRINT USER DIRECT (or PRINT file name DIRECT).

The virtual machine running the directory program must have write access to the volume to contain the new directory. If you create a directory that is to be written on the active VM/SP system residence volume, your virtual machine's current directory entry must have write access to the volume containing the current VM/SP directory.

Example: Assume that you have the following: virtual machine for online directory modification.

```
USER SAMPLE NOLOG 1M 2M BG
  ACCOUNT 4 SYSADMIN
  OPTION REALTIMER
    IPL CMS
    CONSOLE 009 3215
    SPOOL C 2540 READER *
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    SPECIAL OFF TIMER
    LINK MAINT 190 190 RR
    LINK MAINT 19E 19E RR
    MDISK 191 3380 200 003 VMSRES MR RDIRM WDIRM MDIRM
    MDISK 193 3380 086 009 VMSEXT MR RDIRM WDIRM MDIRM
    MDISK 195 3380 203 009 VMSRES MR RDIRM WDIRM MDIRM
    MDISK 123 3380 000 885 VMSRES MW
```

Using XEDIT you can create or modify a card-image file of the VM/SP directory input. When you are ready to write a new directory, enter the command:

```
direct filename
```

where

`filename`

is a CMS file (usually named USER) with file type DIRECT containing the necessary directory program control statements. The DIRECT command formats this file into the form of a directory, and replaces the old directory with this new one.

Loading the DMKDIR object deck by way of the card reader is the same as entering the DIRECT command in CMS, except that after IPL, the program asks you for the address of a card reader containing the directory program control statements.

Once the directory is updated, directory changes for a user currently logged on to the system do not take effect until the user logs off the system and then logs back on. Note that the LINK command can use information from the updated online directory immediately. If you add a LINK statement or an MDISK statement to the user's directory, the user can then enter the CP LINK command to get the minidisk without logging off. When a new directory is written for a new system residence volume, the new directory does not take effect until the new system residence volume is loaded (by way of IPL).

DISKMAP EXEC

The DISKMAP EXEC summarizes the minidisk assignments (MDISK statements) in a CP directory file. DISKMAP produces an output file called *fn* DISKMAP A, where *fn* is the file name of the target directory. In this file, information about the directory MDISK statements is organized by CP volume. Gaps between minidisks and overlapping minidisks are flagged.

Note: DISKMAP does **not** replace the EDIT function of the DIRECT command. You should use both to check your directory after changes.

Format

DISKMAP	*fn*	[*ft* / **DIRECT**]

Operands

fn

is the file name of the directory to be mapped.

ft

is the file type of the directory to be mapped. The default is DIRECT.

Usage Notes

1. DISKMAP creates both the map and a workfile on file mode A. If your directory is very large and file mode A is almost filled, you might need to find some additional space in order to run DISKMAP.
2. Because some DASD types come in several sizes, DISKMAP does not list gaps found after all minidisks. You will need to know the maximum cylinder/block value for your DASD type.

3. You can choose to include some overlaps in the directory. DISKMAP flags *all* overlaps; you must understand your layout to determine if a particular overlap is expected or in error. For instance, in the sample CP directory (VMUSERS DIRECT), the same full-pack minidisks (12x) are defined for backup purposes for MAINT and SYSDUMP1.

Invoking the Directory Program (DMKDIR) Under CMS

The VM/SP directory program records the configuration of each user's virtual machine in the VM/SP directory. Each virtual machine configuration includes counterparts of the components found in a real machine: a virtual operator's console, virtual storage, and virtual I/O devices and control units.

The same version of the directory service program deck can be placed in the card reader and loaded directly, or run in a virtual machine under CMS.

The CMS file named DIRECT can be updated with XEDIT to include additional directory entries.

CMS DIRECT Command

The CMS DIRECT command processes a directory file to see if it follows the required format. To actually change or swap the currently active VM/SP directory, you must have write access to the system-owned (system residence or IPL device) volume that contains the current directory up to and including the directory cylinders, or to the volume that is to contain the new directory.

If you have the above qualification and wish to verify that a CMS file can be used as a directory file, you must use the EDIT option. Otherwise, if there are no control statement errors, the file is put into active use.

To build a VM/SP directory on a CP-owned volume using preallocated cylinders, a new directory should be built so as not to overlay an existing directory. You must, therefore, allow space for two directories, or allocate a new area for the VM/SP directory each time it is created. If you run the directory program under VM/SP, the newly created directory is dynamically swapped, and placed in use by VM/SP (provided that the directory you update is currently in use by the system, or the target directory pack is present in the system owned list).

Format

DIRECT	[*filename* [*filetype* [*filemode*]]] **[(EDIT)]** **USER** **DIRECT** *****

Parameters

filename [*filetype* [*filemode*]]

is the identification of the file containing control statements for the directory program. For convenience, the file type should be DIRECT, although other file types are permitted.

- If only file name (no file type) is specified, file type defaults to DIRECT.
- If no file name and file type are specified, the program defaults to a file named USER DIRECT.
- If file mode is not specified, it defaults to *.

(EDIT)
specifies that the directory is to be examined, but not changed. The new directory will be written out to the directory cylinders, but the directory pointers will continue to point to the old directory.

Under CMS, the DIRECT command loads the directory creation module. The first statement encountered must be a DIRECTORY statement. (If not found, the program stops.) A syntax error in any statement generates an error message, and the directory is not updated. If no critical errors are encountered, the remaining statements are checked for syntax.

If the directory program abnormally ends, the old directory is not altered. Normal completion places the directory in use by VM/SP. After the new directory is created, it can be printed by entering the CMS command PRINT USER DIRECT (or PRINT file name file type).

When entering the DIRECT command with the EDIT option, the directory is written out to the directory cylinders; however, the directory pointers remain unchanged.

The DIRECT command produces the following return codes. xx is the CMS routine return code.

Table 19. Return Codes Produced by the CMS DIRECT Command

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

Invoking Directory as a Stand-alone Program

Stand-alone directory program operation in a virtual machine is the same as CMS operation, with this exception: after IPL, the program asks you for the virtual card reader address. If you enter a null line, the IPL device address is the default of 00C. The directory control statements must be in the same virtual reader file as the directory program itself.

When running as a stand-alone program, DDR searches for a typewriter console (printer keyboard or linemode console) first at address 009 then at address 01F. If these consoles are not operational, the utility enters a WAIT state, waiting for an interrupt to identify the address of the console. The following considerations apply to this method of operation:

- If a nontypewriter console (such as a 3270 device) is physically connected to address 009 or 01F, this console will not be located automatically; press ENTER to identify the console. It may be necessary to reset the keyboard before the ENTER key is accepted.
- If any nonconsole device is physically connected to address 009 or 01F, it must be made nonoperational or the results will be unpredictable.
- Channel-to-channel devices (some terminal and network controllers) and certain other devices can present asynchronous interrupts that interfere with the utility's I/O. It may be necessary to disable these devices for the utility to run as a stand-alone program.

Notes:

1. If the directory program is run stand-alone, it does not check for restricted passwords.
2. Remember to occasionally save a backup copy of your directory on tape.

VM/SP Directory Control Statements

Control statements should be in the formats shown in the following pages, with one or more blanks as operand delimiters. All operands are positional from left to right. If any operand is omitted, remaining operands in that statement must be omitted, with the exception of the OPTION statement. Its entries are self-defining and not positional.

Only columns 1 through 71 are inspected by the program. All data after the last possible operand on any card is ignored. Also, blank cards and cards having an asterisk (*) as the first operand are ignored.

If any input card is in error, the program continues to verify all control statements before ending. If the directory runs out of space, the program stops immediately. After an abnormal ending, the old directory is not altered, and the new directory is not saved.

DIRECTORY Control Statement

The DIRECTORY control statement defines the device on which the VM/SP directory is allocated. It must be the first statement.

Format

```
DIRectory cuu devtype volser [alt-cuu]
```

Parameters

cuu
: is the address of the device to contain the directory and is specified in three hexadecimal digits.

  Note: If the program is being run in a virtual machine, the address is a virtual address. If the program is being run in a real machine (stand-alone), the address is a real address.

devtype
: represents a supported device type suitable for the VM/SP directory (2305, 3330, 3340, 3350, 3375, 3380, or FB-512). For a 3350 device in native mode, specify 3350 as the device type. For a 3350 used in 3330 compatibility mode, specify 3330. Specify a 3344 disk as a 3340, and a 3333 as a 3330.

  Note: 3330V (virtual 3330) volumes associated with 3850 Mass Storage System cannot be specified as the residence device for the VM/SP directory.

volser
: is the volume serial number of the directory volume (one-to-six alphanumeric characters).

alt-cuu
: identifies an alternate device address for writing the directory if the primary *cuu* is not available. For multiprocessing systems, an *alt-cuu* specification allows native execution of the directory program on either processor.

PROFILE Control Statement

The PROFILE control statement specifies the start of a PROFILE entry in the source directory. When specified, PROFILE control statements must follow the DIRECTORY control statements and precede USER control statements. There are no limitations on the number of PROFILE control statements that you can specify.

Format

Profile	*profilename*

Parameter

profilename
specifies the name assigned to the profile entry and references the profile.

Usage Notes

1. A *profilename* is an alphanumeric name containing up to eight characters.
2. Only one *profilename* may be specified on a PROFILE control statement.

USER Control Statement

The USER control statement defines a virtual machine and creates a VM/SP directory entry. It identifies the directory entry for one user. A separate USER statement must be prepared for each directory entry required.

Format

```
User  userid pass [stor [mstor [cl [pri  ┌le  ┌ld  ┌cd  ┌es ┐┐┐┐]]]]
                                         │ON  │ON  │ON  │ON ││││
                                         └OFF └OFF └OFF └OFF┘┘┘┘
```

Parameters

userid

is a one-to-eight character user identification. Any alphanumeric characters may be used except the following:

- The character string, SYSTEM. The string SYSTEM is the user ID of the VM/SP system VMBLOK and should never be used for a virtual machine.
- The VM/SP default logical line editing characters, @, ¢, #, and ″. Line editing characters are used to do the following:
  - Delete the preceding character (@)
  - Delete the preceding logical line (¢)
  - Divide a single line into multiple lines (#)
  - Accept the other line editing characters as data (″).

Note: These characters can be changed in two ways:

1. CP Terminal command
2. USER control statement.

If you plan to use any of the following CMS commands, the *userid* must contain those characters valid for CMS file names.

```
FILELIST    NAMEFIND    NAMES       NOTE
RDRLIST     RECEIVE     SENDFILE    TELL
```

These commands create and reference files that have the *userid* as the file name. Valid characters for CMS file names are A to Z, 0 to 9, $, #, @, +, - (hyphen), : (colon), and _ (underscore).

If you plan to enroll the user in an SFS file pool, do not begin the *userid* with plus (+) or minus (-). Also, the *userid* should not contain a period (.) or colon (:).

Note: It is recommended that line characters, as described earlier, not be used in the *userid.*

There must be at least one CONSOLE or SPOOL directory entry for each user. These options are discussed in the directory program control statement descriptions.

Notes:

1. The *userid* should not contain the characters "LOGONxxx," "LOGOLxxx," or "LOGNxxx," where the x's are one of your terminal addresses. During logon this character string is assigned to the terminal at address xxx from the time the initial interrupt is received until the user is identified.
2. The *userid* should not contain the same characters as the "LUNAME" defined in VTAM. The "LUNAME," although unique to each VTAM service machine, may not be unique to VM/SP if more than one VTAM service machine is running under VM/SP.
3. Do not specify ALL as a *userid.* It is reserved for VM/SP.
4. If the *userid* of AUTOLOG1 (a reserved system user identification) is used, during the VM/SP IPL operation, the AUTOLOG1 virtual machine is automatically logged onto the system.

   In application, the AUTOLOG1 virtual machine could be the CMS batch virtual machine, or a virtual machine that, through the use of the directory's IPL statements, loads a CMS named system. Then the CMS system, using a PROFILE EXEC with AUTOLOG command statements within the EXEC file, initiates the logon of other virtual machines to the system.
5. If the *userid* is one-to-four all numeric characters, unpredictable responses may occur for commands using both the *userid* and a one-to-four digit numeric spoolid as parameters (such as the class D query command).
6. If a *userid* is also a command operand, unpredictable results may occur.

pass

is a one-to-eight character user-security password that must be entered by the user to gain access to the VM/SP system and the virtual machine you are defining in these control statements.

Any character string with one-to-eight characters, uppercase alphabetics (A to Z), numerals (0 to 9), or special characters can be used as a password. Blanks are not permitted. Use of the line-editing characters (@,¢,#,") is not recommended. (See discussion on *userid.*) Any valid character may be the first. Some security products (for example, RACF) can restrict the characters allowed in a password.

Note: Use the reserved password NOLOG for users who do not have a virtual machine configuration in the VM/SP directory. The NOLOG user uses the real card reader spool device as a means of entry for processing by the CMS batch facility. NOLOG is used for spooling purposes only. Attempts to logon using this password are not allowed. Specifying NOLOG provides additional security for your VM/SP installation.

Be sure that the password is not restricted. If you define one of the restricted passwords in the RPWLIST DATA file for a user, it will be changed to NOLOG when the directory program is run. That user will not be able to log on. (See "Passwords That Do Not Work" on page 221.)

stor

is one-to-eight decimal digits that define the virtual machine's storage size. It must be a multiple of 4K. The last character must be K or M. The default is 256K. The minimum size is 8K. All entries not on a 4K boundary are rounded up to the next 4K boundary. The maximum size is 16M.

mstor

is one-to-eight decimal digits that define the maximum virtual machine storage size that this user can define after logging on the system. It must be coded in multiples of 4K. The last digit must be K or M. The default size is 1M. All entries not on a 4K boundary are rounded to the next 4K boundary. The minimum size is 8K. The maximum size that can be specified is 16M.

cl

is a one-to-eight character field defining up to eight user classes assigned to the user. Each character represents a single privilege class. Valid characters are the letters A to Z and the digits 1 to 6. Characters may appear in any order, without duplication, and must not be separated by blanks.

An asterisk (*) placed in this field indicates that a CLASS control statement follows. Use the CLASS control statement to define more than eight privilege classes or as another way to define eight or fewer. If the field is not specified or contains an asterisk with no CLASS control statement, the default class or classes assigned by the SYSFCN macro will be used.

Notes:

1. If privilege class F is assigned to a virtual machine, I/O error recording is not automatically done. This lets the class F user set the kind of error recording desired.
2. Generally, a guest user needs only class G authority. Other classes assigned to guest users can lead to unpredictable results such as system abends or restarts.

pri

is a number from 0 to 99 used by the CP priority dispatcher. Zero is the highest priority; 64 is the default.

Note: The same priority value can be used for many users. Also, if the specification for this statement is not entered, line end (le), line delete (ld), character delete (cd), and escape (es) characters default to system-defined values.

The special VM/SP logical editing symbols below may be set ON, OFF, or substituted with two hexadecimal characters or one graphic character of your choice.

Note: In addition to directory specification, you can change these logical editing symbols using the TERMINAL command. The default value for all symbols is ON. The exception to this rule is a virtual machine started by the CP AUTOLOG command. In this case all logical line editing is OFF.

le

is a one-character *line end* symbol or a two-character hexadecimal representation of the symbol. ON sets the system default value (#). OFF disallows *line end* symbol usage. For example: *le* can be coded as + or 4D or ON or OFF.

ld

is a one-character *line delete* symbol or a two-character hexadecimal representation of the symbol. ON sets the system default value (¢). OFF disallows *line delete* usage.

cd

is a one-character *character delete* symbol or a two-character hexadecimal representation of the symbol. ON sets the system default value (@). OFF disallows *character delete* usage.

es

is a one-character *escape-character* symbol or a two-character hexadecimal representation of the symbol. ON sets the system default value ("). OFF disallows *escape character* symbol usage.

INCLUDE Control Statement

The INCLUDE control statement specifies the name of a PROFILE entry to be used as part of this USER entry. When specified, an INCLUDE control statement must directly follow the USER control statement. Only one INCLUDE control statement may be specified in a USER directory entry.

Format

INclude	*profilename*

Parameter

profilename
specifies the name assigned to the profile entry and references the profile.

Usage Notes

1. A *profilename* is an alphanumeric name containing up to eight characters.
2. Only one *profilename* can be specified on an INCLUDE control statement.
3. Any statement(s) following the INCLUDE statement overrides any previous statement(s).

IUCV Control Statement

The IUCV control statement defines an authorization for establishment of a communication path with another virtual machine or a CP system service. This statement is optional and, if used, must follow the USER control statement or another OPTION control statement. It must precede the first device statement (CONSOLE, MDISK, DEDICATE, LINK, or SPOOL). A virtual machine can initiate communications with itself without directory authorization. The ability to initiate a CONNECT to a CP system service usually requires specific authorization. CP system service communications must be restricted to service type virtual machines.

Format

```
IUCV  ┌ resourceid                                                  ┐
      │ gatewayid                                                   │
      │                                                             │
      │ *IDENT  {RESANY} {LOCAL } [REVOKE]                          │
      │         {resid } {GLOBAL}                                   │
      │                                                             │
      │ *IDENT  {GATEANY}  GATEWAY                                  │
      │         {gateid }                                           │
     {                                                               }
      │ [userid  ]  [PRIORITY]  [MSGLIMIT limit]                    │
      │ [*CRM    ]                                                  │
      │ [*CCS    ]                                                  │
      │ [*SIGNAL ]                                                  │
      │ [*LOGREC ]                                                  │
      │ [*SPL    ]                                                  │
      │ [ALLOW   ]                                                  │
      └ [ANY     ]                                                  ┘
```

Parameters

resourceid
is a one-to-eight character or local resource name that connects to a local or global resource manager rather than to a specified virtual machine. The first byte of the resource name must be alphanumeric. (IBM reserves names beginning with the remaining characters for its own use.)

Be sure that the resource name you specify is not the same as a user ID on the system. Also, do not specify the resource name as any of the following: ALLOW, ANY, or SYSTEM.

Specifying **IUCV** *resourceid* does not give authority to connect by user ID to the virtual machine that owns the specified resource. At the same time, specifying **IUCV** *userid* does not give authority to connect by resource ID to the specified virtual machine.

When you explicitly authorize each virtual machine (with "IUCV resource" or "IUCV ANY"), you should also give explicit directory authorization to the TSAF virtual machine residing on the same system as the resource (with "IUCV resource" or "IUCV ANY").

gatewayid
is a one-to-eight character gateway name. This lets the virtual machine establish a connection through that gateway. The first byte of the gateway name must be alphanumeric. (IBM reserves names beginning with the remaining characters for its own use.) Be sure this name is not the same as any user ID or resource on the system, and that it is not ALLOW, ANY, or SYSTEM.

$$\textbf{*IDENT} \quad \left\{\begin{array}{l}\textbf{RESANY}\\ \textit{resid}\end{array}\right\} \left\{\begin{array}{l}\textbf{LOCAL}\\ \textbf{GLOBAL}\end{array}\right\} \quad [\textbf{REVOKE}]$$

lets the virtual machine connect to the Identify System Service (*IDENT).

RESANY
lets the virtual machine identify any resource name.

Be careful when you assign resource names and when you give authorization for RESANY. A virtual machine that has authorization for RESANY can identify a resource using any resource name, including resany.

resid
is a one-to-eight character resource name. APPC/VM programs can connect to the resource manager that manages the resource specified by *resid.* The first byte of the resource name should be alphanumeric. (IBM reserves names beginning with the non-alphanumeric characters for its own use.)

Be sure that the resource name you specify is not the same as a user ID on the system, ALLOW, ANY, or SYSTEM.

LOCAL
authorizes the virtual machine to identify the resource as a local resource known only to the local system. If you specify LOCAL with RESANY, the virtual machine can identify any resource as a local resource.

GLOBAL
authorizes the virtual machine to identify the resource as a global resource known to all systems in the TSAF collection. This operand also lets the virtual machine identify the resource as local. If you specify GLOBAL with RESANY, the virtual machine can identify any resource either locally or globally.

REVOKE
authorizes the virtual machine to revoke the specified resource name. A virtual machine that can revoke resources can also identify them.

If you specify REVOKE with:

- LOCAL, the virtual machine can revoke and identify the resource on the local system only.
- GLOBAL, the virtual machine can do either of the following:
  - Revoke and identify the global resource.
  - Revoke and identify the local resource on the local system.

  A virtual machine cannot revoke both the global and local resources at the same time. The virtual machine must specify which resource to revoke when the connection is made to *IDENT.
- RESANY and LOCAL, the virtual machine can revoke and identify any local resource.
- RESANY and GLOBAL, the virtual machine can revoke and identify any resource, local or global.

Because the TSAF virtual machines do not keep track of the local resources, a virtual machine cannot revoke a local resource on another system.

For information on using multiple IUCV *IDENT statements for resources, see the *VM/SP Connectivity Planning, Administration, and Operation* book.

***IDENT** $\left\{\begin{array}{l}\textbf{GATEANY}\\ \textit{gateid}\end{array}\right\}$ **GATEWAY**

lets the virtual machine connect to Identify System Service (*IDENT).

GATEANY
lets the virtual machine identify any gateway name.

Be careful when you assign gateway names and when you give authorization for GATEANY. A virtual machine that has authorization for GATEANY can identify a gateway name as "gateany." Also, this virtual machine would be authorized to identify any other gateway name.

gateid
is the one-to-eight character gateway name. The first character of the gateway name should be alphanumeric. (IBM reserves names beginning with the remaining characters for its own use.)

Do not specify the gateway name as any of the following: ALLOW, ANY, or SYSTEM. Also, be sure that this name is not the same as any user ID or resource on the system.

GATEWAY
keyword that indicates you are giving authorization for a gateway and not a resource.

For information on using multiple IUCV *IDENT statements for gateways, see the *VM/SP Connectivity Planning, Administration, and Operation* book.

userid
is the 1-to-8 byte character user identification of the virtual machine or CP system service with which this virtual machine is authorized to establish a communication path. In the case of APPC/VM, this can be the name of a private resource manager virtual machine with which this virtual machine is authorized to start a communication path.

The first byte of this name must be alphanumeric (IBM reserves names beginning with the remaining characters for its own use).

Specifying IUCV *userid* does not give authority to connect by *resourceid* to the specified virtual machine.

***CRM**
is the CP system service name (Collection Resource Management) required for the TSAF virtual machine.

***CCS**
is the CP system service name required for VCNA or VSCS

***SIGNAL**
is the CP system service name required to become a member of a Virtual Machine Group.

***LOGREC**
is the required Error Logging System Service name.

***SPL**
is a system service that lets a virtual machine define and use a logical printer which handles print files.

ALLOW
is a general authorization indicating a communication path can be established from any other virtual machine to this virtual machine. No further authorization is required in the virtual machine initiating the communication.

ANY
is a general authorization indicating a communications path can be established from this virtual machine to any other virtual machine. ANY does not apply to CP system service names.

PRIORITY
indicates that a communications path with the specified virtual machine or CP system service can handle priority as well as nonpriority communications if requested through the CONNECT function. If the PRIORITY option is omitted, no path authorized by this entry can handle priority messages.

MSGLIMIT limit
defines the maximum number of outstanding messages allowed on any path authorized by this entry. A lower limit can be specified through the CONNECT and ACCEPT functions. If omitted, the message limit is taken from the CONNECT or ACCEPT parameter list. Limits specified by CONNECT and ACCEPT can be different. If omitted from the parameter list, a default of 10 is used. The maximum allowed value for message limit is 255. VM performance could be affected if this limit is reached.

Usage Notes

1. PRIORITY and MSGLIMIT are keywords and can be specified in any order.
2. When the CONNECT or ACCEPT functions are invoked, the directory entries are searched in a definite order:
   - The invoker's IUCV control statements are searched for an entry for the target's user ID, resource ID, or gateway ID and then
   - The invoker's IUCV control statements are searched for an ANY entry

   For CONNECT, the search may continue:
   - The target's IUCV control statements are searched for an ALLOW entry.

   The first entry found that is applicable establishes the priority status and message limit for the path.
3. Connections invoked from CP system code do not need directory authorization. Priority status and message limit are taken from the CONNECT parameter list.
4. The IUCV control statement can be used as many times as needed for any user to define authorizations needed for that virtual machine.
5. Communications with Console Communication Services (CCS) for VCNA and VSCS require an IUCV control statement with *CCS as the user ID.
6. Communication with the CP Message System Service and the CP Message all system service does not require authorization by the virtual machine. If an IUCV control statement is specified, a user ID of *MSG or *MSGALL, respectively, should be used. Although the PRIORITY and MSGLIMIT options may be specified on the control statement, the specifications will never be used because the virtual machine will only be receiving messages.

7. Communication with the CP Signal system service does not require authorization by the virtual machine. If an IUCV control statement is specified for the Signal system service, a user ID of *SIGNAL should be used.
8. IUCV ANY and IUCV ALLOW apply to the user IDs on the system, the local and global (not private) resource IDs, and gateways.
   - If IUCV ALLOW is in the directory entry of a particular virtual machine, any user ID or resource ID can access the user ID and resource IDs of that virtual machine.
   - If IUCV ALLOW is in the directory entry of a virtual machine that is a gateway owner, any virtual machine can establish a connection through any gateway owned by that virtual machine.
   - If IUCV ANY is in the directory of a particular virtual machine, that virtual machine or resource can establish a connection through any gateway, or to any other local or global resource, or to any user ID in the system.
9. Communication with the Collection Resource Management and Identify System Services requires that the virtual machine be authorized in the directory entry. This is described in the *VM/SP Connectivity Planning, Administration, and Operation* book.
10. Checking authorization to Connect to *IDENT.
    - For a particular user with more than one IUCV *IDENT entry for resources, CP will look for these entries in this order:

      The first *IDENT entry that has the same local or global resource name as specified on the connect to *IDENT

      The first *IDENT entry that has the local or global resource name RESANY.

      If CP does not find a match, or if the LOCAL/GLOBAL and REVOKE parameters do not correspond to those specified in the CONNECT parameter list, then it severs the requested connection to *IDENT.
    - For a particular AVS virtual machine with more than one IUCV *IDENT directory entry for gateways, CP will look for these entries in this order:

      The first *IDENT entry that has the same gateway name as specified on the CONNECT to *IDENT

      The first *IDENT entry that has the gateway name GATEANY.

      If CP does not find a match, or if the GATEWAY and REVOKE parameters do not correspond to those specified in the CONNECT parameter list, then it severs the requested connection to *IDENT.
11. Communication with the Error Logging System Service requires an IUCV control statement with *LOGREC as the user ID.
12. Specify IUCV ANY for all virtual machines in the GCS environment. This has to be specified or the virtual machines can hang in LOCKWORD processing.

APPCPASS Control Statement

The APPCPASS control statement enhances both the usability and security of VM communications by allowing the storage of communications directory user IDs and passwords in the VM/SP directory instead of in communications directory NAMES files.

The information presented by APPCPASS is intended to be used when an application is using CMSIUCV CONNECT with CMS communications directory support to connect to an APPC resource. When such an application specifies a symbolic destination name corresponding to a communications directory entry containing `:security.pgm`, CMSIUCV CONNECT processing extracts the user ID and password values associated with the symbolic destination. If the `:userid` or `:password` tags are not found in the communications directory, CMS tells CP to extract them from a matching APPCPASS statement. There are two cases:

1. The communications directory entry contains a `:userid` tag but no `:password` tag. In this case, CMS tells CP to fill in just the password value from an APPCPASS statement matching both *luname* and *userid* with the communications directory.
2. The communications directory entry does not have a `:userid` tag. In this case, CMS tells CP to fill in both the userid value and the password value from an APPCPASS statement matching just the *luname* with the communications directory.

In either case, if CP does not find the appropriate APPCPASS statement, it rejects the connection attempt.

Format

APPCpass *luname userid password*

Parameters

luname
is the VM locally known *LU name* referred to above as just the *luname*, consisting of two 1 to 8 bytes names separated by one or more blanks.

1. The first name is the *luname qualifier.* It could be a gateway name, *IDENT, or *USERID.
2. The second name is the *target luname.* If the *luname qualifier* is *IDENT, the *target luname* is the character zero (X'F0')

The locally known LU name specifies the name space for the target LU name. For example, if the *luname* is *USERID, the target *luname* is the user ID of the target virtual machine for a private resource connect.

userid
is the 1-to-8 byte *userid* of the target LU that authorizes the communication initiator to access resources located on the target LU.

password
is the 1-to-8 byte *password* that corresponds to the *userid* at the target LU.

Place the APPCPASS directory statement after the USER statement and before any device definition statements (CONSOLE, DEDICATE, LINK, MDISK, SPECIAL, or SPOOL). You can specify multiple APPCPASS directory statements in a single virtual machine.

CP processing of APPCPASS directory statements depends on the value of the security type field CE4 in the Connect Parameter List Extension (CPLE). This field may have been set by the requesting program or set or changed by CMS, depending upon the results of processing the communications directory.

1. If the CPLE field CE4 contains security **type=3** (X'03'), CP accesses the first APPCPASS statement having that *luname*. That is, even if two or more APPCPASS definitions have different user IDs:, CP will process only the first.
2. The CPLE field CE4 contains security **type=4** (X'04'), CP accesses the first APPCPASS having that *luname* and *userid*. In this case, CP will process APPCPASS statements having different user IDs.

Table 20 summarizes what happens when a user program makes a request using CMSIUCV CONNECT with COMDIR=YES and a Connection Parameter List Extension (CPLE) has already been set up.

Table 20. Communications Directory/APPCPASS Statement for SECURITY(PGM)

Matching COMDIR entry has :security.pgm, and	**CMS/CP Action**
Both *userid* and *password* found for :nick entry	CMS sets CPLE field CE6 to user ID and password, and field CE4 to X'02' (tells CP to use supplied user ID and password). CP will not use any APPCPASS statement.
userid but no *password* included for :nick entry	CMS supplies user ID to CP; sets CE4 field of CPLE to X'04' (tells CP to extract only the password from APPCPASS statement having the same target luname and user ID as that supplied in CPLE. **Note:** Typically, *luname* would also have bee supplied in the communications directory entry. CP will use **first** APPCPASS statement having both the matching target luname and user ID (different user IDs will cause CP to use different APPCPASS statements).
No *userid* found for :nick entry	CMS sets field CE4 of CPLE to X'03' (tells CP to extract both user ID and password from APPCPASS statement having the same target luname as that supplied in CPLE. CP will use the **first** APPCPASS statement having the matching target luname (different user IDs are of no consequence).

For more information on the Connection Parameter List Extension, see the *VM/SP Connectivity Programming Guide and Reference.*

For more information on the APPCPASS statement and the CMS communications directory (including examples), see the *VM/SP Connectivity Planning, Administration, and Operation* book.

CLASS Control Statement

The optional CLASS control statement defines up to 32 user classes assigned to a user. If the CLASS control statement is used, it must immediately follow the USER control statement and the USER control statement must have an asterisk (*) in its class field.

Format

```
CLass   classes
```

Parameter

classes
is a 1-to-32 character field defining the classes assigned to a user, one character per class. Valid characters are the letters A to Z and the digits 1 to 6. As long as no characters are duplicated or separated by blanks, they may appear in any order.

ACCOUNT Control Statement

The ACCOUNT control statement defines an account number and a distribution identification. The distribution identification has no internal system use. It is provided for customer use (for example, a code for distribution of printed output). The ACCOUNT statement is optional. If this statement is omitted, both the account number and the distribution code default to the user ID. This statement (if coded) must follow the USER statement and precede the first device statement.

Format

Account *number* [*distribution*]

Parameters

number
is a one-to-eight alphanumeric character account number punched in the accounting data for this virtual machine. The user ID from the USER statement is also punched in the accounting data.

distribution
is a one-to-eight character distribution identification word that is printed or punched with the user ID in the separator for spooled output for this user. This value is optional and defaults to the user ID from the USER statement if omitted.

ACIGROUP Control Statement (Optional)

The ACIGROUP control statement defines a user as a member of an access control group. This control statement is optional. It must precede the first device statement (CONSOLE, MDISK, DEDICATE, LINK, or SPOOL). Only one ACIGROUP can appear in a single user's directory entry.

Format

```
ACIgroup groupname
```

Parameter

groupname
is a one-to-eight alphanumeric character name of an access group. Only one groupname may be specified.

For more information on access control groups, see page 224.

IPL Control Statement

The IPL control statement contains the name of the system to be loaded for the user when they log on. This statement is optional. If specified, it must follow the USER statement, and must precede the first device statement (CONSOLE, MDISK, or SPOOL). This control statement may contain data to be passed to the system being loaded. The IPL statement can be overridden by the user at logon time by specifying "LOGON user ID NOIPL."

Note: If the user is the primary system operator, an automatic IPL *is* done at logon time.

Format

```
Ipl    iplsys [PARM data]
```

Parameters

iplsys
is a one-to-eight character system name or the one-to-three digit I/O virtual address of the device containing the system to be loaded.

PARM *data*
passes up to 64 bytes of data after the keyword, PARM, excluding all leading and trailing blank characters, but including all embedded blanks, to your virtual machine's general purpose registers (4 bytes per register), starting with the high order byte of general register 0.

Optional CMS Initialization Parameters

The following CMS parameters are passed to the System Profile EXEC (SYSPROF EXEC). Any unrecognized parameters are ignored by CMS initialization, but are still passed to the SYSPROF EXEC.

CMS users can use the PARM operand to specify any of the following CMS parameters:

AUTOCR
is the automatic carriage return parameter that simulates the pressing of the ENTER key as input to the virtual machine at the initial VM READ. Your PROFILE EXEC is automatically executed if it exists on your virtual minidisk 191.

BATCH
indicates that the CMS IPL is being performed in a batch instead of an interactive virtual machine.

This parameter does not affect execution of the SYSPROF EXEC. Execution of the SYSPROF EXEC will be suppressed only if the NOSPROF parameter was specified.

At the beginning of each job, the batch facility work disk is accessed and then immediately erased, preventing the current user job from accessing files that might remain from the previous job. Because of this, execution of the PROFILE EXEC is disabled for the CMS Batch Facility machine. However, if the BATPROF EXEC exists on an accessed system disk, it is issued instead of the PROFILE EXEC.

NOSPROF
indicates that the SYSPROF EXEC is bypassed. You must specify NOSPROF if you intend to enter the CMSBATCH command when the initial VM READ is issued. When the IPL is of a non-DASD device or the SYSPROF EXEC does not exist, you do not have to specify NOSPROF to enter the CMSBATCH command.

INSTSEG YES
links the default Installation Discontiguous Shared Segment (DCSS) for this CMS session. The Installation DCSS is an optional shared segment that contains execs and editor macros that your installation provides.

INSTSEG NO
indicates that you do not want to use the Installation (DCSS) during this CMS session.

INSTSEG *name*
links the specified Installation DCSS for this CMS session.

FILEPOOL *filepoolid*
is a one-to-eight character name that identifies the SFS file pool in which the user's top directory resides. When FILEPOOL is specified, the user's top directory is accessed as file mode A during CMS initialization. If FILEPOOL is not specified, the 191 disk is accessed as file mode A.

CMS also recognizes the SAVESYS parameter for Privilege Class E. See the *VM/SP Application Development Guide for CMS* for more information.

Usage Notes

1. Although the VM/SP IPL command allows up to 64 characters on the PARM option, the directory restricts each statement to a single card image, limiting the number of characters that can be entered on the IPL Control Statement in the directory. The *PARM data* on the IPL Control Statement is loaded into general registers 0 to 12 and is passed to the application specified by *iplsys*.
2. All virtual machine registers specified as PARMRGS in the NAMESYS macro for a named system will contain trailing binary zeros following the PARM data.
3. If you want to use the NOSPROF, INSTSEG, or SAVESYS parameters of the IPL command, you must have

   ```
   PARM=(0,15)
   ```

   in the NAMESYS macro for the CMS named system.

OPTION Control Statement

The OPTION control statement selects specific options. This statement is optional and, if used, must follow the USER statement or another OPTION statement. It must precede the first device statement (CONSOLE, MDISK, DEDICATE, LINK, or SPOOL). Multiple OPTION statements can be inserted if the options selected exceed one statement record length.

Format

```
Option  [Realtimer] [Ecmode] [CONceal] [Isam] [VIRT=REAL]

        [Acct] [Svcoff] [Bmx] [Cpuid bbbbbb]

        [AFfinity nn] [VMsave] [STFirst] [370E]

        [Maxconn nnnnn] [MIh] [Diag98] [COMsrv]

|       [Lang langid] [VCUnoshr] [APPLmon]

|       [SVMstat] [DEVInfo] [DEVMaint]
```

Parameters

Realtimer
provides a timer for the virtual machine that is updated during virtual processor run time and during virtual wait time. If the virtual machine does not have the REALTIMER option, its timer reflects only the virtual processor run time used. This option is required for virtual machines running systems or programs that go into a wait state expecting a timer interruption. This timing ability can also be obtained by entering the CP command SET TIMER REAL.

Ecmode
lets the virtual machine run in extended control mode. The Ecmode option must be specified for virtual machines using operating systems that:

- Operate in System/370 extended control mode (such as VM/SP itself).
- Use the DAT facility (such as OS/VS1, OS/VS2, DOS/VS, DOS/VSE, VSE/AF, VM/370, and VM/SP).
- Use control registers other than zero (such as OS GTF (General Trace Facility), which uses Monitor Call and requires control register eight).
- Depend on the System/370 extended channel masking feature.

The Ecmode option must also be specified for the virtual machine that is to perform system support or updating. Ecmode is also required when using the clock comparator.

Note: A virtual machine defined without the Ecmode option in the directory is limited to six I/O channels, while a virtual machine with the Ecmode option may address up to 16 I/O channels. If a virtual machine with Ecmode runs in basic control mode, the I/O masking for channels 6 and higher is simulated by the extended channel feature. If a virtual machine with the Ecmode option runs in extended control mode, the I/O masking for all 16 channels is handled by way of the extended control register 2. This facility can also be obtained by entering the CP command SET ECMODE ON.

CONceal
places the user in a protected application at logon time. When a user is operating in a protected application:

- Multiple attentions will not cause the user to enter CP mode.
- TERMINAL BRKKEY is set to NONE.
- CP will initiate an automatic re-IPL upon encountering errors such as virtual machine disabled wait, paging error, invalid PSW, external interrupt loop, program interrupt loop, and translation exception.
- If a shared page is altered, CP will attempt to resume execution in the virtual machine before initiating an automatic re-IPL.

CP will limit the attempts to re-IPL a user to avoid keeping the user in a re-IPL loop. If an error causing a re-IPL attempt occurs within one minute from the last re-IPL attempt, or occurs when 10 re-IPLs have been attempted since entering the application environment, the user is placed in CP mode and the message corresponding to the error is displayed on the screen.

Note: The system indicated on the IPL card in the directory entry for the user is checked and re-IPLed first. Otherwise, the named system that was running at the time of the error is re-IPLed. This facility can also be obtained by entering the CP command SET CONCEAL ON.

Isam
provides special channel command word translation routines that permit OS/PCP, MFT, and MVT ISAM programs (that dynamically modify their CCWs) to operate properly in a virtual machine. This is required only for virtual machines that use OS/PCP, MFT, or MVT ISAM access methods or OS/VS ISAM when running either in a V=R partition or in nonpaging mode under OS/VS. This option is not needed for DOS, DOS/VS, DOS/VSE, VSE/AF, or OS/VS ISAM when run only in a V=V partition of OS/VS. This facility can also be obtained by entering the CP command SET ISAM ON.

VIRT=REAL
is a performance option that lets you place your virtual machine in lower storage, such that its virtual storage addresses correspond to real storage addresses (except page 0, which is relocated). Real page 0 is controlled by the CP nucleus. No CCW translation is required. This option is required for a virtual machine to successfully execute self-modifying channel programs other than those generated by OS/VS TCAM (Level 5, generated or invoked with the VM/SP option) or OS ISAM. VIRT=REAL can be specified for any number of virtual machines, but only one virtual machine can use this facility at any one time. A named or shared system cannot be loaded (by way of IPL) in a virtual=real area. The device address must be specified in the IPL command. To generate a VM/SP system with a virtual=real machine, see "Specifying a Virtual=Real Machine" on page 161.

Acct
lets users track another's use of virtual machine resources. For example, a user who sends a job to the CMS batch virtual machine can be charged for time used in the batch machine. Note that the Acct option should be specified in the directory of the CMSBATCH virtual machine so user/job identifying information is printed on the forms separators of spooled output files. Also note that specifying this option causes rerouting of spooled print and punch output.

Svcoff

specifies that CP, instead of the virtual machine assist feature or the VM/370 Extended Control Program Support handles all SVC interrupts for this virtual machine. A user whose directory entry contains this option can override it by entering SET ASSIST SVC.

Note: CP handles all SVC 76 interrupts, whether or not the Svcoff option is specified.

Bmx

specifies that all virtual machine I/O operations are to occur as block multiplexer channel operations rather than selector channel (the default) operations. In block multiplexer mode, the virtual channel is not busy until the initial SIO is complete (selector mode operates similarly). Block multiplexer allows the successful start of multiple SIOs to different devices on the same channel. However, virtual I/O operations on channel 0 are processed as byte multiplexer channel operations.

The channel mode setting for all channels except virtual channel zero can be changed by the CP DEFINE CHANNEL command.

Cpuid*bbbbbb*

provides a processor identification (CPUID) to be stored in response to the STIDP instruction. It is necessary to associate a different Cpuid with each virtual machine attached to an MSC port. This is because solicited and/or unsolicited messages are directed to the host system in the virtual environment using the Cpuid. There is no checking by VM/SP to ensure that all virtual machines using the SET Cpuid command have specified different processor serials. The hexadecimal field 'bbbbbb' is the processor identification number. The processor identification number (serial) is only a portion of the complete Cpuid. The Cpuid identification stored in response to a STIDP instruction is a string of 16 hexadecimal digits shown as follows:

```
aabbbbbbccccdddd
```

where

aa

is the version code. These two digits are forced to X'FF' to identify that the virtual machine is running under VM/SP.

bbbbbb

is six hexadecimal digits that indicate the processor identification number. This field is set by the directory OPTION statement values or modified by the SET CPUID command.

cccc

is the model number. This field contains a high order 0 digit, followed by the three digits of the model number (0 to 9). This field defaults to the model number of the real machine.

dddd

is the machine check extended logout. This field is forced to X'0000' because CP does not reflect machine checks to the virtual machine.

If the Cpuid is not specified by the SET CPUID command or the OPTION control statement, the Cpuid stored as a result of the STIDP instruction is the real Cpuid. The first two digits are set to X'FF' and the last four digits are set to X'0000' (present Cpuid logic). A processor serial of more than six digits on the SET CPUID command results in an error message.

A processor identification number (serial) of fewer than six hexadecimal digits results in zeros to the left of the number. A 3-byte field in the VMBLOK (VMCPUID) contains the value set as a result of invoking this DIRECTORY option.

AFfinity *nn*
is two decimal digits between 00 and 63, specifying that virtual machine execution is done on a designated processor (nn). This attribute is useful only in the VM/SP attached processor and multiprocessor environments. Any hexadecimal value from 00 to 3F is a valid main (IPL) or attached (non-IPL) processor address. However, the value selected must match the preset values established for your IPL and non-IPL processors when the system was installed. If the AFfinity option is not selected, the virtual machine is serviced by the first available processor from the VM/SP dispatch queue. An affinity setting in the VM/SP directory can be overridden by the CP SET AFFINITY command. If the system is running in attached processor or multiprocessor mode and an error forces recovery to uniprocessor mode, the affinity setting of virtual machines assigned to the non-IPL processor is cancelled. Virtual machine processing may continue on the IPL processor.

VMsave
specifies that the virtual machine contents are to be saved if VM/SP is terminated or if VM/SP terminates the indicated virtual machine. The contents of the registers and real storage of the virtual machine are saved on DASD space and made available to user ID(s) specified by the NAMESYS macro instruction.

Notes:

1. This option is effective only if you have exactly one VMSAVE DASD area defined. The option is enabled only if that area does not contain a VMSAVE system. If more than one area is defined, or if a valid system is already stored in that area, the SET VMSAVE command must be used.
2. To cancel the VMsave specification use the SET VMSAVE OFF command.

STFirst
specifies that the indicated virtual machine is authorized to use the SET STBYPASS command when virtual machine assist is active on the system for a virtual=virtual user.

Note: This is a restricted performance option that should be reserved only for virtual machine user IDs used to run production MVS, SVS, OS/VS1, or DOS/VS operating systems.

370E
specifies that the MVS/System Extensions support be enabled for the indicated virtual machine. See the table under "Improving Performance" on page 160, for a list of VM/SP processors that support this option.

Maxconn *nnnnn*
is the maximum number of Inter-User Communications Vehicle (IUCV) connections allowed for this virtual machine. If the Maxconn option is omitted, the default is 4. The maximum is 65,535.

Note: The Maxconn option is applicable only to virtual machines. For CP system code, a limit of 4,096 paths is established when the CP system is initiated.

MIh
specifies that Missing Interrupt Handler support be enabled for the indicated virtual machine. When a missing interrupt is detected, CP simulates an interrupt.

Diag98
authorizes the indicated virtual machine to use DIAGNOSE code X'98'. See the *VM System Facilities for Programming* book for information about DIAGNOSE code X'98'.

Note: A virtual machine should only use the real addresses returned by DIAGNOSE code X'98' in its real channel programs. The virtual machine is responsible for any security violations it may cause from using any other real addresses.

COMsrv
authorizes the indicated virtual machine to act as a communication server to:

- Route connections for other virtual machines to other servers
- Establish connections to other servers while handling requests for other users.

With this option, the TSAF virtual machine or any other communications server can put the user ID of the virtual machine that issued APPC/VM CONNECT in the CONNECT parameter list.

When TSAF sends the connect request to the target resource-manager virtual machine, the request contains this information about the originating virtual machine. Without this option, CP would send the connect request with the communications server's user ID.

Lang *langid*
identifies the language that should be set for the virtual machine during logon. Use up to five characters to specify the language ID. It corresponds to the langid listed in the NAMELANG macro. If you do not specify the Lang option, CP sets the system national language for the virtual machine; so, specify the Lang option for every virtual machine user who does not want the system national language set.

If a valid CP message repository cannot be accessed for the langid you specify on this option, CP sets the system national language.

| This facility can be obtained by issuing the CMS SET LANGUAGE command.
| This command issues a DIAGNOSE code X'C8' instruction to CP to change
| the LANGBLOK.

VCUnoshr
specifies that all devices connected to the virtual machine are to be supported using NONSHARED protocol for virtual I/O operations. This does not affect the real protocol of attached devices. If this option is not present, VCU protocol will be determined by the type of device or real control unit attachment.

| **APPLmon**
| lets the user invoke DIAGNOSE code X'DC'. See the *VM System Facilities for Programming* book for more information about DIAGNOSE code X'DC'.

| **SVMstat**
| identifies Service Virtual Machines (SVMs) in the directory. This option is not directly used by CP. Instead, the option is passed to applications through monitor data. Monitor data reduction-programs must receive this information in the monitor data to determine how to interpret the transaction data of a virtual machine.

DEVInfo
authorizes a user to invoke the X'00' and X'01' subcodes of DIAGNOSE code X'E4'. DEVInfo authorizes these functions when a user ID other than the issuer's is provided in the input parameter list. When the user ID specified in the parameter list is the same as the issuer's, no directory authorization is required. See the *VM System Facilities for Programming* book for information about DIAGNOSE code X'E4'.

DEVMaint
authorizes a user to invoke the X'02' and X'03' subcodes of DIAGNOSE code X'E4'. DEVMaint also authorizes subcodes X'00' and X'01' when a user ID other than the issuer's is provided in the input parameter list. See the *VM System Facilities for Programming* book for information about DIAGNOSE code X'E4'.

CONSOLE Control Statement

The CONSOLE control statement specifies the virtual console.

Format

Console *addr devtype* [*class*] [*userid*]

Parameters

vaddr
is the virtual device address of one-to-three hexadecimal digits.

devtype
is the device type:

1052
3210
3215
3270

A 3270 specification forces TERMINAL CONMODE 3270 but is otherwise identical to a 3215 specification. Similar to the TERMINAL CONMODE 3270 specification, the devtype 3270 specification applies only to local non-SNA display devices that have a 3270 compatible command set. These devices are:

- Model 138 console
- Model 148 console
- Model 158 console
- 3277 (Model 2)
- 3278 (Models 2, 3, 4, 5, 2A, 2C, and 3C)
- 3279 (Models 2A, 2B, 2C, 3A, and 3B)
- 3290
- 3036.

Only one console may be specified. If a different console is sometimes required, use the CP DEFINE command to change the console address or add an alternate console.

For a 1052, 3210, 3215, or 3270 specification the system accepts any real console or terminal. The 3270 is not supported for disconnected users; full screen channel programs will be command rejected when a user is disconnected. The virtual device created by any console statement requires a nonshared virtual control unit.

class
is a one-character spooling class. A through Z and 0 through 9 may be specified. The class governs printing of real spooled output. If the class operand is omitted, the default for console spooling is class T.

userid
is the one-to-eight character secondary *userid* whose console is to be used when the user disconnects.

If *userid* is specified, the class operand must also be specified

Note: CP does not let the user mix the two types of subchannel protocols, shared and nonshared, on a single virtual control unit. See "Matching Hardware to Shared or Nonshared Virtual Control Units" on page 407 for the listing of the protocols used by specific devices.

DEDICATE Control Statement

The DEDICATE control statement specifies that a real device is to be dedicated to this user. MSS 3330V (virtual 3330) volumes may be specified by way of the DEDICATE statement. If the device is a unit record device, input and output are not spooled by VM/SP. A real device may be dedicated to only one user at a time. Should a device be specified as dedicated in more than one directory entry, only the first user to log on gains access to it.

Format

```
Dedicate   { NETwork   vaddr resource                                          }
           {           vaddr  {rdev  [VOLID]  [volser]  [3330V]  [R/O] }       }
           {                  {      [VOLID]   volser   [3330V]  [R/O] }       }
```

Parameters

NETwork
is the keyword used if a remote 3270 Information Display System Printer (3284, 3286, 3287, 3288, or 3289) is to be automatically attached to a virtual machine at logon time. If this keyword is omitted, a local device is assumed.

Note: When a network printer is dedicated it must use a shared virtual control unit.

vaddr
is the one-to-three character virtual device address.

resource
is the four character resource ID of a remote device as specified in DMKRIO. This operand *must* be specified if the NETwork keyword is specified, and is only valid if NET is specified.

rdev
is the one-to-three character real device address.

VOLID
is the keyword that must be used if the *volser* is less than four characters long and *rdev* is not specified. It is optional when *rdev* is specified or volser is a length of four or more characters. In cases when *rdev* is not specified, the CP system will find an available *rdev.*

If the VOLID operand is used, the volume must be attached to the system when the user logs on. When the user logs off, the operator can then detach the volume from the system.

volser
is the volume serial number of a disk pack mounted on some real disk storage device, or of an MSS volume to be dedicated to the virtual machine. The *volser* can be from one-to-six alphanumeric characters long.

3330V
specifies that all interruptions, including cylinder faults and attentions received on the rdev are to be passed to the virtual machine in its cuu.

R/O

specifies that the virtual device is to be in read-only status. If this operand is omitted, the status defaults to read/write.

Usage Notes

1. When you dedicate a 2305 device, both the real and virtual device addresses must specify the first exposure on the 2305 (that is, device address 0 or 8). When you dedicate a 2305 to or detach a dedicated 2305 from a user, all 8 exposures are processed.

2. Use caution in defining the hexadecimal addresses of virtual devices (vaddr) in DEDICATE statements, in order to avoid a usage conflict caused by control unit I/O interface protocol. Some devices use a shared subchannel protocol and others do not. (To avoid this problem, see Appendix A, "VM/SP Configuration Aid" on page 401 for a complete listing of the virtual device characteristics.) Devices must be grouped by control unit within a given channel according to their subchannel usage. CP does not permit you to group devices that use the shared subchannel protocol together with devices that do not use the shared protocol. The following is an example of a virtual machine's DEDICATE statements that are rejected at logon time.

   ```
   DEDICATE 10E 30E (30E is a real 3211)
   DEDICATE 10F 30B (30B is a 2400 tape device)
   ```

   The virtual addresses of both the 3211 and the tape device indicate the use of the same channel and control unit. By definition the devices are virtual and therefore will share one virtual control unit (VCUBLOK) in CP. A real 3211 printer operates on a nonshared subchannel, and the real 2400 device is designed for shared subchannel operations. Both of these real devices are mapped to the same VCUBLOK. When the user logs on, the two dedicate statements result in an error message being sent to the user and the second virtual device (10F) is not created. Therefore, when defining devices, make sure the devices are defined (and separated) within their own control unit range and not shared with other devices.

   Because there is no control unit on the real hardware for a system console it should be noted that this restriction applies to any system console such as the 3138, 3148, and 3158. Note that an attached terminal to be used as a guest system console should not share a VCU with any other shared protocol devices.

   Examples:

   ```
   DEDICATE 0B8 0B0
   ```

   is a DEDICATE statement for a device at real address 0B0. Its virtual address is 0B8.

   ```
   DEDICATE 250 MYPACK
   ```

   is a DEDICATE statement that defines, for this virtual machine, virtual address 250 as the real device where DASD volume MYPACK is mounted.

   This restriction also applies to the CONSOLE, MDISK, SPECIAL, SPOOL, and LINK statements. The effects of the DEDICATE, LINK and MDISK statements depend on the real device configuration at LOGON time. Remote 3270 Information Display System Printers can also be attached by the NETWORK ATTACH command. See the *VM/SP Operator's Guide* for more details.

A network attached printer is supported as a shared virtual device. See 2 on page 425 under Appendix B, "VM/SP Restrictions" on page 411 for more information.

3. When the real device is a 3330V, the action VM/SP takes in processing the DEDICATE statement at logon time depends on the combination of operands specified. Following are the allowable combinations and the control program action for each:

   ```
   DED vaddr rdev
   ```

   The real device must have the VIRTUAL feature (not SYSVIRT). The real device will be dedicated to the virtual machine as virtual device cuu, which is a 3330-1. All cylinder fault activity on the rdev will be processed by VM/SP, transparent to the virtual machine.

   ```
   DED vaddr rdev 3330V
   ```

   The real device must again be a VIRTUAL 3330V. All cylinder faults and unsolicited interrupts received by VM/SP on the rdev will be passed to the virtual machine.

   ```
   DED vaddr VOLID volser
   ```

   When processing this statement, the control program will allocate an available SYSVIRT 3330V and dedicate that real device to the virtual machine as virtual device vaddr. The MSS volume having volser will be mounted on the real device, and the virtual device will be a 3330-1. This form of DEDICATE dedicates volumes to non-MSS operating systems, such as CMS, because the control program chooses the real device address and no cylinder fault interrupts are passed to the virtual machine.

   ```
   DED vaddr rdev volser
   ```

   The difference between this example and the previous one is that in this case the real device address is preselected and must have the VIRTUAL feature. This format lets you control which real devices are dedicated to virtual machines, rather than having the control program choose a device address when the statement is processed.

   ```
   DED vaddr rdev volser 3330V
   ```

   This format is the same as the previous one, except that the virtual device becomes a 3330V, such that VM/SP does not intercept any cylinder fault interrupts or the associated attention interrupts.

4. There are considerations that must be made when dedicating real 3330Vs to a virtual machine that also has a dedicated MSC port and is running an OS/VS operating system with MSS support. (See Appendix B, "VM/SP Restrictions" on page 411.)

5. When dedicating a real CTC, the CTC should be on a separate real channel from all other virtual devices because of a possible lock-out problem.

6. A dedicated 328x printer (like any dedicated device) is supported using the same protocol as the real control unit definition in DMKRIO. A dedicated 328x printer is usually supported as a shared virtual device.

7. When dedicating a non-MSS DASD, the volser option cannot be specified with the rdev option.

8. When dedicating an Ethernet or Token Ring LAN adapter to a TSAF virtual machine you must do the following in the TSAF CP directory:

   - Specify option DIAG98 on the OPTIONS statement, for example:

     ```
     USER TSAFVM password 4M 8M G
      OPTIONS  . . . DIAG98 . . .
       ...
     ```

     DIAGNOSE code X'98' lets the TSAF virtual machine perform the I/O on the Continuously Executing Transfer Interface (CETI) group without disrupting CP and other virtual machines.

   - For each CETI group that the TSAF virtual machine uses, you must dedicate (DEDICATE statement) or attach (CP ATTACH command) four devices to the TSAF virtual machine. The real device addresses must be consecutive and the virtual device addresses must be consecutive. For example:

     ```
     DEDICATE 300 500
     DEDICATE 301 501
     DEDICATE 302 502
     DEDICATE 303 503
     ```

     Note: Virtual subchannel addresses do not have to be the same as real subchannel addresses, but they must be consecutive and start within the range of X'00' and X'F8'.

LINK Control Statement

The LINK control statement makes a device that belongs to another user (user ID) available to this virtual machine at logon time. If you want to make one volume available to many virtual machines:

- Define the volume for one of the virtual machines with an MDISK statement.
- Define a link to that volume, with the LINK statement for all other virtual machines that use the volume.

Later, if you must move or change that volume, you need only update the one MDISK statement; the LINK statements need not be updated.

The LINK control statement and the MDISK control statement have the same authority level (neither has higher priority than the other).

Format

Link *userid* *vaddr1* [*vaddr2*[*mode*]]

Parameters

userid
is the one-to-eight character user identification of the user to be linked to.

vaddr1
is the virtual device address of the device to be linked to, which is owned by *userid*. This virtual device address consists of three hexadecimal digits.

vaddr2
is the virtual device address that the device is to be linked-as for the virtual machine being defined. If not specified, *vaddr2* defaults to the same address as the linked-to device (three hexadecimal digits). If your virtual machine has the ECMODE option, any address up to X'FFF' is valid; otherwise, any address up to X'5FF' is valid.

mode
is the access mode that consists of up to two letters. The first letter specifies the primary access mode (read-only, write or multiple). The optional second letter indicates the alternate access mode (read-only or write access) desired if the primary access is not available. Valid modes are:

Mode	Meaning
R	Primary read-only access. The read-only link is established as long as no other user has the disk in write status. If there is an existing write link to the disk no link is given. R is the default mode if the link is to another user ID.
RR	Primary read-only access or alternate read-only access. The read-only access is established even if another user has the disk in write status. The alternate access of R assures that the user will get the read link no matter what links currently exist to the disk.
W	Primary write access. The write link is established only if there are no other current links to the disk. If another user has the disk in read or write status no link is given.

WR	Primary write access or alternate read-only access. If write access is available then the link is established. Otherwise, the alternate access of a read-only link is given.
M	Primary multiple access. A write link is established unless another user already has write access to the disk, in which case no link is given.
MR	Primary multiple access or alternate read access. A write link is established unless another user already has write access to the disk, in which case a read link is given because it was the alternate access requested. **Note:** Unpredictable results can occur when one user has a read-only (R or RR) link to a device that is being updated by a user who has the device in write status (W or WR).
MW	Primary multiple access or alternate write access. A write link is established in all cases.

Caution

CMS supports multiple accessed read-only disks in full. CMS does not support write access to disks by multiple users. Also, CMS does not protect a user from loss of data on a disk when multiple users have write access to the disk. More than one user writing to the same virtual device can result in a permanent loss of data. CMS disks should never have more than one existing write link at a time.

A disk accessed in write mode by one CMS user is available to other CMS users in read-only mode, but those files altered by the write-mode user cannot be read by the other users.

Note: If the mode is not specified, the default is R.

It is the responsibility of the operating system running in each virtual machine to keep data from being destroyed or altered on shared disks.

If userA owns a virtual device obtained through a directory MDISK statement:

```
MDISK  100  3380  5  10  VMDISK  W  READ  WRITE
```

Then userB may have a directory LINK control statement to obtain this device at logon time:

```
LINK  userA  100  200  RR
```

Any number of users may have directory LINK control statements to either userA's or userB's device. However, if userC has a directory LINK to userB's device (that was obtained by a LINK to userA's device):

```
LINK  userB  200  300  RR
```

then no user can obtain a LINK (either through a directory LINK control statement or the LINK command) to this device through userC because no more than two levels of indirect directory links are permitted.

Note: At logon time, as the directory control statements for the user are processed, CP checks the devices represented by each MDISK, CONSOLE, DEDICATE, LINK, SPECIAL and SPOOL statement for a possible conflict with the VCU interface. This conflict occurs because the virtual control unit cannot support two different subchannel protocols at the same time (shared and nonshared). For each directory control statement that violates the restriction, CP sends an error message to the user and does not create the virtual device. To avoid this problem Appendix A, "VM/SP Configuration Aid" on page 401 should be referenced for a complete listing of the virtual device characteristics.

MDISK Control Statement

The MDISK control statement describes the cylinder extent to be owned by the user on a direct access device. The DASD area assigned with this statement becomes the user's minidisk. During logon, the owner of the minidisk obtains a link to it in the access mode specified on the MDISK control statement.

Warning

Neither CP nor the directory checks that minidisks defined with the MDISK statement do not overlap each other, and (for 3330, 3340, 3350, 3375, and 3380 disks) that they do not overlap the *alternate track* cylinders at the end of the real disk. If overlap occurs, file damage is inevitable.

Format

```
Mdisk  cuu  devtype  ┌ cylr     cyls  volid  [mode [pr [pw [pm]]]] ┐
                     ┤ T-DISK   cyls                                ├
                     └ blkr     blks                                ┘
```

Parameters

cuu
is the virtual device address of one-to-three hexadecimal digits. If your virtual machine has the ECMODE option, any address up to X'FFF' is valid; otherwise, any address up to X'5FF' is valid.

devtype
is the device type:

2305
3330
3340
3350
3375
3380
FB-512

For a 3350 device in native mode, specify 3350 as the device type. For a 3350 used in 3330 compatibility mode, specify 3330. Specify a 3344 disk as a 3340, and a 3333 as a 3330. For a 3330V system volume, specify 3330 as the device type.

$\left\{\begin{array}{l} \textit{cylr} \\ \textit{T-DISK} \\ \textit{blkr} \end{array}\right\}$

is a four-digit decimal cylinder relocation factor that specifies the cylinder on a real disk that corresponds to cylinder 0 of the virtual disk. If T-DISK (temporary disk) is specified, temporary disk space is obtained at logon time from preallocated system disk space. This space must be initialized or formatted by the user when they log on. It is a part of their virtual configuration until they log off or detach the disk. The data area is then returned for reallocation for another T-DISK area. To maintain security, this area should be erased before it is returned. The *blkr* parameter specifies the relocation factor in blocks for FB-512.

Note: It is not advisable to start a minidisk at real cylinder 0 (unless the minidisk is to be used by OS ISAM, in which case it must begin at real cylinder 0). If you do assign a minidisk beginning at real cylinder 0, the user who owns it must realize that the minidisk label is the real label that both they and the VM/SP system use to identify the disk. CP-owned volumes must not have minidisks beginning at real cylinder 0.

cyls
is a one-to-four digit decimal number specifying the number of cylinders.

blks
is a one-to-six digit decimal number specifying the number of FB-512 blocks.

Maximum Minidisk Sizes (cylinders or blocks)

Device Type	Model(s)	CMS/VSAM	CMS 800-byte Format	CMS 512, 1K, 2K, or 4K Format
3310	-	126,016 blocks	Not Supported	126,016 blocks
3330	1 or 2	404 cyls.	246 cyls.	404 cyls.
3330	11	808 cyls.	246 cyls.	808 cyls.
3333	1	404 cyls.	246 cyls.	404 cyls.
3333	11	808 cyls.	246 cyls.	808 cyls.
3340	35	348 cyls.	348 cyls.	348 cyls.
3340	70	696 cyls.	682 cyls.	696 cyls.
3350	Native mode	555 cyls.	115 cyls.	555 cyls.
3370	A1 or B1	558,000 blocks	Not Supported	558,000 blocks
3370	A2 or B2	712,752 blocks	Not Supported	712,752 blocks
3375	-	959 cyls.	182 cyls.	959 cyls.
3380	AD4 or BD4	885 cyls.	121 cyls.	885 cyls.
3380	AE4 or BE4	1,770 cyls.	121 cyls.	1,770 cyls.
3380	AK4 or BK4	2,655 cyls.	121 cyls.	2,655 cyls.
9313	A01 or B01	246,240 blocks	Not Supported	246,240 blocks
9332	400 or 402	360,036 blocks	Not Supported	360,036 blocks
9332	600 or 602	554,816 blocks	Not Supported	554,816 blocks
9335	A01 or B01	804,714 blocks	Not Supported	804,714 blocks

If a device is formatted by IBCDASDI or Device Support Facilities for 3375/3380, the minimum minidisk size is 1 cylinder.

volid
is the volume identifier of the DASD volume (one-to-six alphanumeric characters).

mode
is the access mode that consists of up to two letters. The first letter specifies the primary access mode (read-only, write or multiple). The optional second letter indicates the alternate access mode (read-only or write access) desired if the primary access is not available. An optional 'V' character, when appended to the mode request on an MDISK statement, specifies virtual reserve/release processing. Valid modes are:

Mode	Meaning
R	Primary read-only access. The read-only link is established as long as no other user has the disk in write status. If there is an existing write link to the disk no link is given. R is the default mode if the link is to another user ID.

RR	Primary read-only access or alternate read/only access. The read-only access is established even if another user has the disk in write status. The alternate access of R assures that the user will get the read link no matter what links currently exist to the disk.
W	Primary write access. The write link is established only if there are no other current links to the disk. If another user has the disk in read or write status no link is given.
WR	Primary write access or alternate read-only access. Any prior access of this minidisk, whether it be write or read-only, will result in read-only access for all subsequent attempts.
M	Primary multiple access. A write link is established unless another user already has write access to the disk, in which case no link is given.
MR	Primary multiple access or alternate read access. A write link is established unless another user already has write access to the disk, in which case a read link is given because it was the alternate access requested.
	Note: Unpredictable results can occur when one user has a read-only (R or RR) link to a device that is being updated by a user who has the device in write status (W or WR).
MW	Primary multiple access or alternate write access. A write link is established in all cases.

Caution

CMS does not protect a user from loss of data on a disk when multiple users have write access to the disk. More than one user writing to the same virtual device can result in a permanent loss of data. Users should not be linking with MW mode to obtain the M or MR function. (The M or MR access modes will allow only one write link to a disk.)

If a 'V' is appended to the immediate right of the primary access mode specification (or the alternate access mode specification, if any), then CP's virtual reserve/release support will be used in the I/O operations for the specified device. Thus, if the mode specified for a minidisk is MWV, the minidisk will function with write linkage using CP's virtual reserve/release function.

If a mode specification is omitted from the MDISK statement, it defaults to W.

pr	is the password that lets other users share the device in read-only mode (a one-to-eight character field).
pw	is the password that lets another user access the device in write mode (a one-to-eight character field).
pm	is the password that lets other users gain multiple access to the device (a one-to-eight character field).

Usage Notes

1. A write password (pw) cannot be specified without a read password (pr). A multiple password (pm) cannot be specified without both a read password (pr) and a write password (pw).
2. If ALL is used for pr, pw, or pm, any user is allowed to link with the corresponding access mode to this minidisk without specifying a password.
3. When MSS support is used, the volume serial number may specify an MSS 3330V volume. In this case, the volume serial number must be six characters.
4. If the MSS communicator is initialized when the virtual machine logs on, and the system volume having a volume label of 'volser' is not mounted, VM/SP attempts to find an available SYSVIRT 3330V and mount 'volser' on that device.
5. If virtual reserve/release processing is requested, minidisk users with read or write access are prevented from accessing a minidisk reserved by another virtual machine.
6. Protecting minidisks by specifying passwords on the MDISK statement provides additional security for your VM/SP installation.
7. CP does not let the user mix the two types of subchannel protocols, shared and nonshared, on a single virtual control unit. The virtual minidisk requires a virtual control unit that is compatible with the real control unit defined in DMKRIO. Please see Appendix A, "VM/SP Configuration Aid" on page 401 for a listing of real device and control unit combinations.

Examples:

```
MDISK 230 3380 5 10 WORK01 W ALL WRITE
```

is an MDISK statement for a minidisk with read/write access to 10 cylinders located on a real 3380 disk volume labeled WORK01, beginning at real cylinder 5. A user other than the owner of this minidisk can link to it in read status without specifying a read password, but must specify a password of 'WRITE' in order to gain write access to it.

```
MDISK 191 3380 50 15 CPDSK4 W RDPASS WRX2*
```

is an MDISK statement for a minidisk with read/write access to 15 cylinders located on a real 3380 labeled CPDSK4 starting at cylinder 50. A read password of RDPASS and a write password of WRX2* are provided. This lets the other users access the minidisk through the directory LINK statement (see the description of the LINK statement in this chapter) or the LINK command.

```
MDISK 190 FB-512 75100 15748 FBACMS WR READ WRITE
```

is an MDISK statement for a minidisk with write access to 15748 FB-512 blocks on the real device labeled FBACMS. If the minidisk is already accessed by another user, read-only access is provided. The minidisk begins at relative block 75100 on FBACMS.

SCREEN Control Statement

The SCREEN control statement defines the color and extended-highlight options for the user terminal. The SCREEN control statement is optional. If used, it must follow the USER control statement and precede the first device statement (CONSOLE, MDISK, DEDICATE, LINK, or SPOOL). Note that the SCREEN command is only valid when the Extended Control Feature has been applied to the terminal controller.

Format

$$\textbf{SCReen}\quad \textit{area}\ \left\{\textit{color}\ \begin{bmatrix}\textit{hilight}\\ \underline{\textbf{NONe}}\end{bmatrix}\right\}\left\{\textit{hilight}\ \begin{bmatrix}\textit{color}\\ \underline{\textbf{DEFault}}\end{bmatrix}\right\}\ \ldots\ldots$$

Parameters

area
specifies data to be highlighted or defined in color. Area may be:

ALL
entire screen. If this operand is specified, none of the following may be used:

INArea
input area.

STArea
status area.

OUTarea
output areas. If this operand is specified, none of the following may be used:

CPOut
CP output.

VMOut
virtual machine output.

INRedisp
input redisplay.

$$\left\{\begin{array}{ll}\textit{color} & \textit{hilight}\\ & \underline{\textbf{NONe}}\\ \textit{hilight} & \textit{color}\\ & \underline{\textbf{DEFault}}\end{array}\right\}$$

indicates color and extended-highlight attributes for the specified area. Values for colors are:

BLUe	blue
RED	red
PINk	pink
GREen	green
TURquois	turquoise
YELlow	yellow
WHIte	white

values for *hilight* are:

NONe	none
BLInk	blink
REVvideo	reverse video
UNDerlin	underline

One color and/or one extended-highlight attribute must be entered for each area specified. The extended-highlight and color attributes are not positional. If both are specified, either may be first.

Multiple SCREEN control statements are allowed. If you use multiple statements, they must be in a group with no other types of control statements between them.

Usage Notes

1. A default value of NONE is applied for any unspecified extended-highlight attribute. DEFAULT is used for any unspecified color attribute. The DEFAULT color is monochrome (green and white).
2. If the ALL operand is used, it must be the first operand specified on the first SCREEN control statement for the user.
3. If the OUTAREA operand is used, CPOUT, VMOUT, or INREDISP may not be specified.
4. No SCREEN control statement operands may appear more than once within a user's directory entry.
5. Terminals connected through VM/VTAM must be refreshed before any SCREEN changes can take effect.

Example:

The SCREEN control statement:

```
SCREEN OUTAREA RED NONE INAREA GREEN BLINK STATAREA PINK UNDERLIN
```

results in the following assignments:

Area	Color	Highlight
CPOUT	Red	None
VMOUT	Red	None
INREDISP	Red	None
INAREA	Green	Blink
STATAREA	Pink	Underline

SPECIAL Control Statement

The SPECIAL control statement specifies the I/O units available to the user that need not have a real I/O unit available. Special devices are program simulated devices that may or may not be connected to real or virtual devices after the user has logged on.

Format

```
SPEcial vaddr devtype [IBM|TELE]
```

Parameters

vaddr
is a one-to-three character virtual device address.

devtype
is the device type:

2701
2702
2703
3088
3138 (virtual 3138 console)
3148 (virtual 3148 console)
3158 (virtual 3158 console)
3270 (virtual 3270 only)
CTCA (channel-to-channel adapter)
TIMER (pseudo-timer device)

The preceding virtual devices created by the SPECIAL statement require a nonshared virtual control unit, except the 3270 device type.

IBM
TELE
valid only if devtype is 2701, 2702, or 2703

For example, a virtual machine running a multiple-access system that supports four IBM Type 1 adapter lines, would have four SPECIAL entries, one for each of those addresses. This provides a virtual 270x line to allow a user to dial this multiple-access system rather than logging on as a separate virtual machine.

Note: The Integrated Communications Attachment (ICA) on System/370 Models 135, 135-3, or 138 should be specified as a 2701.

SPOOL Control Statement

The SPOOL control statement specifies the unit record device that is to be spooled. Multiple readers, punches, and printers may be specified, each on a separate SPOOL card.

Format

```
Spool cuu devtype [class [ww [ll [2WCGM] [CFS] [DATCK  ]]]]
                                 [4WCGM] [BTS] [NODATCK]
```

Parameters

cuu
is the virtual device address (one-to-three hexadecimal digits). The note that follows the description of ECMODE in the OPTION control statement describes a restriction on specifying the channel. For CMS, the following unit record addresses must be used:

```
00C (reader) - 00D (punch) - 00E (printer)
```

devtype
is one of the following device types:

1403
1443
2501
2540 P
2540 R
3203
3211
3262
3289
3525
3505
3800
3800-1
3800-3
4245
4248

Notes:

1. 2540P refers to 2540 punch.
2. 2540R refers to 2540 reader.
3. If 3800 is specified, the 3800 Model 1 printer is assumed.

[*class*
is a one-character spooling class. The characters A through Z and 0 through 9 can be used. For spool input devices, a reader is the only valid device that may use an asterisk (*).

For spool output devices, the class governs the punching or printing of the real spooled output. This operand is required for all output devices defined on the spool record. If this operand is omitted, the default class A is used.

For spool input devices, the class controls access to spool files by virtual card readers. The default class for readers, an asterisk (*), means the reader can process any class of spool file.

For example,

```
SPOOL 00E 1403 A
```

specifies a SPOOL record for a virtual 1403 at address 00E. The output class is A.

[*ww*

indicates the hexadecimal width code of the paper for 3800 printers. The default is 0F (14 7/8 inches). See Table 21 for width codes.

[*ll*

indicates the decimal length of the paper in half-inches for 3800 printers. The default is 22 (11 inches).

```
┌       ┐
│ 2WCGM │
│ 4WCGM │
└       ┘
```

specifies the number of writable character generation modules (WCGM) assumed for the virtual 3800 printer. A WCGM is a 64-position portion of the 3800's character generation storage that holds the scan elements of one character set. A 3800 Model 3 can have only four WCGMs. A 3800 Model 1 can have either two or four WCGMs. 4WCGM is the default.

```
┌     ┐
│ CFS │
│ BTS │
└     ┘
```

designates the stacker assumed for the virtual 3800 Model 1 or Model 3 printer. You may specify either CFS (Continuous Form Stacker) or BTS (Burster Trimmer Stacker). If neither is specified, CFS is assumed.

Note: All parameters of the SPOOL control statement are positional except 2WCGM, 4WCGM, CFS, BTS, DATCK, and NODATCK. They must be specified in the order shown.

```
┌         ┐
│ DATCK   │
│ NODATCK │
└         ┘
```

specifies processing of certain virtual 3800 Model 1 or Model 3 data checks. If DATCK is specified, all data checks are reflected to the virtual machine (provided the 'BLOCK DATA CHECK' CCW has not been issued). If NODATCK is specified, only data checks that occur due to invalid translate table specifications or unmatched FCB codes are reflected to the virtual machine. This is the default condition.

Note: DATCK should be used only when necessary as it severely increases the overhead associated with simulation of WRITE and SKIP CCWs to the virtual 3800. In general, the reflection of data checks due to overprinting and invalid EBCDIC codes is not necessary.

Table 21 (Page 1 of 2). Available Form Width Codes

X'01'	6-1/2 in.	(165 mm ISO)
X'02'	Reserved	(180 mm ISO)
X'03'	Reserved	
X'04'	8-1/2 in.	(215 mm ISO)
X'05'	Reserved	
X'06'	9-1/2 in.	(235 mm ISO)
X'07'	9-7/8 in.	(250 mm ISO)

Table 21 (Page 2 of 2). Available Form Width Codes

X'08'	10-5/8 in.	(270 mm ISO)
X'09'	11 in.	(280 mm ISO)
X'0A'	12 in.	(305 mm ISO)
X'0B'	Reserved	(322 mm ISO)
X'0C'	Reserved	
X'0D'	13-5/8 in.	(340 mm ISO)
X'0E'	14-3/10 in.	(363 mm ISO)
X'0F'	14-7/8 in.	(378 mm ISO)

When defining devices, make sure the devices are defined (and separated) within their own control unit range, and not shared with other devices.

Note: CP does not let the user mix the two types of subchannel protocols, shared and nonshared, on a single virtual control unit. See Appendix A, "VM/SP Configuration Aid" on page 401 for the listing of the protocols used by specific devices.

Sample Directory Entries

The following sample VM/SP directory entries provide you with some of the virtual machines necessary for system operation and updating. The indentations are for readability and are not required by the directory program. LINK control statements are used whenever possible to reduce the number of changes to the VM/SP directory whenever a minidisk extent is moved. A brief explanation of some of the virtual machine user IDs and system areas follows.

Warning: The passwords in these sample entries are restricted. You must substitute your own passwords. If you copy any sample password, it will be changed to NOLOG when the directory program is run. That virtual machine will not be able to log on. For more information, see "Passwords That Do Not Work" on page 221.

A Hardware Service Virtual Machine (EREP)

The following directory entry defines the virtual machine (EREP) that can be used by the hardware service representative. This virtual machine usually has class F command privileges. See the *VM/SP OLTSEP and Error Recording Guide* for more information on the hardware service virtual machine.

```
USER EREP NOLOG 1M 2M FG
 ACCOUNT EREP IBMCE
 IPL CMS
 CONSOLE 01F 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19D 19D RR
 LINK MAINT 19E 19E RR
 LINK MAINT 201 192 RR
 MDISK 191 3380 027 001 VMSRES WR READ     WRITE
```

System Operator's Virtual Machine (OPERATOR)

The user ID for this directory entry must be the same as the user ID on the SYSOPER operand of the SYSOPR system generation macro in the DMKSYS file. The USER control statement gives the operator all command privilege classes except class F. Actually, if other virtual machines are defined with command privilege classes appropriate for updating VM/SP, the operator's virtual machine needs only class A command privileges. The MDISK control statement defines the 191 minidisk, which contains CMS files, EXEC procedures, and service programs to update VM/SP.

```
USER OPERATOR OPERATOR  3M 16M ABCDEG
 ACCOUNT 2 OPERATOR
 CONSOLE 009 3215 T MAINT
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19D 19D RR
 LINK MAINT 19E 19E RR
 MDISK 191 3380 040 001 VMSRES MR ROPER WOPER MOPER
```

A Virtual Machine to Receive System Dumps (OPERATNS)

The user ID for the following directory entry is the user ID specified on the SYSDUMP operand of the SYSOPR macro in the DMKSYS file. All abnormal termination dumps are sent to this virtual machine. This user usually is given command privilege classes B, C, E, and G. If the directory entry contains all disks usually attached to the system, described as full-volume minidisks, you can rewrite the VM/SP directory, using the DIRECT command. The operations group can examine any disk while it is attached to the system, when these disks are defined as full-volume minidisks.

```
USER OPERATNS NOLOG 1M  2M BCEG
 ACCOUNT 13 SYSPROG
 IPL CMS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
 LINK MAINT 19D 19D RR
 LINK MAINT 19E 19E RR
 MDISK 191 3380 168 001 VMSRES MR RIPCS    WIPCS    MIPCS
 MDISK 193 3380 249 008 VMSRES MR RIPCS    WIPCS    MIPCS
```

Other System Virtual Machines

In addition to virtual machines discussed to this point, there are virtual machines that:

- Support and update the VM/SP system
- Test new releases of the system before placing them into production
- Provide other users with a remote file spooling capability.

A Virtual Machine for Updating and Supporting VM/SP (MAINT)

The following directory entry defines a virtual machine (MAINT) that can support and update the VM/SP system. The MAINT virtual machine's command privilege classes include class E and class G, so it can issue the SAVESYS command to save CMS and other systems.

```
USER MAINT CPCMS 16M 16M ABCDEFG
 ACCOUNT 1 SYSPROG
 OPTION ECMODE DIAG98
 IPL 190
 CONSOLE 009 3215
 SPOOL 00C 2540 READER *
 SPOOL 00D 2540 PUNCH  A
 SPOOL 00E 1403 A
 MDISK 096 3380 842 006 VMPK01 MW ALL      WMAINT   MMAINT
 MDISK 123 3380 000 885 VMSRES MW RSYSRES  WSYSRES  MSYSRES
 MDISK 124 3380 000 885 VMPK01 MW RSYSRES  WSYSRES  MSYSRES
 MDISK 127 3380 000 885 VMPK04 MW RSYSRES  WSYSRES  MSYSRES
 MDISK 129 3380 000 885 PROFPK MW RSYSRES  WSYSRES  MSYSRES
 MDISK 130 3380 000 885 SQLPK  MW RSYSRES  WSYSRES  MSYSRES
 MDISK 19D 3380 488 041 VMPK01 MW ALL      WMAINT   MMAINT
 MDISK 190 3380 533 078 VMSRES MW ALL      WMAINT   MMAINT
 MDISK 191 3380 158 010 VMSRES MW RMAINT   WMAINT   MMAINT
 MDISK 192 3380 611 005 VMSRES MW RMAINT   WMAINT   MMAINT
 MDISK 193 3380 114 044 VMSRES MW RMAINT   WMAINT   MMAINT
 MDISK 194 3380 048 033 VMSRES MW RMAINT   WMAINT   MMAINT
 MDISK 195 3380 747 007 VMPK01 MW RMAINT   WMAINT   MMAINT
 MDISK 196 3380 028 020 VMSRES MW RMAINT   WMAINT   MMAINT
 MDISK 201 3380 777 023 VMSRES MW RMAINT   WMAINT   MMAINT
 MDISK 293 3380 800 042 VMSRES MW RCMSAUX  WCMSAUX  MCMSAUX
 MDISK 294 3380 862 021 VMSRES MW RCPAUX   WCPAUX   MCPAUX
 MDISK 295 3380 259 027 VMSRES MW RUSRMOD  WUSRMOD  MUSRMOD
 MDISK 296 3380 070 020 VMPK01 MW RCPAUX   WCPAUX   MCPAUX
 MDISK 319 3380 021 006 VMSRES MW ALL      WMAINT   MMAINT
 MDISK 392 3380 740 007 VMPK01 MW ALL      WMAINT   MMAINT
 MDISK 393 3380 336 076 VMPK04 WR RMAINT   WMAINT
 MDISK 394 3380 412 076 VMPK04 WR RMAINT   WMAINT
 MDISK 395 3380 616 005 VMSRES WR RMAINT   WMAINT
 MDISK 396 3380 495 042 VMPK04 WR RMAINT   WMAINT
 MDISK 59E 3380 832 010 VMPK01 MW RMAINT   WMAINT   MMAINT
 MDISK 595 3380 709 031 VMPK01 MW RMAINT   WMAINT   MMAINT
 MDISK 596 3380 848 006 VMPK01 MW RMAINT   WMAINT   MMAINT
```

Minidisks Reserved for the MAINT User ID

The following table lists the minidisks reserved for the MAINT user ID. All minidisks except those marked with an asterisk (*) are defined in the sample system directory.

Table 22 (Page 1 of 3). Minidisks Reserved for MAINT

Address	Volume	Used By	Function/Contents
096	VMPK01	HPO	Apply disks (auxiliary control files and update shells)
190	VMSRES	CMS	System disk (MODULEs, EXECs, MACLIBs, TXTLIBs, stand-alone service programs, some updated text decks)

Table 22 (Page 2 of 3). Minidisks Reserved for MAINT			
Address	**Volume**	**Used By**	**Function/Contents**
191	VMSRES	CMS	Installation tools and profiles; work area
192	VMSRES	CP	Apply disk (auxiliary control files and update shells)
193	VMSRES[1]	CMS Procedures Language/VM IPCS GCS TSAF AVS	Base disk (system generation and maintenance tools, updated text decks); merge disk for preventive service Base disk (text decks); merge disk for preventive service Base disk (text decks); merge disk for preventive service IPCS interface files IPCS interface files IPCS interface files
194	VMSRES	CP	Base disk (text decks); merge disk for preventive service
195	VMPK01[2]	CMS Procedures Language/VM IPCS	Receive disk for corrective service Receive disk for corrective service Receive disk for corrective service
196	VMSRES	HPO	Text decks
19D	VMPK01[2]		HELP files
19E[3]		CMS	Extension disk used by some optional feature program products
201	VMSRES[1]	EREP	TXTLIBs
293	VMSRES[2]	CMS Procedures Language/VM IPCS	Receive disk for preventive service (PTFs; auxiliary and update files; updated TEXT decks; replacement CNTRL, EXEC, MODULE, and XEDIT files) Receive disk for preventive service (serviced text decks and update files) Receive disk for preventive service (replacement CNTRL, EXEC, MODULE, and XEDIT files)
294	VMSRES[4]	CP	Receive disk for preventive service (PTFs; auxiliary and update files; ASSEMBLE, MACRO, COPY, and source files; updated TEXT decks, MACRO files, and COPY files; replacement CNTRL, EXEC, MODULE, and XEDIT files)
295	VMSRES	CP CMS Procedures Language/VM IPCS	System definition files; merge disk for corrective service Nucleus generation profile; merge disk for corrective service Merge disk for corrective service Merge disk for corrective service
296	VMPK01	HPO	Service files
319	VMSRES[1]		Licensed program "Memo to Users"
392	VMPK01[2]	CMS Procedures Language/VM IPCS	Apply disk Apply disk Apply disk
393	VMPK04	CMS IPCS	Source code disk (ASSEMBLE, $EXEC, $XEDIT files) Source code disk
394	VMPK04	CP	Source code disk (ASSEMBLE, MACRO, COPY files)
395	VMSRES	CP	Receive disk for corrective service
396	VMPK04	HPO	Source code disk
595	VMPK01[5]	GCS	System disk
596	VMPK01	HPO	Receive disk for corrective service

Table 22 (Page 3 of 3). Minidisks Reserved for MAINT			
Address	**Volume**	**Used By**	**Function/Contents**
59E	VMPK01[2]	GCS	Extension disk for optional GCS applications

Notes:

[1]VMPK01 on 9313 and 9332 DASD

[2]VMPK02 on 9313 DASD

[3]Not defined in sample CP directory

[4]VMPK01 on 9313 DASD

[5]VMSRES on 9332 DASD; VMPK02 on 9313 DASD.

Directory Entries for CMS/DOS

The VSE system and private libraries are accessed in read-only mode under CMS/DOS. If more than one CMS virtual machine is using CMS/DOS, you should update the VM/SP directory entries so the VSE system residence volume and the VSE private libraries are shared by all CMS/DOS users.

The VM/SP directory entry for one CMS virtual machine should contain MDISK statements defining VSE volumes. VM/SP directory entries for other CMS/DOS users should contain LINK statements.

For example, assume the VSE system libraries are on cylinders 0-149 of a 3330 volume labeled DOSRES. Also, assume the VSE/AF private libraries are on cylinders 0 to 99 of a 3330 volume labeled DOSPRI. Then one CMS machine (for example, DOSUSER1) would have the MDISK statements in its directory entry.

```
USER DOSUSER1 password 1M 2M G
     .
     .
     .
   MDISK 331 3330 0 150 DOSRES R rpass
   MDISK 231 3330 0 100 DOSPRI R rpass
```

All other CMS/DOS users would have links to these disks. For example:

```
LINK DOSUSER1 331 331 R rpass
LINK DOSUSER1 231 231 R rpass
```

Chapter 8. Preparing the Real I/O Configuration File (DMKRIO)

Contents of Chapter 8

Overview of DMKRIO

DMKRIO (Real I/O) is a CP System Control file. This file, along with DMKSYS and the System Directory, defines the VM/SP system that is generated. It consists of macros that describe the I/O devices, control units, and channels attached to the real processor. VM/SP uses this information to schedule I/O operations and to allocate resources. Therefore, the real I/O macro entries must represent the real hardware configuration accurately. Generally, there must be one real I/O macro entry for each hardware unit in your configuration.

You can include entries for more devices than you have so devices can be added in the future without doing another system generation. Remember that the control blocks generated (RDEVBLOK, RCUBLOK, and RCHBLOK) occupy space in real storage.

For the 3081 Processor Complex, in addition to preparing the Real I/O configuration file, you must prepare the Input/Output Configuration Program source file and run the Input/Output Configuration Program to define the I/O configuration to the processor. See "Considerations for Coding the Input/Output Configuration Program Source File" later in this chapter, for more information.

When preparing the RDEVICE and RCTLUNIT entries, see Appendix A, "VM/SP Configuration Aid" on page 401 to assist you in configuring control units and devices. Following the descriptions of the CLUSTER, TERMINAL, RDEVICE, RCTLUNIT, RCHANNEL, and RIOGEN macros, there is an example showing how these macros are coded for one particular real configuration.

The file should be created in the following order:

```
DMKRIO CSECT        Units Referred To:
       CLUSTER macro      Remote Display Stations
       TERMINAL macro
             .
             .
       RDEVICE macros     I/O Devices
             .
             .
       RCTLUNIT macros    Control Units
             .
             .
       RCHANNEL macros    Channels
             .
             .
       RIOGEN macro       System Console
       END
```

Note: There must be a CLUSTER macro for each 3270 control unit for remote 3270s. Each CLUSTER macro must be followed immediately by the TERMINAL macros representing each display station and printer on that control unit. The CLUSTER and TERMINAL macro groups must come before all other real I/O configuration macros. See special requirements for TERMINAL macros for devices attached to the 3274 Model 1C under "Coding the Real I/O Configuration Macros for Remote 3270s."

All groups of CLUSTER and TERMINAL macros must appear first, followed by all RDEVICE macros, all RCTLUNIT macros, all RCHANNEL macros, and finally by the RIOGEN macro. In addition, the first statement in the file must be the DMKRIO CSECT statement (as shown) and the last statement must be the assembler END statement. The Ethernet or Token Ring LAN subsystems for a TSAF virtual machine must be generated as 3088 devices to CP by specifying RDEVICE, RCTLUNIT, and RCHANNEL macro entries in DMKRIO.

Coding the Real I/O Configuration Macros for Remote 3270s

Two types of remote 3270 configurations are supported: a cluster control unit with multiple terminals and printers attached and stand-alone display stations. The clustered configurations attach to either a 3271, 3274 Model 1C, or 3276 control unit. The stand-alone station is a 3275 display station that contains its own built-in control unit. All remote configurations are attached through binary synchronous communication lines.

To define remote 3270 stations you must code CLUSTER, TERMINAL, and RDEVICE macros. Code one RDEVICE macro for each binary synchronous line that supports a remote 3270 configuration. Code one CLUSTER macro to define the 3270 control unit for each of those lines and code one or more TERMINAL macros, as needed, to define the devices in the remote 3270 configuration.

The CLUSTER macro defines the control unit (3271, 3274 Model 1C, 3275, or 3276) for the remote 3270 configuration. Each CLUSTER macro must have a different label. This label is coded on the RDEVICE macro that defines the corresponding binary synchronous line and logically links the line and the cluster. The address of the line (defined by the ADDRESS=cuu operand of the RDEVICE macro) is coded in the LINE=cuu operand of the CLUSTER macro.

Follow each CLUSTER macro with the TERMINAL macros that define the terminals for the remote 3270 control unit. For the 3271 and 3276 directly following the CLUSTER macro, code a TERMINAL macro for each terminal address to which a terminal can be attached (regardless of whether or not the intermediate addresses are unused). For example, if terminals are attached to the third, fourth, and eighth addresses, you code eight TERMINAL macros. The first macro represents the first (lowest) address, the last represents the eighth (highest) address.

For the 3274 Model 1C that has only 3278s, 3279s 3290s (attached through Terminal Adapter Types A1, A2, or A3), 3287s, or 3289s attached, follow the same procedure as for the 3271 and 3276 in coding the TERMINAL macros. If the 3274 Model 1C has 3277s, 3284s, 3286s, 3287s (attached through Terminal Adapter Types B1, B2, B3, or B4), or 3288s attached, directly following the CLUSTER macro, first code TERMINAL macros for all 3278s, 3279s, 3287s, 3290s (attached through Terminal Adapter Types A1, A2, or A3), and 3289s. These devices must occupy the first 8, low-order addresses, and each following block of 8 addresses until all of these devices are attached. As before, a TERMINAL macro must be coded for all unused addresses in each block of 8 addresses that are required. Immediately following the last TERMINAL macro in the block of 8, 16, or 24, code a TERMINAL macro for each 3277, 3284, 3286, 3287 (attached through Terminal Adapter Types B1, B2, B3, or B4), and 3288 that can be attached. These devices will occupy the higher-order addresses on the controller. Again, a TERMINAL macro must be coded for each unused address to which a terminal can be attached up to the last address occupied.

For the 3275, directly following the CLUSTER macro, code a single TERMINAL macro specifying TERM=3275. If the 3275 has a 3284 or 3286 Model 3 Printer attached, specify MODEL=3 to define the printer; otherwise, the printer is ignored.

After all CLUSTER-TERMINAL groups of macros have been coded, code the other real I/O configuration macros. You must code an RDEVICE macro for each binary synchronous line that supports remote 3270 stations. Specify the label of the corresponding CLUSTER macro on the RDEVICE macro (CLUSTER=label).

CLUSTER Macro

The CLUSTER macro defines a control unit associated with a remote 3270. Each CLUSTER macro represents a display control unit (a 3271, 3274 Model 1C, or 3276) on a leased BSC line, or a stand-alone 3275 on either a switched or leased BSC line. One CLUSTER macro must be specified for each 3271, 3274 Model 1C, 3275, and 3276.

Note: Each CLUSTER macro must immediately precede the TERMINAL macros defining the devices attached at each remote 3270 station. The groups of CLUSTER and TERMINAL macros must come before all other macros in the DMKRIO file.

Format

label	**CLUSTER**	**CUTYPE =** $\left\{\begin{array}{l}\textbf{3271}\\\textbf{3274}\\\textbf{3275}\\\textbf{3276}\end{array}\right\}$ **,GPOLL =** *cudv* **,LINE =** *cuu* **,DIAL =** $\left\{\begin{array}{l}\textbf{YES}\\\textbf{NO}\end{array}\right\}$

Parameters

label
is a name of the CLUSTER macro. It must be specified. The label may be any assembler language symbol. The label establishes a special symbolic name for this cluster control unit or stand-alone station.

CUTYPE =
is the station control unit. It is either 3271, 3274 Model 1C, 3275, or 3276.

GPOLL =
are the general polling characters that represent the general polling technique to be used for this station. When general polling is used, the first device ready to send data over the line is allowed to do so. The characters, cudv, are the 4-digit hexadecimal general polling characters assigned to the station control unit. The hexadecimal equivalent of the EBCDIC transmission code is in the form cudv.

where:

cu
are the polling characters for the control unit.

dv
are the characters for any available input device.

The general polling characters for a remote 3270 device (dv) are always X'7F' and the general polling characters for the control unit are defined when the control unit is installed. Use Table 23 on page 289 to determine what you should code as the general and specific polling characters for the control unit. GPOLL is ignored if CUTYPE=3275 and DIAL=YES are specified.

Note: The 3274 and 3276 terminal control unit address switches are set by you to match polling and selection address characters shown in Table 23 on page 289. Note also that this table is very general.

LINE=
is the line interface address. It is the address specified on the RDEVICE macro associated with this CLUSTER macro.

DIAL=
specifies whether the 3275 has the Dial feature.

Examples:

The following CLUSTER macro describes a 3271 control unit with a control unit address of 2 and a line address of 078.

```
CLUST001 CLUSTER CUTYPE=3271,GPOLL=C27F,LINE=078,DIAL=NO
```

The following CLUSTER macro describes a 3275 display station (without the Dial feature) that has a control unit address of 0 and a line address of 080.

```
CLUST020 CLUSTER CUTYPE=3275,GPOLL=407F,LINE=080,DIAL=NO
```

In the Real I/O configuration file (DMKRIO), the CLUSTER macro must immediately come before TERMINAL macros that define stations attached to that cluster or stand-alone station.

TERMINAL Macro

The TERMINAL macro defines the following:

- A display station or printer that is attached to the remote 3270 display system or
- A terminal address that is available to attach an additional remote 3270.

Each terminal address attached to a cluster must be represented by a TERMINAL macro. Only one TERMINAL macro is specified for a stand-alone 3275 display station.

Code one TERMINAL macro for each display device and each 5K printer attached to a cluster control unit (3271, 3274 Model 1C, or 3276). You must code a TERMINAL macro for every terminal address to which a terminal can be attached, even if a terminal address is unused. When you code a TERMINAL macro for an unused terminal address, specify a valid TERM= operand and the correct selection or addressing characters. An adapter card position must be present in the control unit for any terminal address generated, whether a terminal is physically attached or not. Failure to meet this requirement will result in timeouts with remote device type 3274 and 3276 control units.

For a 3274 Model 1C Control Unit that has 3277s, 3284s, 3286s, 3287s (attached through terminal Adapter Types B1, B2, B3, or B4), or 3288s attached, code a TERMINAL macro for all 3278s, 3279s, 3287s, 3289s, and 3290s in groups of eight until all 3278s, 3279s, 3287s, 3289s, and 3290s have been included. You must code a TERMINAL macro for every terminal address in each group of eight. Following these macros, code a TERMINAL macro for each 3277, 3284, 3286, 3287, or 3288. Again, you must code a TERMINAL macro for every terminal address to which a terminal can be attached.

Code only one TERMINAL macro to define the display station, and optionally a printer, attached to a stand-alone station (3275). Because the 3276 is a cluster controller and not a stand-alone, code each 3276 with a TERMINAL macro. Code TERM=3275 to define the 3275 display station and, optionally, code MODEL=3 to define a 3284 or 3286 printer attached to the 3275.

Format

```
label     TERMINAL   TERM= ⎧3275⎫
                           ⎪3276⎪
                           ⎪3277⎪
                           ⎪3278⎪
                           ⎪3279⎪
                           ⎨3284⎬
                           ⎪3286⎪
                           ⎪3287⎪
                           ⎪3288⎪
                           ⎪3289⎪
                           ⎩3290⎭
                     ,SELECT=cudv
                     ⎡,MODEL= ⎧1⎫⎤
                     ⎢        ⎪2⎪⎥
                     ⎢        ⎨3⎬⎥
                     ⎢        ⎪4⎪⎥
                     ⎣        ⎩5⎭⎦
                     [,FEATURE=OPRDR]
```

Note: All TERMINAL macros defining devices attached to a remote 3270 station must follow the CLUSTER macro that defines the control unit for that station. Groups of CLUSTER and TERMINAL macros must come before all other macros in the DMKRIO file.

Parameters

label
is any desired label.

TERM=
is the device type of the remote 3270 station attached to the clustered or stand-alone 3270 control unit. If TERM= 3276, 3278, or 3279, MODEL= *must* be specified.

SELECT=
are the 4-digit hexadecimal selection or addressing characters assigned to this device, where:

cu
are the characters for the control unit

dv
are the characters for the device

Use Table 23 on page 289 to determine the selection and addressing characters for this device. The SELECT operand is ignored if DIAL=YES is specified for the 3275 in the CLUSTER macro.

Note: If a printer is attached to the 3275, it has the same address as the 3275 display station.

MODEL=
is the model number of the terminal or printer. The default is model 2.

Note: If TERM= 3276, 3278, or 3279, MODEL= *must* be specified, and should be equal to the actual model of the real device. If the model specification does not match the real device, unpredictable results may occur. If TERM=3290, model number should not be specified.

The following is a list of terminals and their model numbers:

Terminal No.	Approximate Model No.
3275	2
3276	2, 3, or 4
3277	2
3278	2, 3, 4, or 5
3279	2 or 3
3284	2 or 3
3286	2 or 3
3287	1 or 2
3288	2
3289	1 or 2
3290	

Note: If TERM= 3276, 3278, or 3279, the model number 2, 3, 4, or 5 must be specified.

The following IBM printers can be attached to *both* the 3271 and 3274 Model 1C cluster control units:

Printer No.	Approximate Model No.
3262	1 and 2
3284	2
3286	2
3287	1 and 2
3288	2

Note: The IBM 3289 printer (models 1 and 2) can *only* be attached to a 3274 Model 1C cluster control unit and *not* the 3271 cluster control unit.

The following IBM printers can be attached to a 3276 cluster control unit:

- 3287 Models 1 and 2
- 3289 Models 1 and 2.

The following IBM printers can be attached to a stand-alone 3275 station:

- 3284 Model 3
- 3286 Model 3 (via RPQ MB4317).

FEATURE=
specifies the optional operator identification card reader feature, available on the 3277 Display Station, Model 2, or the magnetic slot reader on a 3276, 3278 Display Station, Models 2, 2A, 3, 4, and 5, or 3279 Color Display Station, Models 2A, 2B, 3A, and 3B.

Examples:

Example 1: This TERMINAL macro describes a 3277 with a selection address of 2, and a device address of 2.

```
TERMINAL TERM=3277,SELECT=E2C2,FEATURE=OPRDR
```

Example 2: This TERMINAL macro describes a 3286 with a selection address of 3 and a device address of 3.

```
TERMINAL TERM=3286,SELECT=E3C3
```

Example 3: This TERMINAL macro describes a 3284 with a selection address of 4 and a device address of 4.

```
TERMINAL TERM=3284,SELECT=E4C4,MODEL=2
```

Example 4: This TERMINAL macro describes a 3275 Display Station with a 3284 Printer, Model 3, attached and a device address of 0.

```
TERMINAL TERM=3275,SELECT=6040,MODEL=3
```

If no printer is attached to the 3275, code:

```
TERMINAL TERM=3275,SELECT=6040
```

Table 23. Remote 3270 Control Unit and Device Addressing

Use this column for: • Device selection • Specific poll • General poll • Fixed return addresses		Use this column for: • 3270 control unit selection addresses	
If the Control Unit or Device Number is:	The EBCDIC Code (in hexadecimal) is:	If the Control Unit Number is:	The EBCDIC Code (in hexadecimal) is:
0	40	0	60
1	C1	1	61
2	C2	2	E2
3	C3	3	E3
4	C4	4	E4
5	C5	5	E5
6	C6	6	E6
7	C7	7	E7
8	C8	8	E8
9	C9	9	E9
10	4A	10	6A
11	4B	11	6B
12	4C	12	6C
13	4D	13	6D
14	4E	14	6E
15	4F	15	6F
16	50	16	F0
17	D1	17	F1
18	D2	18	F2
19	D3	19	F3
20	D4	20	F4
21	D5	21	F5
22	D6	22	F6
23	D7	23	F7
24	D8	24	F8
25	D9	25	F9
26	5A	26	7A
27	5B	27	7B
28	5C	28	7C
29	5D	29	7D
30	5E	30	7E
31	5F	31	7F

Note: When using this table, look up the control unit address setting in the left hand column and append X'7F'.

Notes:

1. VM only supports one controller per line.
2. A maximum of 32 devices can attach to a 3274.

Example 1 (4 devices)

```
DMKRIO    CSECT
CLUST01   CLUSTER   CUTYPE=3274,GPOLL=407F,LINE=78
          TERMINAL  TERM=3278,SELECT=6040,MODEL=2
          TERMINAL  TERM=3278,SELECT=60C1,MODEL=2
          TERMINAL  TERM=3278,SELECT=60C2,MODEL=2
          TERMINAL  TERM=3287,SELECT=60C3,MODEL=2
```

Example 2 (12 devices)

```
DMKRIO    CSECT
CLUST02   CLUSTER   CUTYPE=3274,GPOLL=407F,LINE=78
          TERMINAL  TERM=3278,SELECT=6040,MODEL=2
          TERMINAL  TERM=3278,SELECT=60C1,MODEL=2
          TERMINAL  TERM=3278,SELECT=60C2,MODEL=2
          TERMINAL  TERM=3287,SELECT=60C3,MODEL=2
          TERMINAL  TERM=3278,SELECT=60C4,MODEL=2
          TERMINAL  TERM=3278,SELECT=60C5,MODEL=2
          TERMINAL  TERM=3278,SELECT=60C6,MODEL=2
          TERMINAL  TERM=3287,SELECT=60C7,MODEL=2
          TERMINAL  TERM=3278,SELECT=6048,MODEL=2
          TERMINAL  TERM=3278,SELECT=6049,MODEL=2
          TERMINAL  TERM=3278,SELECT=604A,MODEL=2
          TERMINAL  TERM=3287,SELECT=604B,MODEL=2
```

How CLUSTER and TERMINAL Macros Affect Addressing Bisync Terminals and Printers

Now that we have discussed the CLUSTER and TERMINAL macros, let's discuss the effect these macros have on the addressing of remote bisync terminals and printers. For example, if the following macros were coded as such:

```
CLUST022 CLUSTER CUTYPE=3274,GPOLL=407F,LINE=022,DIAL=NO
         TERMINAL TERM=3278,SELECT=6040,MODEL=4
         TERMINAL TERM=3279,SELECT=60C1,MODEL=2
         TERMINAL TERM=3279,SELECT=60C2,MODEL=3
CLUST023 CLUSTER CUTYPE=3274,GPOLL=C17F,LINE=023,DIAL=NO
         TERMINAL TERM=3287,SELECT=6140,MODEL=2
         TERMINAL TERM=3278,SELECT=61C1
         TERMINAL TERM=3278,SELECT=61C2
CLUST024 CLUSTER CUTYPE=3274,GPOLL=407F,LINE=024,DIAL=NO
         TERMINAL TERM=3278,SELECT=6040,MODEL=4
         TERMINAL TERM=3279,SELECT=60C1,MODEL=2
         TERMINAL TERM=3279,SELECT=60C2,MODEL=3
```

The addressing for remote bisync terminals and printers depends on the sequence that they are coded in the DMKRIO. The first CLUSTER macro generates a resource *id* of 00nn. The TERMINAL macros under CLUST022 would then have resource IDs of 0001, 0002, and 0003 respectively. The second CLUSTER macro and set of TERMINALS would have resource IDs of 0101, 0102, and 0103. The third set would generate resource IDs of 0201, 0202, and 0203. The first two numbers of the resource ID identify the CLUSTER number. The range is X'00' to X'FF'. The second two numbers identify the terminal resource attached to the CLUSTER. The range for this number is X'01' to X'FF'.

RDEVICE Macro

The RDEVICE macro generates a real device block (RDEVBLOK). You must code an RDEVICE macro for each real I/O device in your I/O configuration. The algorithm to calculate the maximum number of devices is:

(X'8000' * X'10' / (RDEVSIZE * 8)

If there is a remainder, 1 should be added to the quotient.

RDEVICE macro instructions describe each device, or group of devices, attached to your processor. These can be in any order (except when used in conjunction with the CLUSTER macro[26]). They must be contiguous and must come before all RCTLUNIT and RCHANNEL macros in the Real I/O configuration file (DMKRIO). Also, RDEVICE macro instructions must follow all groups of CLUSTER and TERMINAL macros, if any. The first RDEVICE macro generates the label DMKRIODV, which indicates the start of real device blocks to CP.

The name field may not be specified for the RDEVICE macro instruction. If a name is specified it is ignored. The RDEVICE macro generates a name by appending the device address to the characters RDV. For example, the name RDV234 is generated for the device address 234.

Before you code an RDEVICE macro for a 3704, 3705, or 3725 device, see "Special Considerations for Coding the 3704/3705/3725 RDEVICE Macro" on page 306.

The RDEVICE macro statement is not used for SNA supported terminals. A local 3274-41A SNA control unit, however, may be generated using the RDEVICE macro if you code the RDEVICE macro as either a 3705 or a 3725 device, for example:

```
RDEVICE ADDRESS=nnn,DEVTYPE=3705,ADAPTER=TYPE4,MODEL=H8
```

You must then code the corresponding RCTLUNIT macro to match this RDEVICE macro.

Note that on an MP processor, such as the 4381, you cannot use the VM utilities DDR or Format/Allocate in a *stand-alone* mode (bare machine). This happens if all the involved devices are not configured on the IPL processor.

[26] See "Planning for 3270s" on page 189 before you code an RDEVICE macro for a binary synchronous line used by remote 3270s.

Format

```
label     RDEVICE    ADDRESS={cuu     } ,DEVTYPE=type [,MODEL=model]
                             {(cuu,nn)}

                     [,FEATURE=(feature[,feature]...)]

                     [,CLASS= {(cl[,cl]...)}]
                     [        {DASD        }]
                     [        {TAPE        }]
                     [        {TERM        }]
                     [        {GRAF        }]
                     [        {URI         }]
                     [        {URO         }]

                     [          {BSCA }]      [,CPTYPE= {EP }]
                     [          {IBM1 }]      [         {NCP}]
                     [          {SDLC }]      [         {PEP}]
                     [,ADAPTER= {TELE2}]
                     [          {TYPE1}]
                     [          {TYPE2}]
                     [          {TYPE3}]      [,ALTCU= cuu]
                     [          {TYPE4}]
                     [          {TYPE5}]

                     [,SETADDR=sadnum]

                     [,CPNAME= cpname]

                     [,BASEADD=cuu]

                     [,CLUSTER=label]

                     [,IMAGE=imagelib]

                     [,CHARS=ffff]

                     [,FCB=lpi]

                     [,DPMSIZE= n]
```

Parameters

label
is any desired label.

ADDRESS=
is the real I/O device address (or addresses).

cuu is three-hexadecimal digits from 000 to FFF. The high-order digit is the address of the channel to which the device is attached. The low-order two digits represent the control unit and device address.

nn is the number of RDEVBLOK entries to be generated. It may be any number from 001 to 256. For example, if ADDRESS=(100,5) is specified, RDEVBLOKs with device addresses 100, 101, 102, 103, and 104 are generated. If *nn* is omitted, a value of 1 is assumed for all devices except the 2305, which has a default value of 8. For a 2305, the last character of *cuu* should be 0 or 8. The maximum value of *nn* is 16.

If DEVTYPE=3066, 3138, 3148, or 3158, or if DEVTYPE=3278 and Model=2A, *nn* can only be 1. This is because only one system display console can be specified for each RDEVICE macro.

Every installed 3705 and/or 3725 must have the following:

- A valid RDEVICE macro that describes the device characteristics and the device address
- A valid RCTLUNIT macro covering the address specified in the RDEVICE macro.

If each installed 3705 and/or 3725 is not coded this way, CP will consider each device to be a 27xx line, not a 37x5 device.

DEVTYPE=
is the type of device. For a list of possible device types that IBM recommends see Appendix A, "VM/SP Configuration Aid" on page 401.

Notes:

1. When specifying a 3262 Model 3 or 13, specify the 3289 with no model number.
2. A 3262 Model 1 or 2 may only be generated on channel 0 and a 3262 Model 5 can be generated on any channel (no restrictions).
3. If specifying a device type of 3380 for the 3380 Model CJ2, when RDEVICE is processed, it is not possible for RDEVICE to determine that the control unit is a 3380 Model CJ2. Similarly, it is not possible for RCTLUNIT to determine RDEVICE parameters while RCTLUNIT is processing input. As a consequence, these parameters are not validated for the 3380 Model CJ2 when DMKRIO is assembled.

Coding Considerations

Additional information relating to the support of HFGD (High Function Graphic Device) devices can be found in the *Graphics Access Method/System Product General Information Manual*, GC33-0125. During device coding, remember to specify a 5080 as a HFGD.

For TWX terminals, 3101 display terminals, or 3232 keyboard terminals, specify 270x as the device type and ADAPTER=TELE2. Remote terminals such as a 2741 or a 3767 must be coded as a 2701, 2702, 2703, 3704, 3705, or 3725. For a 3350 device in native mode, specify 3350 as the device type. For a 3350 being used in 3330 compatibility mode, specify 3330. Specify a 3344 disk as a 3340, and a 3333 as a 3330. Specify a 3250 device as a 2250. Specify a 3230 and a 3268 as a 3287-2. An MSS 3330V device address must be defined as DEVTYPE=3330 with one of the two FEATURE= operands allowed. See the explanation of the FEATURE operand that follows.

For 3287 printers attached through a 3272 Control Unit Model 2, specify DEVTYPE= 3284 or 3286.

For a 3289 Model 4 to be attached to a 4331 Display Printer Adapter as a system printer, specify DEVTYPE= 3289E, or DEVTYPE= 3289, MODEL= 4.

For a locally attached 3290 Information Panel you may specify either of the following:

1. DEVTYPE=3290

   In this case, MODEL= should not be specified. For 3278 compatibility mode, VM assumes that this is a 3278-2. Note that for many applications, you should define one logical terminal of the 3290 as a 3278-2 before logging onto the system.

2. DEVTYPE=3278

   In this case, MODEL= is required.

For either a 3178 or 3191 display, specify DEVTYPE=3278,MODEL=2. For either a 3180 or 3192 (Monochrome model) display, specify DEVTYPE=3278. Also, for either a 3179 or 3192 color display specify either DEVTYPE=3279 or DEVTYPE=3278.

For a remotely attached 3290, you should specify DEVTYPE=3278 and specify the MODEL= operand. See "Miscellaneous Restrictions" on page 424 for more information about defining the 3290.

To support the 4248-1 printer in 3211 compatibility mode, specify 3211 as the device type. To support the 4224 printer, specify 3287 as the device type.

For 3179 and 3180 terminals, specify 3279 MODEL=2 as the device type. If you have a 3179-G terminal, specify 3279 MODEL=3 as the device type. For a 3178 terminal, specify 3278 as the device type. Also, the PC/AT/370, PC/XT/370, and PS/2 should be specified as a 3277, 3278, or 3279 device types. This depends on which card the PC terminal is using.

Because a CTCA can tie up a channel, it is recommended that you generate only one per channel. If other devices are to be attached to the same channel as a CTCA, they should be noncritical devices such as readers or printers.

System consoles must be defined using RDEVICE macros. The system console types listed on the right should be coded for the processor types listed on the left. See Table 24. The RIOGEN macro should also be coded to correspond to the defined system console address. Specify the system console in both macros as follows:

Table 24 (Page 1 of 2). System Console and Processor Types

Processor	**System Console Specified**
135, 135-3, 145, 145-3, 155 II	3210 or 3215
138	3138 (if in display mode) 3215 (if in printer-keyboard mode)
148	3148 (if in display mode) 3215 (if in printer-keyboard mode)
158	3158 (if in display mode) 3215 (if in printer-keyboard mode and has the 3213 Printer Model 1)
165 II, 168	3066

Table 24 (Page 2 of 2). System Console and Processor Types

Processor	System Console Specified
3031, 3032, 3033-N 3033-S, 3042	3036
4331, 4341, 4361, 4381	3278 Model 2A (if in display mode) 3279 Model 2C (if in display mode) 3215 (if in printer-keyboard mode)
3081	3278 Model 2A (if in display mode)
9373 Model 20 9375 Models 40 and 60 9377 Model 90	3278 Model 2 (if in display mode)

Device types 2540R and 2540P refer to the same 2540 Card Read Punch (as do 2520P and 2520R). Each logical device must be specified in a separate RDEVICE macro.

In addition, any other device that can be attached to a real processor can be specified in the RDEVICE macro by its device type. For unsupported devices that do not have a device type listed under the DEVTYPE operand, you should code the subclass on the CLASS operand. Then unsupported devices can be dedicated to a virtual machine, and CP can log any error recordings. CP does not use unsupported devices for its own operations.

If a device specified in the RDEVICE macro is not supported by VM/SP, the following MNOTE message (warning level) is generated:

```
UNSUPPORTED DEVICE TYPE
```

The device is generated as an unsupported device. An unsupported device can be used only if it is dedicated to a virtual machine. It is dedicated to a virtual machine if a DEDICATE control statement is coded in the VM/SP directory for the virtual machine, or if it is attached to it by the CP ATTACH command.

Notes:

1. If you code a 2702 device type the SETADDR value must be specified.
2. If you code a 3278 or 3279 device type the MODEL= operand must be specified.

MODEL=
is the model number for a particular device.

The model number must be coded for: 2305, 3330, and 3333 DASD; 3278 and 3279 display devices; 3410 and 3411 tape drives; and 3203, 3262, and 3800 printers. If not specified, model number defaults to zero except for:

- 3203 printer which defaults to model 4
- 3262 printer which defaults to model 1
- 3800 printer which defaults to model 1
- 3704, 3705, or 3725 Communication Controller that causes an MNOTE to be generated.

Model is a value that can be:

Model	Device
1 or 2	2305
4 or 5	3203
1, 5, or 11	3262
1, 2, or 11	3330
1 or 11	3333
A1 - H8, J1 - L4	3704, 3705-I, or 3705-II
1 or 2	3725-I or 3725-II
1, 2, 3, 4, 5, or 6	2415
5 or 7	2420
1, 2, or 3	3410 or 3411
3, 4, 5, 6, 7, or 8	3420
1	3272 or 3274, Model 1B
2, 3, 4, or 5	3278
2A	3278 consoles for 4300 processors
2C	3279 consoles for 4300 processors
2 or 3	3279
4	3289
1, 3, or 8	3800
D12, D14	6262

Notes:

1. The 3277 Model 1 is a 480-character display screen and is supported by VM/SP only as a dedicated device.
2. If a model number is included for devices that do not require model numbers, system generation is ended with an error message. If a model number is specified for a FB-512 device, it is ignored.
3. If DEVTYPE=3278 or 3279, MODEL= *must* be specified.
4. If DEVTYPE=3290, MODEL should not be specified. For 3278 compatibility mode, VM assumes that this is a 3278. See "Miscellaneous Restrictions" on page 424 for more information about defining the 3290.
5. The MODEL operand should not be coded for a 3480, 3422, or 9347 tape drive. If it is, it will be ignored, and an MNOTE will be issued.
6. If the console is a real 3205, the RDEVICE can either be a 3278 or 3279 and the *model* can either be 2A or 2C, respectively.
7. The model operand should not be coded for 9332 or 9335 devices. If it is, it will be ignored and an MNOTE will be issued.

FEATURE=
are the device's optional features. Features can be written in any order. They are:

Feature	Explanation
7-TRACK	7-track head on a tape drive
CONV	Conversion feature on a 7-track tape drive
DUALDENS	Dual density on a tape drive
OPRDR	Operator identification card reader on a 3277 Model 2, or magnetic slot reader on a 3278 or 3279
EMUL3270	The device is being emulated as a 3270. (Valid for a 3277, 3278, 3279, or 3290.)
E3270HLD	A device with this feature will have its line held upon logoff, disconnect, or force. (Invalid without the EMUL3270 feature.)
SYSVIRT	A 3330V (DEVTYPE=3330) device that may be used by VM/SP for mounting MSS system volumes
TRANS	Translation feature on a 7-track tape drive
UNVCHSET	Universal character set printer
VIRTUAL	A 3330V (DEVTYPE=3330) device that may be dedicated to a virtual machine
2CHANSW	Two-channel switch feature for tape or DASD drive
4CHANSW	Four-channel switch feature for tape or DASD drive
2WCGMS	A 3800 (DEVTYPE=3800) device with four Writeable Character Generation Modules
4WCGMS	A 3800 (DEVTYPE=3800) device with four Writeable Character Generation Modules
FH	3350 Fixed-head Feature (3340 optional)

Note: For a 3330V device, either FEATURE=VIRTUAL or FEATURE=SYSVIRT must be specified.

Coding Considerations

To allow CMS to correctly verify tape mode set operations, the correct feature code for a tape device must be specified.

Note: The dual density selected by the DUALDENS feature is dependent on the tape device and model specified in the DEVTYPE and MODEL operands.

| The FEATURE operand should not be coded for a 3480 or 9347 tape drive.

If the local 3277, 3278, or 3279 display device is equipped with the optional operator identification card reader or magnetic reader attachment, then the virtual machine operator can gain access to the system (log on) only by inserting a magnetically encoded card.

The FEATURE=OPRDR operand of the RDEVICE macro specifies that this is a display device with a card reader. FEATURE=OPRDR is invalid if DEVTYPE=3158.

Notes:

1. The features EMUL3270 and OPRDR cannot both be specified for the same device.
2. The 7-TRACK, CONV, DUALDENS, and TRANS features are not allowed for the 8809.
3. The 7-TRACK, CONV, and TRANS features are not allowed for the 3430.

Although allowable, it is not necessary to designate FEATURE=(2CHANSW/4CHANSW) on the RDEVICE macro. DMKCPI dynamically determines if the hardware has a two or four-channel switch feature.

FEATURE = FH is valid only for a 3350 DASD device or a 3330 in emulation mode. For all other DASD devices that may have the FH feature installed, it is either provided by the device type (for example, 2305) or determined at IPL or VARY ONLINE time.

You can specify FEATURE=(2CHANSW/4CHANSW) on the RDEVICE macro to indicate hardware support of reserve/release CCWs. But CP will not use this information. Instead, DMKCPI gets this information at initial CP IPL time by issuing a release CCW to the tape or count-key-data DASD volumes. For FB-512 devices, CP checks the RDFEAT bit in the appropriate FB-512 RDCBLOK. If you do specify FEATURE=(2CHANSW/4CHANSW) on the RDEVICE macro for documentation, one of the following MNOTEs will be issued:

```
2CHANSW    FEATURE IGNORED

4CHANSW    FEATURE IGNORED
```

CLASS=
is the device class. It is either the output spooling class or a special subclass for unsupported devices.

Note: * is a valid output class for printers. It indicates that the printer will accept any output class.

Output Spooling Classes

The spooling classes (cl,cl...) list up to four output spooling classes separated by commas. This form of the CLASS operand can be specified only for a 1403, 1443, 3203, 3211, 3262, or 3289 Model 4 printer, or 2520P, 2540P or 3525 card punch. The spooling class, cl, is one alphanumeric character. If you specify more than one class, you must separate them by commas. If no class is specified, class A is assumed for printers and punches.

CLASS is used by the CP START command and may be changed by this command. For a complete description of the START command, and more information about spooling classes, see the *VM/SP CP System Command Reference.*

Subclass for Unsupported Devices

Specify a device subclass for unsupported device types only. CP uses the subclass when it translates virtual CCW strings directed to unsupported devices. This form of the CLASS operand is valid only if the device type specified on the DEVTYPE operand does not appear in the list of valid device types.

Subclasses are:

DASD	Direct Access Storage Devices
TAPE	Tape devices
TERM	Terminals
GRAF	Display mode terminals
URI	Unit record input devices
URO	Unit record output devices

You must determine the correct subclass to specify for any device type that does not appear in the list of valid device types under the DEVTYPE operand. Do not code a subclass for any device type that appears in that list. For example, a 1287 Optical Reader is an unsupported device for VM/SP. It does not appear in the list of VM/SP supported devices and is not listed as a device type for the DEVTYPE operand of the RDEVICE macro. However, you can define a 1287 and use it if you dedicate it to a virtual machine. You must decide the correct subclass. For example:

```
RDEVICE ADDRESS=010,DEVTYPE=1287,CLASS=URI
```

defines a 1287 Optical Reader at address 010. The 1287 belongs to the unit record input (URI) subclass.

Notes:

1. If you use this form of the CLASS operand, and the unsupported device does not function properly, try dedicating the device to a virtual=real machine and stopping CCW translation (by issuing SET NOTRANS ON). A maximum of 32 sense bytes can be in the RDEVBLOK created for an unsupported device.
2. The CLASS operand is invalid if you are specifying service record file devices.

ADAPTER =

is the terminal control or transmission adapter used to connect a telecommunication I/O device to its control unit. This operand is required if a DEVTYPE of 2701, 2702, 2703, 3704, 3705, 3725, or ICA is specified. It is ignored if specified for any other device type.

BSCA specifies an IBM Binary Synchronous Terminal Adapter Type II for a 2701, or an IBM Binary Synchronous Terminal Control Type II for a 2703, 3704, 3705, or 3725. BSCA must be specified for remote 3270 terminals and printers.

IBM1 specifies that an IBM Terminal Adapter Type I attaches a 1050 or 2741 to a 2701, or that an IBM Terminal Control Type I attaches a 1050 or 2741 to a 2702 or 2703, or that a Line Interface Base Type I attaches a 1050 or 2741 to a 3704, 3705, or 3725 Communication Controller.

SDLC specifies that a 4331 Communications Adapter operates its teleprocessing lines in Synchronous Data Link Control (SDLC) mode. ADAPTER = SDLC is valid only when you specify DEVTYPE = ICA.

TELE2 specifies that a 3101 display terminal, or a CPT-TWX (Models 33/35) Terminal attaches to:

- A Telegraph Terminal Adapter Type II in a 2701
- A Telegraph Terminal Control Type II in a 2702 or 2703
- A Line Interface Base Type I in a 3704, 3705, or 3725.

TYPE1 specifies the channel adapter accessed by a 3704. For DEVTYPE = 3705, TYPE 1 or TYPE4 may be coded. TYPE5 specifies the channel adapter accessed by a 3725 Communication Controller. In identifying the channel adapter, TYPE1, TYPE4, or TYPE5 according to the DEVTYPE parameter must be specified for the Emulation Program (EP). In identifying the line adapter, IBM1, TELE2, or BSCA can be specified only in relation to another RDEVICE macro containing ADAPTER = TYPE1 or TYPE4 (if DEVTYPE = 3704 or 3705) or TYPE5 (if DEVTYPE = 3725).

SETADDR =

is the set address (SAD) command issued for a telecommunication line attached to a 2702, 3704, or 3705 control unit. This operand is required if the device is a 2702.

Sadnum Value	**Command**
0	SADZERO
1	SADONE
2	SADTWO
3	SADTHREE
4	(no SAD command is issued)

CPTYPE =

is the 3704/3705/3725 control program to be run in a 37XX Communication Controller.

EP specifies the 2701, 2702, or 2703 Emulation Program.

NCP specifies the Network Control Program.

PEP specifies the Partitioned Emulation Program.

ALTCU=

specifies an alternate control unit address to be used if paths through the primary control unit are unavailable. *cuu* is a three-digit hexadecimal address. Only one ALTCU can be specified.

The ALTCU *cuu* must specify an address with a low order of 0 or 8. Otherwise, the following MNOTE is issued:

```
INVALID ALTCU ADDRESS
```

The ALTCU operand is valid only for tape and DASD volumes. An MNOTE is issued if an invalid device type is specified.

```
"ALTCU" IS INVALID FOR DEVICE TYPE "devtype"
```

The ALTCU operand should be specified when you have the String Switch feature to support two control unit paths to a device.

The ALTCU operand should be specified to designate a second control unit path to a string of tape drives when two 3480 control units are attached through the Dual Control Unit Communications Feature.

In an MP (multiprocessor) system, the alternate control unit address may be the same as the primary control unit address. This specification is required when two control units, each having just one channel attachment to one unique side of an MP complex, attach to the same string of DASD or tape and have the same address.

For example, suppose you had the following conditions:

- An MP complex composed of processors A and B
- A 3880 storage director attached to channel 1 as X'120' on processor A
- A second 3880 storage director attached to channel 1 as X'120' on processor B
- The two storage directors attached to the same string of DASD.

Then, the following macro definitions would be required:

```
     RDEVICE ADDRESS=(120,8),ALTCU=120
     RCTLUNIT ADDRESS=120,FEATURE=32-DEVICE
```

Figure 37 on page 303 shows an example of multiple storage directors attached to a single DASD.

When these definitions are used in an MP configuration, CP correctly reflects the real hardware configuration by creating two RCUBLOKs to match the two real storage directors attached to the single string of DASD.

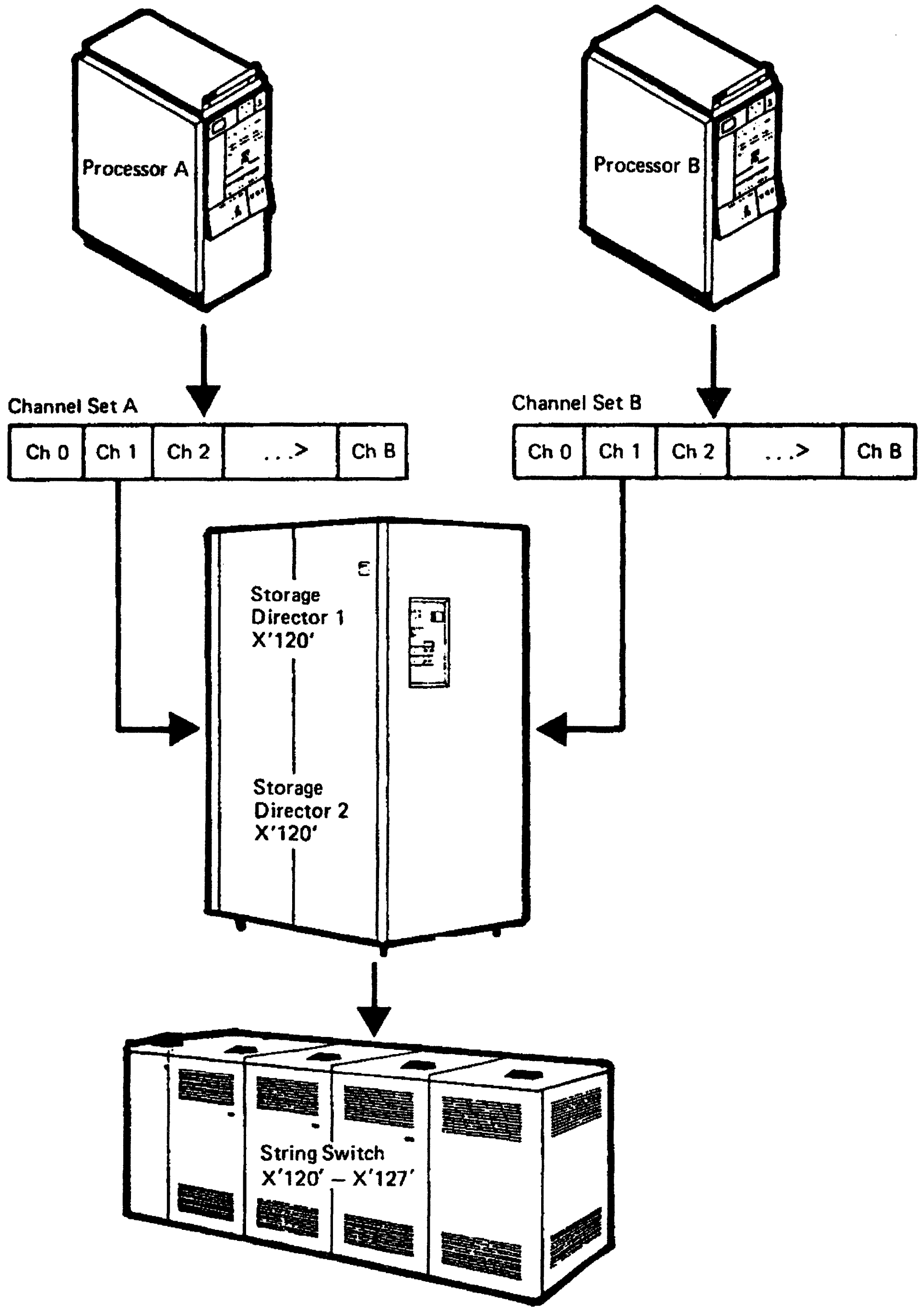


Figure 37. Storage Directors Attached to a Single DASD

The ALTCU cuu address should specify the low address associated with the alternate real control unit. When the FEATURE=xxx-DEVICE operand indicates that the control unit supports more than 16 devices and the devices on the second or following group of 16 devices are defined by separate RDEVICE macros, the ALTCU cuu should identify the logical RCUBLOK in VM/SP. VM/SP constructs one RCUBLOK for each set of 16 devices supported by the real control unit.

Assuming an alternate control unit configuration where each of two control units support 32 devices, the following two macro definitions are acceptable:

```
        RDEVICE ADDRESS=(300,32),ALTCU=400
        RCTLUNIT ADDRESS=300,FEATURE=32-DEVICE
        RCTLUNIT ADDRESS=400,FEATURE=32-DEVICE
        RCHANNEL ADDRESS=3
        RCHANNEL ADDRESS=4

        RDEVICE ADDRESS=(300,16),ALTCU=400
        RDEVICE ADDRESS=(410,16),ALTCU=310
        RCTLUNIT ADDRESS=300,FEATURE=32-DEVICE
        RCTLUNIT ADDRESS=400,FEATURE=32-DEVICE
        RCHANNEL ADDRESS=3
        RCHANNEL ADDRESS=4
```

CPNAME =
is the one-to-eight alphanumeric character name of a 3704/3705 control program that is to be automatically loaded in the 3704 or 3705 at IPL time. If an automatic load is not desired, omit this operand. This operand is only coded if VM/SP created the EP. Do not code this operand for a 3725 Communication Controller.

Note: Failure to code the CPNAME operand on the RDEVICE macro for the 3704/3705 base address causes VM/SP to mark the device "not operational" at IPL time. The cluster on that 3704/3705 may be unusable.

BASEADD =
is the native address (load address) of the 3704/3705/3725 that controls the physical line(s). This operand is required for correct operation of VM/SP recovery management for emulation lines based on a 3704/3705/3725. This operand is valid only if ADAPTER = IBM1 (or = TELE2 or = BSCA). It must be specified to use the 370x EP Line Trace Facility.

CLUSTER =
is the label of the CLUSTER macro that defines the clustered or stand-alone remote control unit attached to this line. This operand is valid only if ADAPTER = BSCA is specified.

IMAGE =
is the image library to be used by the 3800 printer device after a cold start if none is specified on the START command. If this operand is omitted, the default is IMAG3800.

CHARS =
is one-to-four characters that represent the character arrangement table for the 3800 printer device to be used after a cold start if none is specified on the START command. If this operand is omitted, the default is GF10.

FCB =
is the lines per inch to be used for the page separator for the 3800 printer after a cold start if none is specified on the START command. You can specify 6, 8, 10 (for the 3800 Printer Model 3,) or 12 lines per inch. If this operand is omitted, the default is 6 lines per inch.

DPMSIZE =
is the maximum size of the delayed purge queue for the 3800 printer device to be used after a cold start if none is specified on the START command. If this operand is omitted, the default is 1. (The maximum allowed is 9.)

Examples:

The following examples show how the RDEVICE macro instructions describe a 1403 printer with the Universal Character Set (UCS) feature, four 9-track, 800 bpi tape drives, and eight CPT-TWX lines on a 2702.

```
RDEVICE ADDRESS=00E,DEVTYPE=1403,FEATURE=UNVCHSET,       X
        CLASS=(A,C)
RDEVICE ADDRESS=(0C0,4),DEVTYPE=2401
RDEVICE ADDRESS=(030,8),DEVTYPE=2702,ADAPTER=TELE2,      X
        SETADDR=2
```

The following example shows how the RDEVICE macro invocation generates the Ethernet or IBM Token Ring LAN subsystems to CP. The RDEVICE macro must specify four consecutive device addresses for each CETI group. An Ethernet subsystem on a 9370, must have one CETI group specified and a device type of 3088. For example:

```
RDEVICE     ADDRESS=(500,4),DEVTYPE=3088
```

A Token-Ring LAN 9370 subsystem must have between one and three CETI groups specified and a device type of 3088. For example:

```
RDEVICE     ADDRESS=(500,12),DEVTYPE=3088
```

The Token-Ring Adapter may be used by three different access methods, one for each CETI group. For example, TSAF will only use one CETI group, VTAM may use another, and TCP/IP may use a third CETI group.

Special Considerations for Coding the 3704/3705/3725 RDEVICE Macro

The 3704/3705/3725 Communication Controllers have varied uses. Consequently, there are special considerations for coding the RDEVICE macro.

3704 Communication Controller

Model	Storage
A1	16K
A2	32K
A3	48K
A4	64K

3705-I Communication Controller

Model	Storage
A1, B1, C1, D1	16K
A2, B2, C2, D2	48K
B3, C3, D3	80K
B4, C4, D4	112K
C5, D5	144K
C6, D6	176K
D7	208K
D8	240K

3705-II Communication Controller

Model	Storage
E1, F1, G1, H1	32K
E2, F2, G2, H2	64K
E3, F3, G3, H3	96K
E4, F4, G4, H4	128K
E5, F5, G5, H5	160K
E6, F6, G6, H6	192K
E7, F7, G7, H7	224K
E8, F8, G8, H8	256K
J1, K1, L1	320K
J2, K2, L2	384K
J3, K3, L3	448K
J4, K4, L4	512K

Figure 38. 3704/3705 Models

Note: Specify any 3704/3705 outside the range of models listed in Figure 38 as MODEL=H8.

EP-Only Control Programs: If the 3704/3705 is to be run in emulation mode:

- The (cuu,nn) form of the ADDRESS operand generates multiple RDEVBLOKs.
- Specify the appropriate name for CPNAME if VM/SP created the EP. CPNAME is not valid for a 3725 Communication Controller.

To generate additional emulator lines for the same 3704/3705, use the following coding guidelines on subsequent RDEVICE macros:

- Omit the CPTYPE, CPNAME, and MODEL operands.

- Specify the ADAPTER as IBM1, TELE2, or BSCA, as appropriate.

For ADAPTER=IBM1 (or TELE2), the SETADDR operand must also be specified, exactly as if the device were a 2702 or 2703.

Note: If you use the (cuu,nn) form of the ADDRESS operand to generate multiple RDEVBLOKs and specify the CPNAME and ADAPTER=TYPE1 operands on the RDEVICE macro, the additional RDEVBLOKs are generated as ADAPTER=IBM1 and SETADDR=4.

Other 3704/3705/3725 RDEVICE Considerations: For each physical 37XX there should be only one RDEVICE macro that specifies the ADAPTER=TYPE1, TYPE2, TYPE3, TYPE4, or TYPE5, MODEL, CPTYPE, and CPNAME operands. This RDEVICE macro defines the base address of the 3704/3705/3725 (that is, the real address used to do the load and dump operations). If the physical device is a 3705/3725 with two channel adapters installed, there may be a second RDEVICE macro that specifies the ADAPTER=TYPE1, TYPE2, TYPE3, TYPE4, or TYPE5, MODEL, and CPTYPE operands. There must *never* be a second use of the CPNAME operand. Even if CPTYPE=EP is specified, the 3704/3705/3725 base address cannot be used as a telecommunication line. Its function is only to load and dump the 37XX. The device type and class are different from those of all other lines generated.

Whenever there is more than one subchannel address (CPTYPE=EP), include in the DMKRIO file all RCTLUNIT macros required to specify real addresses that the EP control program may use.

If you have a 3704/3705/3725 and a 2701/2702/2703 on the same VM/SP system, the virtual addresses for the 3704/3705/3725 must not be the same as any of the real 2701/2702/2703 addresses.

Examples:

Examples of RDEVICE macro specifications follow.

Example 1

```
RDEVICE ADDRESS=(020,16),                               X
        DEVTYPE=3704,                                   X
        MODEL=A2,                                       X
        ADAPTER=TYPE1,                                  X
        CPNAME=VMEP01
```

This describes a 32K 3704 at address X'020', with 15 emulator lines addressed X'021' to X'02F' and with the default parameter of ADAPTER=IBM1 and SETADDR=4. The 3704 is to be loaded with the Emulation Program 'VMEP01'.

Example 1a

```
RDEVICE ADDRESS=(030,16),                               X
        DEVTYPE=3704,                                   X
        ADAPTER=IBM1,                                   X
        SETADDR=2,                                      X
        BASEADD=020
```

This describes an additional 16 emulator lines on the same 3704 specified by Example 1.

Example 2

```
RDEVICE ADDRESS=0E0,                               X
        DEVTYPE=3725,                              X
        ADAPTER=TYPE5,                             X
        MODEL=1
```

Example 3

```
RDEVICE ADDRESS=cuu,                               X
        DEVTYPE=3705,                              X
        ADAPTER=TYPE4,                             X
        MODEL=F8,                                  X
        CPTYPE=NCP
```

This describes a 3705 at address cuu.

3704/3705/3725 Error Messages: The RDEVICE macro instruction generates an entry in a table of programmable communication controllers when DEVTYPE = 3704, 3705, or 3725 is specified. This table has a maximum of 10 entries. The following message results if more than ten 3704, 3705, or 3725 devices are specified:

```
MORE THAN 10 TP CONCENTRATORS
```

This message indicates that the RDEVBLOK is generated, but no entry is made in the Programmable Communication Controller table.

Example of Coding an SNA Controller

Each of the following devices is treated the same by the control portion of VM/SP:

- 3705 running NCP
- 3725 running NCP
- 3720 running NCP
- 3274 local SNA (model 1A, 21A, 31A, or 41A) controller
- 3174 local SNA (model 1L in SNA mode) controller
- 3174 model 1L Token-Ring Gateway supported device (PC, 3174 model 3R, 3174 model 53R).

CP does not directly communicate with these devices. Native VM/VTAM or a guest machine running an SNA access method is required to communicate to these control units. (CP also does not communicate directly with the NCP portion of a PEP controller and this environment follows the following rules as well.)

To define these SNA controllers to VM/SP, use one of the following RDEVICE specifications.

```
RDEVICE ADDRESS=4F0, DEVTYPE=3725, ADAPTER=TYPE5, MODEL=2, CPTYPE=NCP
RDEVICE ADDRESS=4E0, DEVTYPE=3705, ADAPTER=TYPE4, MODEL=E8, CPTYPE=NCP
```

ADDRESS should specify the native subchannel address of the 37X5 controller. SNA controllers are typically on block channels.

DEVTYPE specifies the controller type. This is for documentation purposes only. For a 3274-41A, 3174-1L, or a device supported by the 3174 Token Ring Gateway feature either of the preceding examples will work. In fact, the 3705 definition will work for the 3725 and vice-versa.

ADAPTER specifies the 37X5 adapter type. For a 3720, 3274, 3174, or a device support by the 3174 Token-Ring Gateway feature, use one of the preceding definitions.

MODEL is device dependent and must be specified but is only for documentation purposes. This parameter is a leftover from when CP loaded an EP control program. It is ignored by CP when an NCP or PEP is to be loaded.

CPTYPE=NCP or CPTYPE=PEP tells CP not to load an EP.

```
RCTLUNIT ADDRESS=4E0,CUTYPE=3705,FEATURE=16-DEVICE
RCTLUNIT ADDRESS=4F0,CUTYPE=3725,FEATURE=16-DEVICE
```

Unit Record Error Messages

The RDEVICE macro instruction generates an entry in a table of printers or a table of punches or a table of readers for spooling when DEVTYPE = 1403, 1443, 2501, 2540P, 2540R, 3203, 3211, 3262, 3289 (MODEL = 4), 3289E, 3505, 3525, 3800, 4245, or 4248 is specified. Each table has a maximum of 32 entries; one of the following messages results if more than 32 readers, printers, or punches are specified.

```
MORE THAN 32 READERS
MORE THAN 32 PRINTERS
MORE THAN 32 PUNCHES
```

If any of these messages prints, it indicates that the RDEVBLOK is generated, but no entry is made in the printer or punch table; the device cannot be used for CP spooling.

Control Unit Error Messages

The RCTLUNIT macro generates an RCUBLOK containing an index to each of 16 possible devices. When ALTCU is specified, both the primary and alternate RCUBLOKS contain an index to the same RDEVBLOK. The following MNOTE is issued when an RDEVICE macro includes the ALTCU operand and another RDEVICE macro, defining a separate device and having the same address (cuu) as the alternate control unit address:

```
CONTROL UNIT TABLE for raddr1 IN USE by raddr2
```

Error Example:

```
RDEVICE ADDRESS=140,ALTCU=150
RDEVICE ADDRESS=150
RCTLUNIT ADDRESS=140
RCTLUNIT ADDRESS=150
RCHANNEL ADDRESS=1
```

Device 140 is defined with a primary control unit address of 140 and an alternate control unit address of 150. The ALTCU=150 specification indicates that the 150 RCUBLOK will contain an index to the 140 RDEVBLOK. In this example, an RDEVICE macro also appears for device 150. A conflict arises because the RCUBLOK index for control unit 150 cannot index to both RDEVBLOK 140 and RDEVBLOK 150. In the previous example, the user must remove the 150 RDEVICE macro to resolve the conflict.

RCTLUNIT Macro

The RCTLUNIT macro generates a real control unit block (RCUBLOK). One RCTLUNIT macro must be specified for each real control unit. The maximum number of real control units is 511, providing you have enough real storage to hold the real control unit blocks (RCUBLOKs). Control units generally fall into two classes: those supporting eight or fewer devices, and those supporting more than eight devices.

A control unit that supports eight or fewer devices must be assigned an address that is a hexadecimal multiple of eight (that is, it must be divisible by hex 8). All devices with an address equal to the control unit's address (the base address) or any of the next seven sequential addresses are mapped to this control unit. For example, devices with addresses of 018 through 01F are mapped to a control unit with address 018.

On a multiplexer channel, many device addresses may fall within the address range of one RCTLUNIT macro. When this occurs, only one RCTLUNIT macro may be coded, even though more than one real control unit is present. This case is an exception to the general rule that one RCTLUNIT macro must be specified for each real control unit. For example, a system console at address 009, a 2540 reader at address 00C and a 2540 punch at address 00D would be defined in a single RCTLUNIT macro with a control unit address of 008, even though the card reader/punch and the system console have different real control units. In this case, any valid control unit type can be coded. The only exception to this is that control units that operate on a shared subchannel must be specified by separate RCTLUNIT macros.

For control units supporting a range of more than eight device addresses, use the FEATURE operand. The base address must be divisible by 16. All devices from the base address up to the number of devices specified by the FEATURE= operand are mapped to the specified control unit. When a control unit supports more than eight devices, the RCTLUNIT macro must specify FEATURE=xxx-DEVICE, where xxx is the maximum number of addressable devices that can be attached to this control unit. The number of devices specified must be divisible by 16 and rounded to the next higher increment of 16 if not divisible. The maximum number of devices that can be attached to a control unit is 256.

For example, if you have a 3830 control unit with the 64-device feature installed, you must specify FEATURE=64-DEVICE for it, even if fewer than sixty-four 3330s are installed.

VM/SP requires that all devices on one physical control unit be specified on a single RCTLUNIT macro. The microcode in the 3830-2 that supports 3350 DASD allows address skipping (in blocks of eight addresses) on the same physical control unit.

Error Example:

```
Device Addresses 150-157 and 160-167 on first 3830-2
Device Addresses 158-15F and 168-16F on second 3830-2
```

This address scheme is **not supported** by CP. All addresses on a physical control unit must be specified with a single RCTLUNIT macro using the FEATURE=xxx-DEVICE operand, where appropriate, for a contiguous range of addresses.

A device that attaches directly to the channel without a separate control unit must still have an RCTLUNIT macro coded for it. For example, if a 3210 is defined with an RDEVICE macro, it must have a corresponding RCTLUNIT macro.

The 3880 Storage Control Unit contains two director modules. Each director module acts as a control unit providing I/O operations to a string of devices. Because each director module is separately addressable, one RCTLUNIT macro statement is required for each module. The optional ALTCH operand of the RCTLUNIT macro allows specification of up to three alternate channel interfaces to a single director module. VM/SP supports a maximum of four channel paths to a single director module.

RCTLUNIT macro instructions describing the control units for your system may be in any order, but they must be contiguous and follow all of the RDEVICE macro instructions in module DMKRIO. The first RCTLUNIT macro instruction also generates the label DMKRIOCU, which indicates the start of the real control unit blocks to CP.

The name field need not be specified for the RCTLUNIT macro instruction. If a name is specified it is ignored. The macro generates a name by appending the control unit address to the characters RCU. For example, if the control unit address is 230, RCU230 is generated.

Format

label	**RCTLUNIT**	**ADDRESS=** *cuu* **,CUTYPE=** *type* [**,ALTCH=** *(n,n,n)*] [**,FEATURE=** *xxx* **-DEVICE**] [**,UCW=**$\left\{\begin{matrix}\textbf{SHR}\\\textbf{UNS}\end{matrix}\right\}$]

Parameters

label
is any desired label.

ADDRESS=
is the real address of the control unit. The cuu consists of 3 hexadecimal digits. The high-order digit is the channel address of this control unit. The low-order two digits must be the lowest address of the control unit. The first digit may be any hexadecimal number from 0 to F. If the xxx-DEVICE feature is supported, the low-order digit must be 0. Otherwise, it must be either 0 or 8. Note that you do not need real devices attached to all eight addresses. However, the addresses that are not being used cannot be used by another control unit.

Notes:

1. If you have a 2701, 2702, or 2703, and a 3704, 3705, or 3725 running in emulation mode, make sure that their addresses are not the same.
2. If a device does not have a control unit (for example, 1403 at 00E, its address must still be covered by an RCTLUNIT definition. Any control unit address beginning at 008 would suffice.

CUTYPE=

is the device type of the control unit. For a list of possible device types that IBM recommends see Appendix A, "VM/SP Configuration Aid" on page 401.

Note that only one CTCA may be defined at a time. Also, any other control unit that can be attached to a real processor may be specified in an RCTLUNIT macro instruction by its device type.

Notes:

1. Specify an Integrated Printer Adapter (IPA) as a 2821.
2. Specify a 3274 Model 1B as a 3272.
3. Specify a 5088 as a 5088.
4. If you are using a 3289 Model 4 Printer as a system printer (DEVTYPE=3289E) attached to a 4331 Display Printer Adapter, specify CUTYPE=SVPC in the corresponding RCTLUNIT macro. Be careful not to specify a control unit type of 3272 or 3274. This will result in locking out other devices on the adapter, such as 3278s or a second printer, while a printer is operating.
5. If you are using a 4361-3 Workstation Adapter, specify CUTYPE=3274 in the RCTLUNIT macro.
6. If you are using a non-SNA 3174 model 1L, specify CUTYPE=3274 in the RCTLUNIT Macro. If you are using a SNA 3174 model 1L, specify CUTYPE=3725 in the RCTLUNIT Macro.

Even though some devices attach directly to the channel without a separate control unit, an RCTLUNIT macro instruction must be included for them. For example, if you want to define a 3215, you must code an RDEVICE and RCTLUNIT macro for the 3215. Even though the 3215 does not require a control unit, it requires an RCTLUNIT macro. If many devices have addresses that are within the same control unit address, only one RCTLUNIT macro can be specified. The control unit you specify is not important.

ALTCH=

specifies the alternate channel(s) to be used with the control unit address if the primary channel path is unavailable or offline. n represents the one-digit channel addresses for the alternate channel paths. You can specify up to three alternate channels for AP or UP systems. Only one alternate channel can be specified for multiprocessor systems. Specification of more than one alternate channel path for MP generated systems produces the following MNOTE:

```
INVALID ALTCH SPECIFICATION
```

There can be no splitting of control units when using alternate channels. All devices on one physical control unit must be defined as having alternate channel(s) or no alternate channel(s).

For the 3480 tape subsystem, you can specify up to three alternate channels for a control unit. In an MP environment, you can specify a maximum of two channels between a control unit and a processor.

Note: The ALTCH option is invalid for the 3380 Model CJ2.

FEATURE=

is the optional control unit feature. The feature, xxx-DEVICE, indicates that the control unit is capable of controlling more than eight devices. The prefix, xxx, can be 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240, or

256. Note that if the xxx-DEVICE feature is supported, the low-order digit must be 0. Otherwise, it must be either 0 or 8. For a list of the maximum number of devices that can be specified for each control unit, see Appendix A, "VM/SP Configuration Aid" on page 401. This feature can be specified for a 2403, 2702, 2703, 2803, 2835, 2840, 3088, 3272, 3274, 3345, 3380, 3480, 3704, 3705, 3725, 3803, 3830, 3880, 3990, FTA (File Tape Adapter,) HFCU (High Function Control Unit,) ICA (Integrated Communications Attachment,) IFA (Integrated File Adapter,) or ISC (Integrated Storage Control.)

A 3174 model 1L control unit with the Token-Ring Gateway feature can be customized to support up to 140 devices, with each Gateway supported device requiring an address on the channel. To support the 3174 1L Gateway and its Token-Ring devices, a single RCTLUNIT = 3725 macro must be specified with ADDRESS = cuu set to the real 3174-1L's address and FEATURE = xxx-DEVICE covering the address range of the 3174-1L and the Gateway supported devices. For example, if you had the 3174 at address 250 and had 30 Gateway supported devices, FEATURE = xxx-DEVICE must cover the addresses from 250 to 25D. You should code the RCTLUNIT macro as follows:

```
RCTLUNIT ADDRESS=250,CUTYPE=3725,
FEATURE=32-DEVICE
```

The prefix xxx for a 3088 must be either 32 or 64.

The Integrated File Adapter's 9821 feature, when used with 3344s, is too large for the RCTLUNIT macro to handle. The range of addresses 160-1F7 is invalid.

A DASD controller, such as a 3830, can be ordered with a hardware feature to support 16, 32, or 64 devices. The RCTLUNIT macro *must* specify FEATURE = XXX-DEVICE to match the hardware feature of the controller, regardless of the actual number of devices attached to it. The ADDRESS operand is also very critical. For 32 devices, the ADDRESS must be x00, x20, x40, or another address (where x is the channel number); always an even multiple of x20 (decimal 32). Likewise, for 64 devices, the only correct addresses are x00, x40, x80, and xC0.

This information is important when an upgrade of installed DASD controllers occurs. Make sure the ADDRESS value is changed in addition to the FEATURE specification. For any unsupported control unit, FEATURE = 16-DEVICE is valid and is the maximum you can specify. Unsupported control units are any that do not appear in Chapter 2, "Planning Device Configurations for VM/SP" on page 13.

Warning: The starter system does not provide support for configurations over 16 devices when you are defining those needed to do the system generation. Therefore, if you have control units that share more than 16 devices that are switchable to another processor, the channel interface enable switch on the other processor should be in the disable position while you do the system generation.

For the 3480 tape subsystem, specify FEATURE = 16-DEVICE if two control units are attached through the Dual Control Unit Communications Feature.

UCW =

is the attribute of the UCW. This can be specified for only 3272 or 3274 control units.

SHR indicates a shared UCW (this is the default)

`UNS` indicates an unshared UCW

Examples:

The following examples show how the RCTLUNIT macro instructions describe the control units for: a 3215 console printer-keyboard with address 01F, a 3880, and a 3705 with lines 040 through 04B.

```
RCTLUNIT ADDRESS=018,CUTYPE=3215
RCTLUNIT ADDRESS=220,CUTYPE=3880,FEATURE=32-DEVICE
RCTLUNIT ADDRESS=040,CUTYPE=3705,FEATURE=16-DEVICE
```

The following example shows how the RCTLUNIT macro invocation generates the Ethernet or IBM Token Ring LAN subsystems to CP. The RCTLUNIT macro must specify an address of the first port of the CETI group, a control unit type of 3088, and specify a 32-DEVICE feature.

```
RCTLUNIT    ADDRESS=500,CUTYPE=3088,FEATURE=32-DEVICE
```

Channel Error Messages

The RCHBLOK contains an index to each of 32 possible RCUBLOKS. When the ALTCH operand is specified on the RCTLUNIT macro, both the primary and alternate RCHBLOKS contain an index to the same RCUBLOK. The following error message is issued when one RCTLUNIT macro is coded (specifying the ALTCH operand), and an additional RCTLUNIT macro is coded, which creates an RCUBLOK for the alternate channel address (specified by the first RCTLUNIT macro).

```
CHANNEL TABLE FOR RCUxxx IN USE BY RCUyyy
```

Error Example:

```
RDEVICE ADDRESS=250
RDEVICE ADDRESS=350
RCTLUNIT ADDRESS=250,ALTCH=(3)
RCTLUNIT ADDRESS=350
RCHANNEL ADDRESS=2
RCHANNEL ADDRESS=3
```

The ALTCH specification indicates that the RCHBLOK for channel three should index to the 250 RCUBLOK. The RCTLUNIT macro for 350 causes a conflict because the RCHBLOK cannot index to both the 250 and 350 RCUBLOKS. In the previous configuration, the RCTLUNIT macro and RDEVICE macro for 350 must be removed.

RCHANNEL Macro

The RCHANNEL macro generates a real channel block (RCHBLOK). An RCHANNEL macro instruction must be coded to define each real channel in the I/O configuration.

The RCHANNEL macro instructions describing your channels may be in any order, but they must be contiguous and follow all of the RCTLUNIT macro instructions in DMKRIO. The first RCHANNEL macro instruction generates the label DMKRIOCH, which indicates the start of the real channel blocks to CP.

No name need be specified for the RCHANNEL macro instruction. If a name is specified, it is ignored. The RCHANNEL macro generates a name by appending the channel address to the characters RCHAN. For example, if the channel address is 2, the name RCHAN2 is generated.

Format

label	**RCHANNEL**	**ADDRESS** = *address* **,CHTYPE** = { **SELECTOR** | **MULTIPLEXOR** | **BLKMPXR** | **FTA** }

Parameters

label
is any desired label.

ADDRESS =
is the real address of the channel. It is a hexadecimal number from 0 to F.

CHTYPE =
is the type of channel.

SELECTOR
indicates a selector channel.

MULTIPLEXOR
indicates a byte-multiplexer channel.

BLKMPXR
indicates a block-multiplexer channel.

FTA
indicates a file-tape adapter on a 43xx.

Examples:

The following examples show how the RCHANNEL macro instructions describe a multiplexer channel whose address is 0, a selector channel whose address is 1, and a block multiplexer channel whose address is 2.

```
RCHANNEL  ADDRESS=0,CHTYPE=MULTIPLEXOR
RCHANNEL  ADDRESS=1,CHTYPE=SELECTOR
RCHANNEL  ADDRESS=2,CHTYPE=BLKMPXR
```

If any errors are detected, the real channel block is not generated. This results in undefined symbols in the real control unit blocks for this channel.

The following example shows how the RCHANNEL macro invocation generates the Ethernet or IBM Token Ring LAN subsystems to CP. The RCHANNEL macro must specify a channel address and a channel type of byte multiplexor.

```
RCHANNEL    ADDRESS=5,CHTYPE=MULTIPLEXOR
```

RIOGEN Macro

The RIOGEN macro instruction generates the channel index table and unit record and console tables. RIOGEN must appear as the last macro instruction before the END statement in the DMKRIO file.

The name field must not be specified for the RIOGEN macro.

Format

label	**RIOGEN**	CONS=*cuu* [,ALTCONS=(*cuu*[,*cuu*,*cuu*...])] [,SRF=(*cuu*[,*cuu*,*cuu*...])]

Parameters

label
is any desired label.

CONS=
is the address of the VM/SP primary system console. The address is a hexadecimal device address that was previously specified in an RDEVICE macro entry. This device must be either a 3036, 3066, 3210, 3215, 7412, 3277, 3278, or 3279 (local attachment), or a 3278 Model 2A, 1052 (via a 2150 freestanding console), a system console for the 3158 (in printer-keyboard mode with the 3213 Printer Model 1 required), or a system console for the 3138 or 3148 (in printer keyboard mode with a 3286 printer required, or in display mode).

ALTCONS=
is the address or a list of addresses of alternate consoles. These addresses are hexadecimal device addresses that were previously specified in an RDEVICE macro instruction. There is no limit on the number of alternate consoles that may be specified. These devices, which should be located as close as possible to the primary system console, may be any device supported as a VM/SP logon device (except for those remote terminals connected through 3704/3705/3725 Communication Controllers). If the primary system console is not operational at VM/SP system initialization, an attempt is made to access the first alternate console. If the first alternate console is not operational, an attempt is made to start the next alternate console. If an operational console is found, the console will be used as the VM/SP system operator's console. If no operational alternate console is found (or if no alternate console was specified), CP enters a disabled wait state with a wait state code of X'005' in the instruction address register (IAR).

Coding Considerations: The alternate console must not be a telecommunications line on a real 3704/3705/3725 Communication Controller unless the 3704/3705/3725 was previously loaded by some other operating system with a 270X Emulator Program.

If the alternate console is a 2741 Communication Terminal, or 3767 Communication Terminal (operating as a 2741), it must use the EBCDIC transmission code. If the alternate console is a local 3277, it must be a Model 2.

SRF=
is the address or a list of addresses of SRF (service record file) devices used for the 3031, 3032, or 3033 processors. *cuu* is the hexadecimal device address that was previously specified in an RDEVICE macro statement. The device type of the SRF is 7443.

In a 3033AP or 3033MP system, there are two 3036 consoles. Each of these consoles has two SRF devices; therefore, you should specify multiple SRF devices at system generation. The SRF addresses you specify in the RIOGEN macro statement should be the same as the addresses of the SRF devices attached to the service support consoles (see Note 3). The RIOGEN macro statement produces an MNOTE warning message if you specify more than 32 SRF devices.

Usage Notes

1. In 3033AP or 3033MP systems with I/O configured asymmetrically to one processor, to access the SRF devices in both 3036 consoles, a channel path must be available from the I/O processor to both SRF devices.
2. If an SRF device is inaccessible during initialization of the error recording cylinders, an error message is sent to the system operator. Processing continues, but the frames from that SRF device are not placed on the error recording cylinders.
3. Only one of the two SRF devices of a 3036 console is accessible at any one time by the VM/SP control program. Therefore, if both SRF devices of a 3036 are specified on the RIOGEN macro, message DMKIOH559W will be issued for *one* of these SRF devices during initialization of the error recording cylinders. Because both SRF devices of a 3036 console contain identical frame data, only one SRF per 3036 needs to be successfully accessed during error recording initialization.

Examples:

The following examples define a primary system console (01F) with an alternate console (050), and a system console (009) with no alternate console.

```
RIOGEN CONS=01F,ALTCONS=050
RIOGEN CONS=009
```

Example of Coding the Real I/O Configuration File (DMKRIO)

In this example, macros are coded to support the following real devices:

Number of Devices	Device Type
1	2540 Card Reader/Punch
2	1403 Printers with the Universal Character Set feature
1	3211 Printer with the Universal Character Set feature
2	3215 Console Printer-Keyboards
1	2955 Data Adapter Unit
1	3278 Console, Model 2A
8	3279 Color Display Stations
1	3705 Communication Controller (with an IBM1, TELE2 and BSCA adapter)
1	2305 Fixed Head Storage with 8 addresses
2	3330 Disk Storage devices (One unit has eight modules and the other has 10. The unit with 10 modules is switchable between two channels.)
1	3350 Direct Access Storage with eight addresses (string switched)
1	3380 Direct Access Storage with 16 addresses
1	3420 Magnetic Tape Unit, Model 8
2	3420 Magnetic Tape Units, Model 7
1	Multiplexer channel
1	Selector channel
3	Block multiplexer channels
1	Channel-to-Channel Adapter
2	Channel interfaces on the 3851 MSC
96	3330V Direct Access Storage devices, 48 of which can be dedicated to one or more virtual machines and 48 of which are to be used for VM/SP system volumes
4	3330-1 device addresses that are not real spindles, but rather allow the processor to have direct access to the MSC tables through the 3830-3 Staging Adapter
1	4245 Printer
1	4248 Printer

Figure 39 on page 321 shows an example of a real configuration. The Real I/O configuration file (DMKRIO) that supports this example is shown in Figure 40 on page 322.

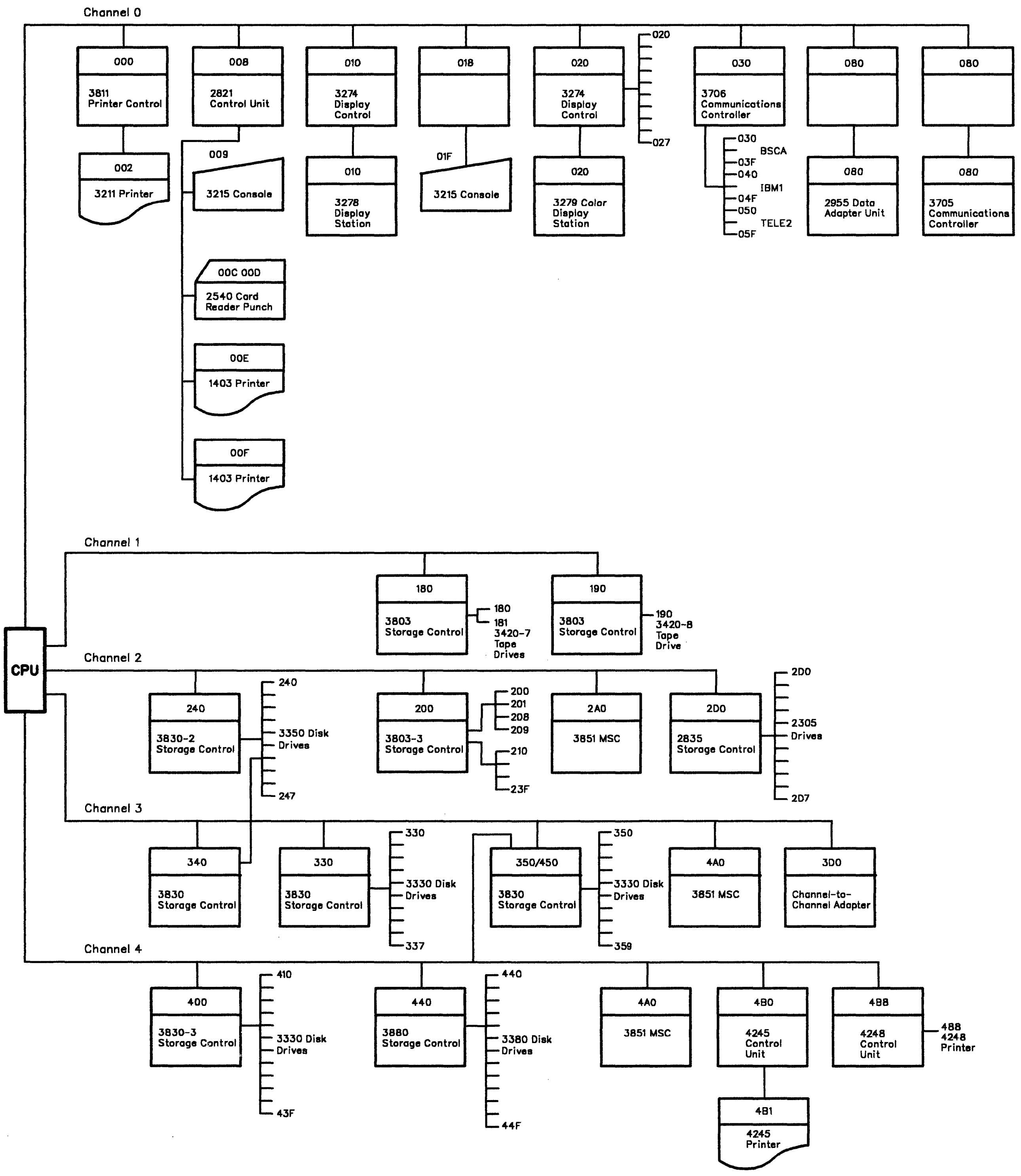


Figure 39. Example of a Real Configuration

```
DMKRIO CSECT
       COPY OPTIONS
       RDEVICE ADDRESS=002,DEVTYPE=3211,CLASS=(X,A),FEATURE=UNVCHSET
       RDEVICE ADDRESS=009,DEVTYPE=3215
       RDEVICE ADDRESS=00C,DEVTYPE=2540R
       RDEVICE ADDRESS=00D,DEVTYPE=2540P,CLASS=(X,A)
       RDEVICE ADDRESS=00E,DEVTYPE=1403,CLASS=(X,A),FEATURE=UNVCHSET
       RDEVICE ADDRESS=00F,DEVTYPE=1403,CLASS=(S),FEATURE=UNVCHSET
       RDEVICE ADDRESS=010,DEVTYPE=3278,MODEL=2A
       RDEVICE ADDRESS=01F,DEVTYPE=3215
       RDEVICE ADDRESS=(020,8),DEVTYPE=3279,MODEL=3
       RDEVICE ADDRESS=(030,16),DEVTYPE=3705,ADAPTER=BSCA,BASEADD=0B0
       RDEVICE ADDRESS=(040,16),DEVTYPE=3705,ADAPTER=IBM1,BASEADD=0B0
       RDEVICE ADDRESS=(050,16),DEVTYPE=3705,ADAPTER=TELE2,BASEADD=0B0
       RDEVICE ADDRESS=080,DEVTYPE=2955
       RDEVICE ADDRESS=0B0,DEVTYPE=3705,ADAPTER=TYPE4,MODEL=F4,CPTYPE=EP
       RDEVICE ADDRESS=(180,2),DEVTYPE=3420,FEATURE=DUALDENS,MODEL=7
       RDEVICE ADDRESS=190,DEVTYPE=3420,FEATURE=DUALDENS,MODEL=8
*      DEVICE ADDRESSES 200, 201, 208, 209 ALLOW ACCESS TO MSC TABLES
       RDEVICE ADDRESS=(200,2),DEVTYPE=3330,MODEL=1
       RDEVICE ADDRESS=(208,2),DEVTYPE=3330,MODEL=1
       RDEVICE ADDRESS=(210,48),DEVTYPE=3330,MODEL=1,FEATURE=SYSVIRT
       RDEVICE ADDRESS=(240,8),DEVTYPE=3350,ALTCU=340
       RDEVICE ADDRESS=2A0,DEVTYPE=3851
       RDEVICE ADDRESS=2D0,DEVTYPE=2305,MODEL=2
       RDEVICE ADDRESS=(330,8),DEVTYPE=3330,MODEL=1
       RDEVICE ADDRESS=(350,8),DEVTYPE=3330,MODEL=1
       RDEVICE ADDRESS=(358,2),DEVTYPE=3330,MODEL=11
       RDEVICE ADDRESS=3D0,DEVTYPE=CTCA
       RDEVICE ADDRESS=(410,48),DEVTYPE=3330,MODEL=1,FEATURE=VIRTUAL
       RDEVICE ADDRESS=(440,16),DEVTYPE=3380
       RDEVICE ADDRESS=4A0,DEVTYPE=3851
       RDEVICE ADDRESS=4B1,DEVTYPE=4245,CLASS=(A)
       RDEVICE ADDRESS=4B8,DEVTYPE=4248,CLASS=(A)
       RCTLUNIT ADDRESS=000,CUTYPE=3811
       RCTLUNIT ADDRESS=008,CUTYPE=2821
       RCTLUNIT ADDRESS=010,CUTYPE=3274
       RCTLUNIT ADDRESS=018,CUTYPE=3215
       RCTLUNIT ADDRESS=020,CUTYPE=3274
       RCTLUNIT ADDRESS=030,CUTYPE=3705,FEATURE=48-DEVICE
       RCTLUNIT ADDRESS=080,CUTYPE=2955
       RCTLUNIT ADDRESS=0B0,CUTYPE=3705
       RCTLUNIT ADDRESS=180,CUTYPE=3803
       RCTLUNIT ADDRESS=190,CUTYPE=3803
       RCTLUNIT ADDRESS=200,CUTYPE=3830,FEATURE=64-DEVICE
       RCTLUNIT ADDRESS=240,CUTYPE=3830,FEATURE=16-DEVICE
       RCTLUNIT ADDRESS=2A0,CUTYPE=3851
       RCTLUNIT ADDRESS=2D0,CUTYPE=2835
```

Figure 40 (Part 1 of 2). Real I/O Configuration File (DMKRIO)

```
        RCTLUNIT ADDRESS=330,CUTYPE=3830,FEATURE=16-DEVICE
        RCTLUNIT ADDRESS=340,CUTYPE=3830,FEATURE=16-DEVICE
        RCTLUNIT ADDRESS=350,CUTYPE=3830,FEATURE=16-DEVICE,ALTCH=4
        RCTLUNIT ADDRESS=3D0,CUTYPE=CTCA
        RCTLUNIT ADDRESS=400,CUTYPE=3830,FEATURE=64-DEVICE
        RCTLUNIT ADDRESS=440,CUTYPE=3880,FEATURE=16-DEVICE
        RCTLUNIT ADDRESS=4A0,CUTYPE=3851
        RCTLUNIT ADDRESS=4B0,CUTYPE=4245
        RCTLUNIT ADDRESS=4B8,CUTYPE=4248
        RCHANNEL ADDRESS=0,CHTYPE=MULTIPLEXOR
        RCHANNEL ADDRESS=1,CHTYPE=SELECTOR
        RCHANNEL ADDRESS=2,CHTYPE=BLKMPXR
        RCHANNEL ADDRESS=3,CHTYPE=BLKMPXR
        RCHANNEL ADDRESS=4,CHTYPE=BLKMPXR
        RIOGEN CONS=01F,ALTCONS=(010,009)
        END
```

Figure 40 (Part 2 of 2). Real I/O Configuration File (DMKRIO)

Note: You should code FEATURE = 16-DEVICE for 3830 and 3880 control units, even if only eight devices are attached. If the control unit is physically capable of 16 devices and the 16-DEVICE feature is not coded, missing interrupts will result.

Considerations for Coding the Input/Output Configuration Source File

If you are generating VM/SP on a 3081 Processor complex, you must define additional I/O information to the processor controller. To define the I/O configuration to the 3081 processor, create an I/O configuration source file and write it to the processor controller using the Input/Output Configuration Program. The IOCP source file you create should contain entries that match the Real I/O configuration file (DMKRIO).

When preparing the IOCP source file, be sure you code the IOCP macro definitions in the following order:

- Channel paths
- Control units
- I/O devices.

It is not necessary to group all macro definitions of the same type together; however, you must ensure that devices and control units are defined under the appropriate channel macros. If not, IOCP fails with an error message.

Do not intermix IOCP macro definitions and VM/SP I/O macro definitions in the same file. You must maintain two distinct files; the Real I/O configuration file (DMKRIO) and the corresponding IOCP source file. See the *Input/Output Configuration Program User's Guide and Reference* for IOCP source file coding conventions.

Figure 41 on page 325 is an example of the IOCP source file needed to match the real I/O configuration defined in Figure 40 on page 322. For additional information about the Input/Output Configuration program, see the *VM Running Guest Operating Systems* book.

Example of Coding the Input/Output Configuration Source File

```
ID    MSG1=SAMPLE IOCP FILE
CHPID PATH=((00,0,0)),TYPE=BY
CHPID PATH=((01,1,0)),TYPE=BL
CHPID PATH=((02,2,0)),TYPE=BL
CHPID PATH=((03,3,0)),TYPE=BL
CHPID PATH=((04,4,0)),TYPE=BL
CNTLUNIT CUNUMBR=000,PATH=00,SHARED=N,UNIT=3811,                      X
      UNITADD=((00,8))
CNTLUNIT CUNUMBR=001,PATH=00,SHARED=N,UNIT=2821,                      X
      UNITADD=((08,8))
CNTLUNIT CUNUMBR=002,PATH=00,SHARED=N,UNIT=3274,                      X
      UNITADD=((10,8))
*IOCP  DUMMY DEVICE FOR CP ONLY
*IOCP CNTLUNIT CUNUMBR=003,PATH=00,SHARED=N,UNIT=3215,                X
*IOCP      UNITADD=((18,8))
CNTLUNIT CUNUMBR=004,PATH=00,SHARED=N,UNIT=3274,                      X
      UNITADD=((20,8))
CNTLUNIT CUNUMBR=005,PATH=00,SHARED=N,UNIT=3705,                      X
      UNITADD=((30,48))
CNTLUNIT CUNUMBR=006,PATH=00,SHARED=N,UNIT=2955,                      X
      UNITADD=((80,8))
CNTLUNIT CUNUMBR=007,PATH=00,SHARED=N,UNIT=3705,                      X
      UNITADD=((B0))
CNTLUNIT CUNUMBR=010,PATH=01,SHARED=Y,UNIT=3803,                      X
      UNITADD=((80,8))
CNTLUNIT CUNUMBR=011,PATH=01,SHARED=Y,UNIT=3803,                      X
      UNITADD=((90,8))
CNTLUNIT CUNUMBR=020,PATH=02,SHARED=N,UNIT=3830,                      X
      UNITADD=((00,64))
CNTLUNIT CUNUMBR=021,PATH=02,SHARED=N,UNIT=3830,                      X
      UNITADD=((40,16))
CNTLUNIT CUNUMBR=024,PATH=02,SHARED=N,UNIT=3851,                      X
      UNITADD=((A0,8))
CNTLUNIT CUNUMBR=025,PATH=02,SHARED=N,UNIT=2835,                      X
      UNITADD=((D0,8))
CNTLUNIT CUNUMBR=031,PATH=03,SHARED=N,UNIT=3830,                      X
      UNITADD=((30,16))
CNTLUNIT CUNUMBR=032,PATH=03,SHARED=N,UNIT=3830,                      X
      UNITADD=((40,16))
CNTLUNIT CUNUMBR=033,PATH=(03,04),SHARED=N,UNIT=3830,                 X
      UNITADD=((50,16))
CNTLUNIT CUNUMBR=034,PATH=03,SHARED=N,UNIT=CTCA,                      X
      UNITADD=((D0,8))
CNTLUNIT CUNUMBR=040,PATH=04,SHARED=N,UNIT=3830,                      X
      UNITADD=((00,64))
```

Figure 41 (Part 1 of 2). IOCP Source File

```
CNTLUNIT CUNUMBR=041,PATH=04,SHARED=N,UNIT=3880,                      X
      UNITADD=((40,16)),PROTOCL=S
CNTLUNIT CUNUMBR=042,PATH=04,SHARED=N,UNIT=3851,                      X
      UNITADD=((A0,8))
CNTLUNIT CUNUMBR=043,PATH=04,SHARED=N,UNIT=4245,                      X
      UNITADD=((B0,8))
CNTLUNIT CUNUMBR=044,PATH=04,SHARED=N,UNIT=4248,                      X
      UNITADD=((B8,8))
IODEVICE ADDRESS=002,UNIT=3211,CUNUMBR=000
IODEVICE ADDRESS=009,UNIT=3215,CUNUMBR=001
IODEVICE ADDRESS=00C,UNIT=2540R,CUNUMBR=001
IODEVICE ADDRESS=00D,UNIT=2540P,CUNUMBR=001
IODEVICE ADDRESS=00E,UNIT=1403,CUNUMBR=001
IODEVICE ADDRESS=00F,UNIT=1403,CUNUMBR=001
IODEVICE ADDRESS=010,UNIT=3278,MODEL=2A,CUNUMBR=002
*IOCP    DUMMY DEVICE FOR CP ONLY
*IOCP    IODEVICE ADDRESS=01F,UNIT=3215,CUNUMBR=003
IODEVICE ADDRESS=(020,8),UNIT=3279,MODEL=3,CUNUMBR=004
IODEVICE ADDRESS=(030,16),UNIT=3705,CUNUMBR=005
IODEVICE ADDRESS=(040,16),UNIT=3705,CUNUMBR=005
IODEVICE ADDRESS=(050,16),UNIT=3705,CUNUMBR=005
IODEVICE ADDRESS=080,UNIT=2955,CUNUMBR=006
IODEVICE ADDRESS=0B0,UNIT=3705,CUNUMBR=007
IODEVICE ADDRESS=(180,2),UNIT=3420,MODEL=7,CUNUMBR=010
IODEVICE ADDRESS=190,UNIT=3420,MODEL=8,CUNUMBR=011
IODEVICE ADDRESS=(200,2),UNIT=3330,MODEL=1,CUNUMBR=020
IODEVICE ADDRESS=(208,2),UNIT=3330,MODEL=1,CUNUMBR=020
IODEVICE ADDRESS=(210,48),UNIT=3330,MODEL=1,CUNUMBR=020
IODEVICE ADDRESS=(240,8),UNIT=3350,CUNUMBR=(021,032)
IODEVICE ADDRESS=2A0,UNIT=3851,CUNUMBR=024
IODEVICE ADDRESS=2D0,UNIT=2305,MODEL=2,CUNUMBR=025
IODEVICE ADDRESS=(330,8),UNIT=3330,MODEL=1,CUNUMBR=031
IODEVICE ADDRESS=(350,8),UNIT=3330,MODEL=1,CUNUMBR=033
IODEVICE ADDRESS=(358,2),UNIT=3330,MODEL=11,CUNUMBR=033
IODEVICE ADDRESS=3D0,UNIT=CTC,CUNUMBR=034,TIMEOUT=N
IODEVICE ADDRESS=(410,48),UNIT=3330,MODEL=1,CUNUMBR=040
IODEVICE ADDRESS=(440,16),UNIT=3380,CUNUMBR=041
IODEVICE ADDRESS=4A0,UNIT=3851,CUNUMBR=042
IODEVICE ADDRESS=4B1,UNIT=4245,CUNUMBR=043
IODEVICE ADDRESS=4B8,UNIT=4248,CUNUMBR=044
```

Figure 41 (Part 2 of 2). IOCP Source File

Chapter 9. Preparing the CP System Control File (DMKSYS)

Contents of Chapter 9

Overview

The CP system control file consists of macro statements that describe the following:

- CP system residence device
- System storage size
- CP-owned direct access devices
- System operator's user identification
- System timer value
- System pointer variables
- Automatic performance monitoring parameters
- Accounting parameters
- System identification
- System spool parameters
- Security journaling parameters
- System dump space parameters
- System T-DISK security parameters
- Missing I/O interruption timer intervals.

You are responsible for ensuring the presence and accuracy of the macros described below.

The DMKSYS ASSEMBLE file, which is provided with the starter system, does not assemble properly unless you have reserved adequate space for the CP nucleus.

Format

```
DMKSYS  CSECT
        SYSOWN   macro
        SYSRES   macro
        SYSOPR   macro
        SYSCOR   macro
        SYSTIME  macro
        SYSMON   macro
        SYSJRL   macro
        SYSACNT  macro
        SYSFORM  macro
        SYSPCLAS macro
        SYSIPL   macro
        SYSID    macro
        SYSORD   macro
        SYSFCN   macro (optional)
        SYSMIH   macro (optional)
        SYSLOCS  macro
        END
```

Usage Notes

1. Sample DMKSYS files are provided on the product tape. If you use these, you can save the CMS system at the end of system generation. You may modify this file if you wish. For example, you could modify the starter system DMKSYS file to add other CP-owned volumes.
2. SYSLOCS must always be the last macro coded.

3. You must code all macro statements, except for SYSFCN and SYSMIH, even if you code no operands on some macro statements. SYSFCN and SYSMIH are optional.

4. If either of the two optional macros, SYSFCN and SYSMIH, is coded, it must be placed immediately before the SYSLOCS macro. If both are coded, either one may come first.

Performance Considerations for Coding the DMKSYS File Macros

The following recommendations may help reduce arm and channel contention, and may improve the performance of a VM/SP system.

- Provide separate CP volumes for paging and spooling and have the volumes mounted on separate channels.

  Note: CP paging packs should not be generated on channel 0 because channel 0 is disabled during extend processing. Violation of this restriction will cause the system to hang in a deadlock with outstanding paging I/O on channel 0.

- If you have a heavy I/O production virtual machine (for example, one running OS/VS1 or VSE/AF), try to keep all its major I/O devices on a separate channel from a channel handling the CMS system residence volume or other user's disks.

- Try to keep read-only minidisks (for example, the CMS system residence disk and source disks) that are frequently accessed on separate volumes from users' read/write minidisks. If possible, also keep them on separate channels.

- If you have disks with fixed head areas (2305/3340/3350/3330) you should use them for preferred paging. To do this, allocate this area as preferred paging space using the Format/Allocate program.

- The relative amounts of storage for dynamic paging and free storage can be optimized by using the FREE operand of the SYSCOR macro statement. You should initially allocate 1 page of fixed free storage for each virtual machine that is logged on, based on the average number of users that you expect to have logged on at any one time.

- Using the automatic monitoring facilities, study the load conditions and performance profile for your system as soon as possible. These facilities, used with programs similar to the IBM FDP (Field Developed Program) Virtual Machine Facility/370: Virtual Machine Monitor Analysis Program (VMMAP) (5664-191) are designed to make data collection and reduction easy, thus allowing the analyst to concentrate on analysis. Data collection can be done on a regular basis by specifying AUTO=YES on the SYSMON macro instruction. The system assumes default values for other operands if none are specified.

- You can determine the monitoring interval for the missing interruption handler support. Some classes of devices may require an interval other than the default interval provided in DMKSYS. For instance, if you have many direct storage devices, you may need to lengthen the time interval for the DASD class. This would eliminate the unnecessary missing interruption handler processing for devices that are functioning properly. Be careful not to lengthen the time interval too much, because you may lose the usefulness of missing interruption monitoring.

  The SYSMIH macro statement controls the time intervals for missing interruption monitoring. If you have already generated the system, you can change the time intervals with the SET MITIME command.

SYSOWN Macro

The SYSOWN macro generates the list of up to 255 CP-owned DASD volumes. A CP-owned volume is either the CP system residence volume, or a volume that contains VM/SP paging, spooling, dump, directory, or temporary disk space. It must contain a CP allocation table allocating these areas at cylinder 0, record 4 for count-key-data devices, or blocks 3 and 4 for FB-512 devices. Even if a volume has a VM/SP allocation table in the appropriate area, allocation data is ignored unless the volume appears as an operand in the SYSOWN macro instruction.

Note: The SYSOWN macro must appear before the SYSRES macro in the assembly listing.

The name field must not be specified for the SYSOWN macro.

Format

SYSOWN	*volid,[volid,...]*

Parameters

volid
is the CP-owned volume identifier of from one-to-six alphanumeric characters.

Example:

The following is an example of the SYSOWN macro:

```
SYSOWN VMSRES,VMPK01
```

Special Considerations for Allocating Space on CP-Owned Volumes: The following considerations should help you to allocate space efficiently on CP-owned volumes:

- The SYSOWN macro statement does not distinguish preferred from nonpreferred cylinders on a volume. Use the SYSORD macro statement to specify the order in which the preferred and/or nonpreferred devices are to be searched to satisfy paging and spooling operations. The CP Format/Allocate service program accepts PAGE and TEMP as operands on the control statements to designate preferred and nonpreferred paging areas (spool, page overflow).
- If a volume is specified in a SYSOWN statement, but is not mounted when the generated system is loaded (by way of the IPL command), that volume is considered not available to VM/SP. Processing continues, if possible. The operator can mount and attach the volume later, if it is needed.
- Only volumes that contain paging, spooling, dump, directory, or T-DISK space need be identified as CP-owned volumes. All other volumes are described either by directory entries or by logically attaching the entire device.
- If you add another volume to the SYSOWN list, you must add it at the end of the list. (Otherwise, if you attempt a warm start after regenerating and loading CP, the relative entry number used to locate system spool buffers is incorrect.) Then reassemble DMKSYS, build the CP nucleus, and reload it on the system residence volume or on an alternate IPL device. Use the VMFASM EXEC to reassemble DMKSYS. Then use the VMFBLD EXEC to reload the CP nucleus. See the *VM/SP Service Guide* for details on using the VMFASM and VMFBLD EXECs.

Note: If you deleted a volume from the SYSOWN list, you cannot IPL warm; you must do a cold start.

- If you have saved systems (systems that can be loaded by name, thus not performing the initial program load procedure), you must reserve space on a CP-owned volume to hold the named systems you want saved. The DASD space you reserve, for each named system you wish to save, should be enough to contain the number of pages specified in the SYSPGCT operand of the NAMESYS macro, plus 1 page for system use.
- If your VM/SP system has a 3704 or 3705, you must reserve space on a CP-owned volume to contain the 3704/3705 control program image. See "Planning for the 3704/3705/3725 Control Program" on page 195 for information about how much DASD space you should reserve.
- If there is more than one directory area on the volumes in the SYSOWN list, the first available path to a directory on a SYSOWN volume is taken. This will be the system directory.

SYSRES Macro

The SYSRES macro describes characteristics of the CP system residence volume. Note that the name field must not be specified for the SYSRES macro instruction.

Special Considerations for Coding the SYSRES Macro: The following information should help you when you code the SYSRES macro:

- All operands must be specified with appropriate values.
- Areas required for SYSNUC, SYSERR, SYSWRM, and SYSCKP must be formatted using the CP Format/Allocate service program, and must be allocated as permanent space on the SYSRES volume, but not in cylinder 0.
- On a 3340, *alternate track* cylinders cannot be used for regular data. On a 3340 Model 35, use only cylinders 0-347. On a 3340 Model 70, use only cylinders 0-695.
- An MSS 3330V volume may not be used as the VM/SP SYSRES volume.

You can improve system availability if you define and save more than one copy of your CP nucleus. If the primary nucleus is then damaged or unavailable, the system operator can select an alternate nucleus to IPL.

See Chapter 5, "Extending VM/SP" on page 139 for a description and sample alternate nucleus configuration.

Format

```
SYSRES    SYSVOL = { volid }
                   { *     }

          SYSRES = [([{address [,altaddr]}[)]

          SYSTYPE = {type}

          SYSNUC = { start                     }
                   { (start[, [length]])       }

          SYSCLR = { YES }
                   { NO  }

          SYSERR = { start                     }
                   { (start[,length[,[volid]]]) }

          SYSCKP = { start                     }
                   { (start[,length[, [volid]]]) }

          SYSWRM = { start                     }
                   { (start[,length[, [volid]]]) }
```

Parameters

SYSVOL=
is the volume identifier of the system residence disk. *volid* is the one-to-six alphanumeric character volume label on the disk where the CP nucleus will be written. * causes CP to store the actual volume label into the nucleus during system generation.

SYSRES =

designates a three-digit hexadecimal device address (vdev) for the DASD volume to contain the newly generated system. *altaddr* identifies an alternate address for writing the CP nucleus. If both address and *altaddr* are specified, parentheses are required. For multiprocessing systems, specifying an alternate address allows native execution of CP module DMKSAV on either processor.

Note: Executed on a virtual machine, the addresses are virtual addresses; on a real machine (stand-alone,) they are real addresses. For information on the creation of an IPLable nucleus tape, see the *VM/SP Installation Guide.*

When the CP nucleus is written from tape to the system residence volume, the following steps occur after IPLing the CP nucleus tape:

1. DMKLD00E (the loader) passes control to DMKSAV.
2. DMKSAV uses the SYSRES operands of the SYSRES macro in DMKSYS to find the DASD address on which to write the CP nucleus:
   a. The address operand is sought first on the processor where the tape was IPLed. If the DASD at address is found, the CP nucleus is written to address.
   b. If the DASD specified by address is not found, then DMKSAV looks for [altaddr]. If the DASD at [altaddr] is found, the CP nucleus is written to [altaddr].
   c. If neither address nor [altaddr] are found, the PSW is loaded with a wait state of X'00000010'. To recover from the wait state, attach the system residence volume to the IPLed processor as either address or [altaddr]. Then, execute an external interrupt to DMKSAV to write the CP nucleus to this device.

SYSTYPE =

is the device type. The device types can be:

2305-1 (Models 1 and 2), 3310, 3330, 3340, 3350, 3370, 3375, 3380, 9313, 9332, 9335, and FB-512.

The following table shows device types and their required device type specification:

Device	Device Type Specification
3350 (native mode)	3350
3350 (compatibility mode)	3330
3344	3340
3333	3330
3380 (all models)	3380

SYSNUC =

designates where the CP nucleus resides on your disk storage device.

start is either the page number or cylinder number that identifies the beginning of the CP nucleus area in storage. For FB-512 storage devices, specify the one-to-six digit number of the starting page. For CKD storage devices, use the one-to-four digit number of the starting cylinder.

length is the number of pages or contiguous cylinders that are needed to contain the CP nucleus. If you do not specify 'length', your SYSRES volume will not be protected from *nucleus area overflow*. This condition occurs when a CP nucleus exceeds its defined area and can be the source of IPL problems.

Typical requirements for each device are shown in the following table:

Device Type	Cylinders	Pages
2305	10	
3330	8	
3333	8	
3340	10	
3350	7	
3375	7	
3380	5	
FB-512		285*

*approximate number

SYSCLR=

specifies automatic clearing of all previously written data and directory areas residing on T-DISK DASD space. This prevents other users from accidentally accessing these areas. If (YES) is specified, T-DISK DASD space is cleared to binary zeros at CP initialization and when attaching a CP-owned volume containing T-DISK allocations. T-DISK space is also cleared when a user detaches a T-DISK. Specifying SYSCLR=YES provides additional security for your VM/SP installation. If you specify (NO), the system will clear cylinder 0, track 0 on T-DISK DASD.

The SYSCLR option is required. Specifying SYSCLR with invalid or misspelled options produces the following MNOTE:

```
INVALID SYSCLR OPTION
```

and the specification defaults to YES.

SYSERR=

designates the area where error records are written on your disk storage device.

start is either the page number or cylinder number that identifies the beginning of the error recording area on your disk storage device. For FB-512 storage devices, specify the one-to-six digit number of the starting page. For CKD storage devices, use the one-to-four digit number of the starting cylinder.

length is the number of pages or contiguous cylinders that are needed to contain the error recording area. For FB-512 storage devices, specify a number from 1 to 999999. The default is 100 pages. For count-key-data storage devices, use a number from 2 to 9. The default is 2 cylinders.

volid is the one-to-six alphanumeric character volume label of the CP-owned volume where this error recording area resides. If 'volid' is not specified, the SYSRES volume is assumed.

SYSCKP=

designates the area in storage where CP checkpoint start data is to be saved. *start* is either the page number or cylinder number that identifies the beginning of the checkpoint area on your disk storage device. For FB-512 storage devices, specify the one-to-six digit number of the starting page. For CKD storage devices, use the one-to-four digit number of the starting cylinder. *length* is the number of pages or contiguous cylinders that are needed to contain the checkpoint area. For FB-512 storage devices, specify a number from 1 to 999999. The default is 50 pages. For count-key-data storage devices, use a number from 1 to 20. The default is 1 cylinder.

volid is the one-to-six alphanumeric character volume label of the CP-owned volume where this checkpoint area resides. If 'volid' is not specified, the SYSRES volume is assumed.

The checkpoint space available determines how many spool file IDs may be assigned. The maximum number of spool file IDs is the lesser of either the 9900 system limit or the number of checkpoint slots allocated. For example: 1 cylinder of a 3330 will allow slightly less than 2000 spool file IDs. The number of cylinders or pages required for the checkpoint start data is dependent upon the device type. They are as follows:

Device Type	Minimum Recommended No. of Cylinders/Pages	Cylinders/Pages for 9900 Spool File IDs
2305	3 cylinders	14 cylinders
3330	1 cylinder	6 cylinders
3340	3 cylinders	14 cylinders
3350	1 cylinder	4 cylinders
3375	1 cylinder	5 cylinders
3380	1 cylinder	3 cylinders
FB-512	50 pages	298 pages

Use the following formulas to calculate space needed for the dynamic checkpoint start area. The formula for each device type is:

Table 25. Dynamic Checkpoint Start Area Calculations

Device Type	Formula
2305/3340	$N = \frac{[26+(NSP/32)+(NRS/34)]}{24}$
3330	$N = \frac{[59+(NSP/32)+(NRS/34)]}{57}$
3350	$N = \frac{[122+(NSP/32)+(NRS/34)]}{120}$
3375	$N = \frac{[98+(NSP/32)+(NRS/34)]}{96}$
3380	$N = \frac{[152+(NSP/32)+(NRS/34)]}{150}$
FB-512	$N = [2+(NSP/32)+(NRS/34)]$

where:

`N`	is the number of cylinders or pages required for checkpoint start data.
`NSP`	is the number of spool files to be checkpointed. There are 32 entries per 4096-byte record.
`NRS`	is the number of real spooling devices defined in DMKRIO.

Note: When using the preceding formulas for count-key-data DASD, disregard any remainder in the answer.

SYSWRM=

designates the area in storage where CP warm start information is to be saved.

start is either the page number or cylinder number that identifies the beginning of the warm start area on your disk storage device. For FB-512 storage devices, specify the one-to-six digit number of the starting page. For CKD storage devices, use the one-to-four digit number of the starting cylinder.

length is the number of pages or contiguous cylinders that are needed to contain the warm start area. For FB-512 storage devices, specify a number from 1 to 999999. The default is 50 pages. For count-key-data storage devices, use a number from 1 to 20. The default is 1 cylinder.

volid is the one-to-six alphanumeric character volume label of the CP-owned volume where this warm start area resides. If 'volid' is not specified, the SYSRES volume is assumed.

Use the following formulas to calculate the number of warm start cylinders or pages required. When you use the formulas, round up all remainders to the nearest whole number. For example, for a 3330 system residence volume with:

- A maximum of 32 spool files in the system at one time
- A maximum of 128 cylinders available for spool files
- A maximum of 50 active users at one time

the calculation is:

$$N = \frac{[59 + 32/32 + 128/128 + 200/50]}{57} = \frac{65}{57} = 1$$

The formula for each device type is:

Table 26. Warm Start Area Calculations

Device Type	Formula
3340/2305	$N = \frac{[26+(NSF/32)+(NCS/128)+((NAUx4)/50)]}{24}$
3330	$N = \frac{[59+(NSF/32)+(NCS/128)+((NAUx4)/50)]}{57}$
3350	$N = \frac{[122+(NSF/32)+(NCS/128)+((NAUx4)/50)]}{120}$
3375	$N = \frac{[98+(NSF/32)+(NCS/128)+((NAUx4)/150)]}{96}$
3380	$N = \frac{[152+(NSF/32)+(NCS/128)+((NAUx4)/50)]}{150}$
FB-512	N = [2+(NSF/32)+(NPS/128)+((NAUx4)/50)]

where:

`N`	is the number of cylinders or pages required for warm start data.
`NSF`	is the maximum number of spool files in the system at any one time. There are 32 spool file blocks per 4096-byte record
`NCS`	is the number of cylinders available for spool files. There are 128 allocation blocks per 4096-byte record.
`NPS`[27]	is the number of pages available for spool files. There are 128 allocation blocks per 4096-byte record.
`NAU`	is the maximum number of active users in the system at any one time. There are 50 accounting records per 4096-byte record.

Example:

The following SYSRES macro defines the system residence volume as the 3380 volume with a serial number of VMSRES. During the system generation procedure this volume is found at address 123. The VM/SP nucleus resides at cylinder 882 and the warm start storage area is cylinders 877 and 878. The error recording area starts at cylinder 879 and the checkpoint start storage area is cylinder 442. The format of the SYSRES macro is:

```
SYSRES SYSVOL=VMSRES,SYSRES=123,SYSTYPE=3380,SYSNUC=882,SYSCLR=,X
          SYSWRM=(877,2),SYSERR=(879,2),SYSCKP=(442,1)
```

[27] For each allocation extent type (PERM, TEMP, and so on) on an FB-512 device, 24 bytes should be added to this total. For example, if the FB-512 device in question is allocated with PERM, TEMP, and PERM space, divide the number of pages available for spooling (NPS) by 200 (that is, (24 x 3) + 128).

SYSOPR Macro

The SYSOPR macro specifies the system operator's user ID, and the user ID of the operator who is to receive VM/SP system dumps. The same user ID may be specified in both operands.

The name field must not be specified for the SYSOPR macro instruction.

Format

SYSOPR	$\left[\underline{\text{SYSOPER}}=\left\{\begin{array}{l}\underline{\text{OPERATOR}}\\ \textit{userid}\end{array}\right\}\right]$ $\left[\text{SYSDUMP}=\left\{\begin{array}{l}\underline{\text{OPERATNS}}\\ \textit{userid}\end{array}\right\}\right]$

Parameters

SYSOPER =
: is the user ID of the virtual machine to be assigned to the system operator. If SYSOPER is not specified, the user ID OPERATOR is used.

  The user ID is a character string up to eight characters long.

SYSDUMP =
: is the user ID (a string of up to eight characters) of the virtual machine whose spool input receives the system dump file after a system restart. This user ID also receives guest virtual machine dumps produced by CP VMDUMP, if you specify the destination as SYSTEM. If SYSDUMP is not specified, the user ID OPERATNS is used. If you plan to use IPCS, let this operand default to OPERATNS or specify the IPCS user ID.

Example:

The following SYSOPR macro designates the OP virtual machine as the system operator and directs the system dumps to the CPSYS virtual machine.

```
SYSOPR SYSOPER=OP,SYSDUMP=CPSYS
```

Note: The first Class A user ID to log on after the current VM operator's virtual machine is logged off becomes the new system operator.

SYSCOR Macro

The SYSCOR macro generates the internal control block called the CORTABLE. The AP and MP operands specify whether VM/SP will try to make use of an additional processor.

The name field must not be specified for the SYSCOR macro instruction.

Format

```
SYSCOR   RMSIZE = { xxxxxK }  [,FREE = ffff]
                  {   yyM  }
                               [,AP = YES ]
                               [      NO  ]
                               [          ]
                               [,MP = YES ]
                               [      NO  ]
         [TRACE = nnn]
```

Parameters

RMSIZE =
is the amount of real storage available for VM/SP. If a real machine has more storage than what you specified on SYSCOR, the system does not use the extra storage. If less storage is available in the machine than what you specified on SYSCOR, the system uses the lesser figure. Note that a message indicating the amount of storage being used is displayed at the operator's console at system IPL.

The value, xxxxx, is a three-to-five digit number that designates the amount of real storage in terms of K bytes, where 1K = 1024 bytes. This value may range from 384K to 16384K. It must always be a multiple of 2.

The value, yy, is a one or two-digit number that designates the amount of storage in terms of M bytes, where 1M = 1024K bytes. This value may range from 1M to 16M.

Note: Do not specify a value much larger than the size of real storage, because the generated core table uses a large amount of real storage.

FREE =
is a one-to-four digit number that specifies the number of fixed free storage pages to be allocated at VM/SP initialization. This number must be greater than three. The amount of storage represented must not be greater than either 25% of the value specified for RMSIZE, or RMSIZE less the V = R area if generated.

| The recommended initial value for ffff is 1 page for each virtual machine logged on, based on the average number of virtual machine users. Note that if you are running a second level VM system, the second level ID should own the cache of the 3880 Storage Subsystem. If it does not, monitor activity on the 3880 Storage Subsystem can cause I/O errors due to command reject.

If FREE is not specified, VM/SP allocates three pages for the first 256K of real storage and 1 page for each additional 64K thereafter not including the V = R size, if any. In AP and MP modes, the default is increased by 25%.

One method of determining if you need to increase the value of FREE (after your initial sysgen) is to XEDIT your CPNUC MAP and locate the label "DMKFRENP."

Write down the hex location of DMKFRENP, get out of XEDIT, and enter the command:

```
dcp hexloc.8
```

The first 4 bytes are the number of extends (in hex). The last 4 bytes are the number of disextends. If the extends are greater than the disextends, you should increase FREE by the difference.

AP =

YES specifies that processing is in attached processor mode if the attached processor is available at system IPL.

NO specifies that processing is in uniprocessor mode regardless of the presence of an attached processor.

MP =

YES specifies that initialization processing occurs in multiprocessor mode.

NO specifies that initialization processing occurs in uniprocessor mode regardless of the presence of a second processor.

MP = YES and AP = YES parameters are mutually exclusive. A system generated for MP execution will run on an MP, AP, or UP system. However, it will not function with maximum efficiency for AP or UP modes of operation. If you code both AP = YES and MP = YES, the following MNOTE is issued:

```
"MP=YES AND AP=YES BOTH SPECIFIED; MP=YES ASSUMED"
```

Note: An additional 25% of free storage is allocated in AP or MP mode. (See FREE = .)

TRACE =

is the decimal number of 4K pages for the trace table. The default is 1 page for each 256K bytes of real storage up to 16M, plus 1 page for each megabyte above 16M. The minimum is 1 page (4K), but IBM strongly recommends that you do not specify fewer than 4 pages (16K). The maximum is 999 pages (3996K), but this is a much larger value than needed. Many installations find that a 4-page trace table is large enough.

Note: You may wish to generate one CP nucleus with a 4-page trace table and an alternate nucleus with a default trace table.

Examples:

The first example defines real storage as 256K (262,144 bytes), the second example defines real storage as 1M (1,048,576 bytes), and the third example defines the trace table as 4 pages (16K bytes or 16,384 bytes).

```
SYSCOR   RMSIZE=256K,
SYSCOR   RMSIZE=1M,
SYSCOR   TRACE=4
```

Usage Notes

1. If you did not code the TRACE operand, the minimum size of the TRACE table is the computed default size.
2. If you coded the TRACE operand, the minimum size of the TRACE table is the size you specified.

SYSTIME Macro

The SYSTIME macro generates information needed to set the hardware time of day (TOD) clock. The value stored in the TOD clock represents time taken at Greenwich Mean Time, and must be corrected to local time whenever it is examined. The system operator can alter the defined time value by using the store clock function.

The name field must not be specified for the SYSTIME macro instruction.

Format

```
SYSTIME    [ZONE= { 0       } ]
                  { h       }
                  { (h,m)   }
                  { (h,m,s) }
                  { (h,,s)  }
           [,LOC= { EAST } ]
                  { WEST }
           [,ID=  { GMT } ]
                  { xxx }
```

Parameters

ZONE=
is the time zone difference from Greenwich Mean Time. If ZONE is not specified, a value of 0 hours (Greenwich Mean Time) is used.

The variable *h* is a number that represents hours. It can have a value from 0 to 13, but when coupled with the *m* and *s* fields, the total effective zone difference must not exceed 13 hours. The variable *m* is a number that represents minutes. The variable *s* is a number that represents seconds.

LOC=
specifies whether the time zone difference is to be taken EAST or WEST of Greenwich Mean Time. The default value for LOC is EAST. When the effective value of ZONE is 0, the setting of LOC is meaningless.

ID=
is the name of the time zone. The default for ID is GMT. The variable *xxx* is a three-character string.

When you want to change the time zone (for instance, from Eastern Standard Time to Eastern Daylight Time), change the ID and ZONE parameters, assemble the DMKSYS file again, and rebuild CP. The time zone change will take effect when you IPL the new CP system. Until the change takes effect, the system operator should use the old time zone when setting the time of day clock. (In the above example, when the operator's watch says 9:30 EDT, the time of day clock should be set to 8:30 EST.)

Examples:

The following examples show how to code the SYSTIME macro for many different time zones.

```
SYSTIME ZONE=5,LOC=WEST,ID=EST  (Eastern Standard Time)
SYSTIME ZONE=4,LOC=WEST,ID=EDT  (Eastern Daylight Time)
SYSTIME ZONE=6,LOC=WEST,ID=CST  (Central Standard Time)
SYSTIME ZONE=7,LOC=WEST,ID=MST  (Mountain Standard Time)
SYSTIME ZONE=1,LOC=EAST,ID=SET  (Standard European Time)
SYSTIME ZONE=1,LOC=EAST,ID=BST  (British Summer Time)
SYSTIME ZONE=10,LOC=EAST,ID=EST (Australian Eastern Standard Time)
```

SYSMON Macro

The SYSMON macro starts daily automatic performance data collection with the VM Monitor. The Virtual Machine Facility/370: Performance/Monitor Analysis Program is equipped with a front end assembly language routine that contains the appropriate diagnose commands to read the file and perform data reduction.

Format

```
SYSMON   [USERID= {OPERATOR}]
                  {userid  }

         [,CLASS= {M    }]
                  {class}

         [,AUTO= {NO }]
                 {YES}

         [,ENABLE= {(PERFORM,USER,DASTAP) }]
                   {(classa,classb,classc,...)}

         [,TIME= {(09:00,17:00)}]
                 {(h1:m1,h2:m2)}
                 {ALL          }
                 {NONE         }

         [,LIMIT= {(50000,NOSTOP)}]
                  {(limit, STOP) }
                  {(limit, NOSTOP)}
                  {(limit, SAMPLE)}

         [,BUFFS= {CPU}]
                  {n  }
```

Parameters

USERID=
: is the user ID of the virtual machine that will receive the monitor spool file in its virtual reader. The default is OPERATOR but any system directory entry may be specified.

CLASS=
: specifies the spool file class to be generated to contain monitor data. Any class (A through Z and 0 through 9) may be used but the default M is preferred because the VMAP data reduction Field Developed Program is designed to reduce only spool files of that class.

AUTO=
: specifies whether or not automatic monitoring should take place according to remaining SYSMON parameter specifications. The default, NO, requires you to make a specific change to cause automatic monitoring. All other parameters may be system default values, giving positive and useful monitoring results.

ENABLE=
: specifies any combination of monitor classes of data collection. It is assumed that the system analyst understands the use of various classes, overhead resulting from data collection, and relative magnitude of the corresponding data reduction. The default specifies sampled data classes only and is considered the least that can be specified for useful data reduction. The default classes are

sufficient for analysis of a system's load and performance profile with a view to diagnosis of possible bottlenecks and for establishing long term growth patterns.

TIME =
specifies the time period in each day that automatic monitoring (performance data collection) should take place. This parameter may indicate a start and stop time in hours and minutes using a 24-hour clock. Continuous monitoring (if ALL is specified), or no monitoring (if NONE is specified) occurs unless the operator or system analyst overrides this specification with the MONITOR command. If a system restart occurs during an automatic monitoring period, the old spool file is closed out and a new one is started, according to the SYSMON specifications. For useful data reduction, many hours of monitoring are suggested.

Note: This same closeout occurs at midnight if ALL is specified.

LIMIT =
specifies the maximum number of monitor record buffers that can be added to the monitor spool file before it is closed, and whether or not monitoring should be stopped when the limit is reached or the periodic closing of the monitor spool file after a specified number of samples (also defined by the value of LIMIT) have been collected. This parameter gives you more control over the amount of spool space that can be used by the automatic monitoring facility. It can also create many small monitor spool files, rather than one large file, and give the data reduction facility an opportunity to start processing the morning's data while collecting the afternoon's data. The 'limit' value can be any decimal number between 10 and 50000. When determining the value for 'limit', take into consideration the classes of data collection enabled, the size of the associated records, and the sampling interval. Remember that each monitor buffer contains approximately 4000 bytes of data space.

Specifying SAMPLE lets your analyst define the rate at which spool files will be produced. Because sampled data is collected at very precise intervals of time, according to the value specified in the MONITOR INTERVAL command (default 60 seconds), the spool file may be consistently and repeatedly closed. Monitor spool files obtained in this manner contain performance data covering consecutive, and equal intervals of time. This data contains the same number of PERFORM, DASTAP, and, possibly, USER (if no users logged on or off) records. This capability could form the basis of a real time performance analysis facility.

BUFFS =
specifies the number of data collection buffers needed by the monitor to avoid suspension incidents. Data collection suspension occurs when output to tape or spool files cannot keep ahead of data collection, and an overrun condition occurs. By increasing the number of monitor buffers, suspension incidents can be reduced or eliminated. The default depends on the storage size of the processor on which the system is running. (See the *VM/SP CP System Command Reference* for a description of the MONITOR command.) If not satisfied with the defaults, you may specify any number of buffers from 1 to 10.

Example:

```
SYSMON USERID=ANALYST,AUTO=YES,ENABLE=(PERFORM),        X
      TIME=ALL,BUFFS=1
```

This example specifies automatic monitoring for 24 hours a day using only the PERFORM class of data collection and one buffer. The spool file created is practically unlimited in size, taking the 50000 default and will be sent to the ANALYST virtual machine's reader each midnight, at system restart, or shutdown. M is the default spool file class.

Note: All of the preceding automatic monitoring specifications may be overridden by the operator or system analyst using the MONITOR command.

SYSJRL Macro

The SYSJRL macro specifies the inclusion of the journaling and/or password suppression facility.

Format

```
SYSJRL    [,JOURNAL= {NO      }]
                     {YES     }
          [,STQUERY= {NO      }]
                     {YES     }
          [,LOGUID=  {OPERATOR}]
                     {userid  }
          [,LOGLOC=  {(10,60) }]
                     {(n,m)   }
          [,LOGLMT=  {(2,3,4) }]
                     {(x,y,z) }
          [,LNKUID=  {OPERATOR}]
                     {userid  }
          [,LNKLMT=  {(2,5,10)}]
                     {(x,y,z) }
          [,PSUPRS=  {NO      }]
                     {YES     }
```

Parameters

JOURNAL=
: indicates whether or not the journaling facility is to be operative in the system being generated.

STQUERY=
: indicates whether or not the ability to SET and QUERY the journaling function should be a part of the system being generated. STQUERY=YES may be specified only if JOURNAL=YES is also specified.

LOGUID=
: is the user ID that should receive the indication that an invalid logon password count has been reached or exceeded. If this user ID is disconnected or logged off, the operator will receive the message generated.

LOGLOC=
: is the logon inductor. *n* is the maximum number of invalid password attempts. When *n* is exceeded, an error message is issued. *m* is the delay time, in minutes, until the next logon attempt. If you exceed *n* logon attempts and try logging on before *m* minutes have passed, the same error message will be issued. Both the user ID and terminal will be locked out.

Note: *n* may be any decimal number between 1 and 255. *m* may be any decimal number from 0 to 255. The defaults are 10 for *n* and 60 for *m*.

<u>**LOGLMT =**</u>
is the invalid LOGON/AUTOLOG threshold specification. The value specified applies to all unsuccessful LOGON attempts on a single terminal. *x* is the value which, when reached or exceeded, causes a type 04 accounting record to be generated for that and each following LOGON/AUTOLOG containing an invalid password. *y* is the value which, when reached or exceeded, causes a message to be sent to the user ID specified by LOGUID for that and each following LOGON/AUTOLOG containing an invalid password. *z* is the value which, when reached, causes the LOGON/AUTOLOG command to be disabled. A new VM logon screen will be displayed and a new logon sequence can be started.

Note: *z* replaces the present fixed limit of 4 and may be any decimal number from 1 to 255. *x* and *y* may be any decimal number from 0 to 255. 0 is a special case that indicates the applicable function should be bypassed. For example, if LOGLMT = (0,5,5) is specified, no accounting records are generated.

<u>**LNKUID =**</u>
is the user ID that should receive the indication that an invalid link password count has been reached or exceeded. If this user ID is disconnected or logged off, the operator will receive the message generated.

<u>**LNKLMT =**</u>
is the invalid LINK password threshold specification. The value specified applies to a single user ID for a single LOGON session. *x* is the value that, when reached or exceeded, causes 06 accounting record to be generated for that and each following LINK containing an invalid password. *y* is the value that, when reached or exceeded, causes a message to be sent to the user ID specified by LNKUID for that and each following LINK containing an invalid password. *z* is the value that, when reached, causes the LINK command to be disabled for the current LOGON session.

Note: *z* replaces the current fixed limit of 10 and may be any decimal number from 1 to 255. *x* and *y* may be any decimal number from 0 to 255. 0 is a special case that indicates the applicable function to be bypassed. For example, if LNKLMT = (2,0,10) is specified, no message is sent.

<u>**PSUPRS =**</u>
indicates whether or not the facility that suppresses the password on the command line should be part of the system being generated.

Note: If PSUPRS = YES is specified for a 2741 terminal, passwords will be typed upon a mask. Specifying PSUPRS = YES provides additional security for your VM/SP installation.

SYSACNT Macro

The SYSACNT macro specifies spooling of accounting records.

Format

label	SYSACNT	[USERID = { OPERATOR | *userid* }
		,OUTPUT = { PUNCH | READER }
		,CLASS = { C | *class* }
		,LIMIT = { 0 | *nnnnn* }]

Parameters

label
: is any desired label.

USERID =
: is the virtual machine identification to which all files are spooled.

OUTPUT =
: is the type of spool file to be created. PUNCH is the default type.

CLASS =
: is the class of the spool file to be created. A to Z and 0 to 9 can be specified. Class C is the default.

LIMIT =
: is the number of records to be accumulated before the file is closed and made available to the virtual machine. *nnnnn* is up to 5 decimal digits from 0 to 32,767.

  The default, 0, indicates that the file is not to be automatically closed unless the number of records reaches 32,767.

Accounting records are accumulated in a spool file identified by the USERID and CLASS parameters until either the number of records specified in the LIMIT parameter is reached or until the ACNT command is issued with the CLOSE operand. At that time, the records are sent to the virtual punch or reader, as specified in the OUTPUT parameter.

If accounting records are stored in reader files in your virtual machine, they should be processed periodically to avoid accumulating and tying up system resources.

Note: The accounting records can be moved to tape by way of the SPTAPE command for later processing.

SYSFORM Macro

The SYSFORM macro specifies the following:

- A list of user forms with their corresponding operator forms. Operator forms can also be specified as NARROW, so that a narrow (94-character) separator page is printed.
- Default user forms for printer, punch, and console. (The default operator forms are obtained from the list of user/operator forms.) These defaults apply:
  - When creating a virtual printer, punch, or console spool file, unless you have overridden the default with a SPOOL or CLOSE command.
  - When the operator enters a START command for a printer or punch without specifying a form, and this is the first START command since cold starting VM/SP.

Format

SYSFORM	[(*userform,operform* [,NARROW]) ...] [,DEFPRT=*userform*] [,DEFPUN=*userform*] [,DEFCON=*userform*]

Parameters

userform
specifies the one-to-eight character user form name. Use this form in CP SPOOL, CLOSE, CHANGE, PURGE, ORDER, QUERY, and TRANSFER commands.

operform
specifies the one-to-eight character name for the corresponding operator form.

If no *userform* or *operform* pairs are specified, no form list is generated. In this case, no distinction is made between user forms and operator forms.

NARROW
specifies printing on narrow (94-character) paper. Separator information will not print beyond this width.

DEFPRT
specifies the default user (and implicitly, operator) forms when creating virtual printer spool files, or when starting the real printer.

If DEFPRT is not specified, the default is DEFPRT=STANDARD.

DEFPUN
specifies the default user (and implicitly, operator) forms when creating virtual punch spool files, or when starting the real punch.

If DEFPUN is not specified, the default is DEFPUN=STANDARD.

DEFCON
specifies the default user (and implicitly, operator) forms when creating virtual console spool files.

If DEFCON is not specified, the default is DEFCON=STANDARD.

Examples:

1. Default case.

   ```
   SYSFORM
   ```

   No user/operator form list, therefore no distinction between user forms and operator forms.

   Default forms are STANDARD for printer, punch, and console.

2. Defaults specified.

   ```
   SYSFORM DEFPRT=VANILLA,DEFPUN=WHITE
   ```

   No user/operator form list, therefore no distinction between user forms and operator forms.

   Default forms are VANILLA for the printer, WHITE for the punch, and STANDARD for the console.

3. User/operator form list.

   ```
   SYSFORM (LISTING,QN-8-14),                                X
         (DOCUMENT,TN-6-8,NARROW),                           X
         (OUTPUT,PN-6-14),                                   X
         DEFPRT=OUTPUT,DEFCON=DOCUMENT
   ```

   User form LISTING corresponds to operator form QN-8-14.

   You enter the command SPOOL PRINTER FORM LISTING, and the operator sees form QN-8-14. Likewise, user form DOCUMENT corresponds to the narrow operator form TN-6-8, and OUTPUT corresponds to PN-6-14. All other forms have the user form equal to the operator form.

   OUTPUT/PN-6-14 is the default form for the printer. DOCUMENT/TN-6-8 is the default form for console spool files. STANDARD is the default for the punch.

SYSPCLAS Macro

The SYSPCLAS macro classifies printed output with a classification title. This title is printed on the output separator page, and optionally at the top or bottom of each page of output.

You can specify a different classification title for each output class (A to Z and 0 to 9), providing up to 36 different classifications.

The macro specifies a list of class/title pairs. The TOP or BOTTOM option can be specified for any pair. The TOP option specifies that this title is to be printed at the top of all pages of output, not just the separator page. The BOTTOM option prints the title at the bottom of all output pages.

Format

```
SYSPCLAS | [(c, 'title'  [,TOP    ]   ) ... ]
         |               [,BOTTOM ]
```

Parameters

c
: is a one-character spool file class. It must be alphanumeric.

'title'
: is a classification title for this class. It may be up to 46 characters long.

  The title may contain any characters. It must be enclosed in quotes (' '). Quotes within the title are coded as two consecutive quotes, and ampersands (&) are coded as two consecutive ampersands.

TOP
: specifies that this title is to be printed both on the separator page, and at the top of each page of output.

BOTTOM
: specifies that this title be printed both on the separator page, and at the bottom of each page of output.

  See "Usage Notes" on page 354 for more information.

Examples:

1. No classifications desired.

   ```
   SYSPCLAS
   ```

   No classification titles are generated.

2. Classifications generated.

   ```
   SYSPCLAS (R,'CUSTOMER CONFIDENTIAL'),                   *
            (N,'COMPANY NAME',TOP)
   ```

   Class R output is printed with CUSTOMER CONFIDENTIAL on the separator page. Class N output has COMPANY NAME on the separator page, and at the top of each page of output.

Usage Notes

1. The SYSPCLAS macro replaces previous support for class X files in module DMKBOX.
2. The title line is inserted at the top or bottom of each page. To do this, CP inserts a CCW to print one space, followed by the title line before or after each "skip to channel 1" CCW issued by the guest operating system. There are many things to consider about this:
   a. The titles are inserted as the file is being created. If the file is later changed to a different class, the titles in the file are not changed. However, the title on the separator page always reflects the true class of the file.
   b. Inserting the title on the output page adds a line of output to the printed page. If the guest application is counting lines, its count will be incorrect. This can result in alternating output and blank pages.
   c. If the guest does not use "skip to channel 1," the title lines are never printed.
   d. If the guest does not issue "skip to channel 1" at the end of the file, and the BOTTOM option is being used, the last page will not have a classification title on it. Usage of the TOP option keeps this from happening.

SYSIPL Macro

The SYSIPL macro lets an installation specify the type of start to be attempted to re-IPL the system after a power outage. If no operand is specified or if the SYSIPL macro is not specified, the system does not automatically initialize itself.

Format

SYSIPL	[SYSTYPE=WARM|CKPT|FORCE|COLD]

Parameters

SYSTYPE
specifies the last restart type that VM/SP will automatically attempt.

WARM
indicates that a WARM start is attempted when IPLing after a power outage.

CKPT
indicates that a WARM start is attempted when IPLing after a power outage followed by a CKPT start if WARM fails.

FORCE
indicates that a WARM start is attempted followed by a CKPT start if WARM fails followed by a FORCE start if CKPT fails.

The FORCE parameter is not recommended because it can cause a loss of spool files.

COLD
indicates that a COLD start is attempted if WARM, CKPT, and FORCE fail. WARM, CKPT, and FORCE will always precede COLD.

The COLD parameter is not recommended because all spool files will be lost.

Usage Notes

1. Due to the severity of a FORCE start and a COLD start, it is recommended that a WARM start or a CKPT start ensure spool file integrity when using the SYSIPL macro to automatically re-IPL after a power outage. The CKPT start should be specified because it is very unlikely that a WARM start will succeed.

   If no arguments are specified or if the SYSIPL macro is not specified, the system requires operator intervention (IPL prompts) after a power outage.

2. The SYSIPL macro should be specified once in the DMKSYS module. Otherwise, an MNOTE will be generated during the assembly of module DMKSYS and the second and any subsequent invocations do not change the settings of the first invocation.

3. The automatic re-IPL function should not be used unless the hardware has a battery operated clock.

4. An automatic re-IPL can occur if the SYSIPL macro is specified with a valid parameter and if:

   a. A power outage occurs and an IPL with the *clear* option is done, or

   b. The operator stops the system and an IPL with the *clear* option occurs.

   The first start type to be attempted is WARM followed by CKPT, FORCE, and COLD until either one succeeds or until the type specified on the SYSIPL macro fails. If the hardware does not have a battery operated clock and an automatic re-IPL is specified and started, the results are as follows:

not set/stopped
If there is an indication that the clock is not set or that the clock is in stopped state, the IPL proceeds but it is not an automatic re-IPL. The operator is prompted for the clock value and type of start. When initialization completes, the operator is disconnected.

set
If there is an indication that the clock is set, the automatic re-IPL proceeds without prompting, but the clock value is not correct unless the outage was not due to a power outage; for example, the operator pressing the STOP button.

not operational
If the clock is not operational, an abend occurs as before.

SYSID Macro

The SYSID macro lets you identify a system with an eight-character identification, called the system ID. The system ID appears in the status area of the display terminal and is printed on the output separator page.

The processor ID is specified with a model number and a serial number. If you have more than one VM/SP processor, a list of system IDs with corresponding processor model and serial numbers can be specified. When VM/SP is initialized by way of an IPL, the initialized processor is matched with an entry in the list of system IDs, thus identifying the local system ID. The processor ID of the IPLed processor is obtained with the STORE CPUID instruction. If no match is found, a default system ID is assumed.

Format

SYSID	[(*systemid,model,serial*),...] [,DEFAULT=*defid*]

Parameters

systemid
specifies the one-to-eight character system ID that identifies a uniprocessor system.

model
designates a three-to-four digit number that specifies the processor model number (for example 158, 3031, or 4341) for a multiprocessor system.

serial
is a five or six-digit number that specifies the processor serial number corresponding with the system ID. An AP user should use the serial number and model number of the main (IPL) processor in the SYSID statement. For an MP user, use the serial and model number of both processors in the SYSID macro.

DEFAULT=*defid*
designates the one-to-eight character default system ID. This ID is selected if the processor is not found in the list, or if there is no list.

Examples:

1. Default case.

   ```
   SYSID
   ```

   No system ID is printed on the separator page.

2. Single processor user.

   ```
   SYSID DEFAULT=CUSTOMER
   ```

   Gives this VM/SP system the ID "CUSTOMER." This ID will be printed on the separator page.

3. Multiple processor user.

   ```
   SYSID (CUSTSYS1,158,13289),                        X
         (CUSTSYS2,4341,23145),DEFAULT=OTHERSYS
   ```

If this system is IPLed on the 370/158 serial number 13289, the system ID is CUSTSYS1. If it is IPLed on the 4341 serial number 23145, the system ID is CUSTSYS2. If it is IPLed on any other system, the system ID is OTHERSYS.

SYSORD Macro

The SYSORD macro specifies the order in which preferred and/or nonpreferred paging and spooling devices are searched for available pages to satisfy paging/spooling operations. Different search orders may be specified for preferred and nonpreferred devices. SYSORD is a required macro statement. It must be included in DMKSYS before the SYSLOCS macro statement.

Format

SYSORD	[SYSFH= (*devtype*[,*devtype*,...[,(*devtype*,*devtype*,...)]])] [SYSMH= (*devtype*[,*devtype*,...[,(*devtype*,*devtype*,...)]])] [SYSTEMP=(*devtype*[,*devtype*,...[,(*devtype*,*devtype*,...)]])]

Parameters

devtype
is a supported DASD type.

SYSFH=
designates the priority order for searching device types that have preferred fixed-head paging (PAGE) cylinders or extents defined. A specified device type indicates a single device or many devices of the same device type in a chain. Supported fixed-head device types (in order of decreasing performance) are: 2305, 3350, 3330, and 3340. A device type may appear only once within this operand.

Note: The 3330 specification for SYSFH is included only for fixed-head 3350s running in 3330 compatibility mode.

SYSMH=
designates the priority order for searching device types that have preferred moveable-head paging (PAGE) cylinders or extents defined. A specified device type indicates a single moveable-head device or many moveable-head devices of the same device type in a chain. Supported moveable-head device types (in order of decreasing performance) are: 3380, 3375, 3370, 3310, 9335, 9332, 9313, 3350, 3330, and 3340. A device type may appear only once within this operand.

SYSTEMP=
designates the priority order for searching device types that have nonpreferred (TEMP) cylinders or extents defined. TEMP space is used for overflow paging operations and all spooling operations. A specified device type indicates a single fixed or moveable-head device or many devices of the same device type in a chain. Supported device types (in order of decreasing performance) are: 2305, 3380, 3375, 3370, 3310, 9335, 9332, 9313, 3350, 3330, and 3340. A device type may appear only once within this operand.

For *each* operand, the three options indicated are:

1. The option to specify the operand. If specified, at least one device type is required.

   Example:

   SYSFH=(devtype)
   SYSMH=(devtype)
   SYSTEMP=(devtype)

2. The option to specify more than one device type.

   Example:

   SYSFH = (devtype,devtype,...)
   SYSMH = (devtype,devtype,...)
   SYSTEMP = (devtype,devtype,...)

3. The option to specify for devtype, a list of device types indicating that the device types in the inner parentheses are to have equal priority with each other.

   Example:

   SYSFH = (devtype,(devtype,devtype,...),...)
   SYSMH = (devtype,(devtype,devtype,...),...)
   SYSTEMP = (devtype,(devtype,devtype,...),...)

A request for a page of external page space is satisfied by allocating space on a preferred paging cylinder or extent, provided that one exists on the system and that it has a page slot available. You specify preferred and nonpreferred paging space by using the PAGE and TEMP operands of the CP Format/Allocate service program.

Within the class of CP-owned volumes with preferred space, pages are first allocated from volumes with fixed-head cylinders or extents in the order specified by the SYSFH operand. (If SYSFH is not specified, the default order is used.) If no fixed-head space is available, then pages are allocated from volumes with moveable-head cylinders or extents in the order specified by the SYSMH operand. (If SYSMH is not specified, the default order is used.) Finally, if no preferred space is available, pages are allocated from volumes with nonpreferred space in the order specified by the SYSTEMP operand. (If SYSTEMP is not specified, the default order is used.)

Spooling space is not allocated from preferred space. Spooling space is allocated from volumes with nonpreferred space in the order specified by the SYSTEMP operand. (If SYSTEMP is not specified, the default order is used.)

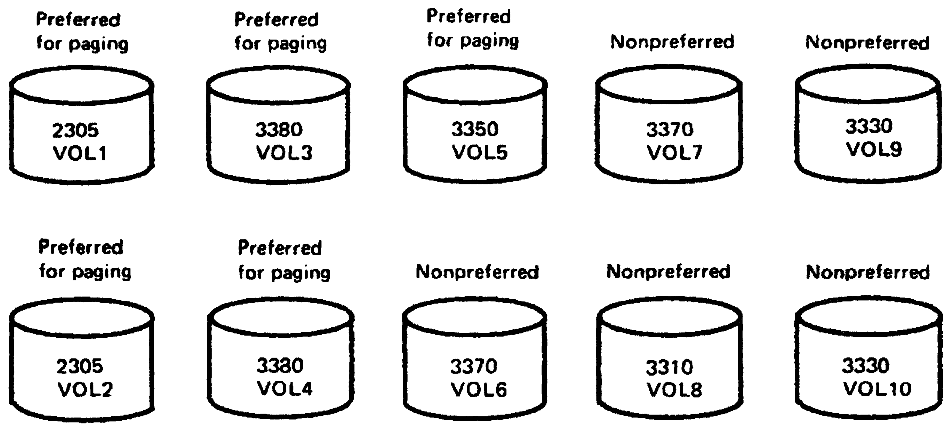
A rotary paging scheme evenly distributes external DASD pages across all volumes of a specific device type and to allow concurrent paging operations. For *each* SYSFH/SYSMH/SYSTEMP operand, a device type order specification of (devtype,devtype,devtype,...) indicates a series of unique device types, where each device type may indicate a single device or a chain of devices of the same type. If a chain exists, pages are allocated on all devices within the chain on a rotating basis. When all available pages on the device chain are used, the next specified device type is examined for available pages. Allocation of DASD pages occurs in this manner until all device types specified within the parentheses are used. A device type order specification of (devtype,devtype,(devtype,devtype,...),...) indicates that the device types in the inner parentheses are to have equal priority with each other.

The same rotary scheme is used for the allocation of spooling space from nonpreferred devices.

The following example shows the sequence in which paging or spooling space would be allocated from all CP-owned volumes having cylinders or extents defined as specified in the SYSORD macro below. The example is meant to show the allocation technique, not necessarily the desirable allocation of paging or spooling space.

Remember that PAGE space on 2305 DASD (VOL1 and VOL2) is entirely fixed-head. It is solely moveable-head PAGE space on 3380s (VOL3 and VOL4). However, the 3350 (VOL5) has a mixture of both fixed and moveable-head PAGE space.

```
SYSORD SYSMH=(3380,3350) SYSTEMP=((3370,3310),3330)
```



Using the previous example, paging cylinders or extents would be allocated in the following sequence:

1. On an alternating basis on VOL1 and VOL2 until all fixed-head PAGE space is used on these volumes.
2. On VOL5 until all fixed-head PAGE space is used on that volume.
3. On an alternating basis on VOL3 and VOL4 until all moveable-head PAGE space is used on these volumes.
4. On VOL5 until all moveable-head PAGE space is used on that volume.
5. On an alternating basis on VOL6, VOL7, and VOL8 until all TEMP space is used on these volumes.
6. On an alternating basis on VOL9 and VOL10 until all TEMP space is used on these volumes.

Spooling cylinders or extents would be allocated in the following sequence:

1. On an alternating basis on VOL6, VOL7, and VOL8 until all TEMP space is used on these volumes.
2. On an alternating basis on VOL9 and VOL10 until all TEMP space is used on these volumes.

SYSORD is a required macro statement. Though the statement is required, the operands are optional. If SYSORD is specified without operands, the following default priority orders apply:

```
SYSFH=  (2305, 3350, 3330, 3340)
SYSMH=  (3380, 3375, 3370, 3310, 9335, 9332, 9313, 3350, 3330, 3340)
SYSTEMP=(2305, 3380, 3375, 3370, 3310, 9335, 9332, 9313, 3350, 3330, 3340)
```

A device type may appear only once within each SYSFH/SYSMH/SYSTEMP statement. If duplicate device types are specified within an operand, the following MNOTE is issued:

```
'SAME DEVICE TYPE SPECIFIED MULTIPLE TIMES'
```

You should eliminate the multiple specification and reassemble DMKSYS.

Specifying an operand other than SYSFH/SYSMH/SYSTEMP produces the following:

```
'KEYWORD PARAMETER 'parameter' UNDEFINED IN MACRO DEFINITION'
```

You have specified an invalid keyword. Only SYSFH, SYSMH, or SYSTEMP are allowed.

Specifying an invalid device type within any operand generates one of these MNOTES:

```
'INVALID DEVICE TYPE FOR SYSFH'
'INVALID DEVICE TYPE FOR SYSMH'
'INVALID DEVICE TYPE FOR SYSTEMP'
```

Missing parentheses or parentheses without following operands generate the MNOTE:

```
'POSITIONAL OR EMPTY PARMS NOT ALLOWED'
```

SYSFCN Macro (Optional)

The SYSFCN macro can change user privilege classes. Both IBM-defined and installation-defined classes having access to restricted internal control program functions can be altered. If you do not code the SYSFCN macro, all the default values will apply.

Whenever you redefine user privilege classes, be sure to change the SYSFCN macro if necessary.

Format

SYSFCN	$\left[\underline{\text{PRIV}}=\left\{\begin{matrix}\underline{\text{ABCDEF}}\\ cl\end{matrix}\right\}\right]$ $\left[\underline{\text{,OPER}}=\left\{\begin{matrix}\underline{\text{A}}\\ cl\end{matrix}\right\}\right]$ $\left[\underline{\text{,CPRD}}=\left\{\begin{matrix}\underline{\text{CE}}\\ cl\end{matrix}\right\}\right]$ $\left[\underline{\text{,CPWT}}=\left\{\begin{matrix}\underline{\text{C}}\\ cl\end{matrix}\right\}\right]$ $\left[\underline{\text{,SERV}}=\left\{\begin{matrix}\underline{\text{F}}\\ cl\end{matrix}\right\}\right]$ $\left[\underline{\text{,DFLT}}=\left\{\begin{matrix}\underline{\text{G}}\\ cl\end{matrix}\right\}\right]$

Parameters

PRIV=
: specifies the user privilege classes that are authorized to access a CCW operation code of X'42' to a 37xx emulation line for which the 37xx control unit is not dedicated to the issuing user. The default value is ABCDEF.

OPER=
: specifies the user privilege classes that are authorized to LOGON during initialization. The default value is A.

CPRD=
: specifies the user privilege classes that are authorized to issue IOCP READ instructions. The default value is CE.

CPWT=
: specifies the user privilege classes authorized to issue IOCP WRITE instructions. The default value is C.

SERV=
: specifies the user privilege classes that are authorized to issue Diagnostic Load/Write and Sense/Read commands. The default value is F.

DFLT=
: specifies the user privilege classes that will be assigned to a user by default if neither the class field in the USER control statement nor the CLASS control statement is coded. The default value is G.

cl is, in all cases, a string of 1 to 32 characters identifying user privilege classes. Each character identifies a single class. Valid class identification characters are the letters A to Z and the digits 1 to 6. The characters may be entered in any order, without duplication, and must not be separated by blanks.

Note: See the *VM/SP Administration* book for more information on privilege classes.

SYSMIH Macro (Optional)

The SYSMIH macro defines the time interval desired for missing interruption monitoring. I/O activity is monitored for the following devices:

- Direct access storage devices
- Graphic devices
- Tape devices
- Unit record devices
- Miscellaneous devices.

If you specify zero for a device group, monitoring is set off for that group.

Do not specify the name field for the SYSMIH macro instruction.

Format

SYSMIH	$\left[\underline{\text{DASD}}=\begin{Bmatrix}\underline{\mathbf{0{:}15}}\\ mm{:}ss\end{Bmatrix}\right]$	$\left[,\underline{\text{GRAF}}=\begin{Bmatrix}\underline{\mathbf{0{:}30}}\\ mm{:}ss\end{Bmatrix}\right]$
	$\left[,\underline{\text{TAPE}}=\begin{Bmatrix}\underline{\mathbf{10{:}00}}\\ mm{:}ss\end{Bmatrix}\right]$	$\left[,\underline{\text{UR}}\ =\begin{Bmatrix}\underline{\mathbf{1{:}00}}\\ mm{:}ss\end{Bmatrix}\right]$
	$\left[,\underline{\text{MISC}}=\begin{Bmatrix}\underline{\mathbf{12{:}00}}\\ mm{:}ss\end{Bmatrix}\right]$	

Parameters

<u>DASD=</u>
specifies the time interval value for count-key-data direct access storage devices (CLASDASD) and fixed block architecture devices (CLASFBA). You can specify a maximum value of 99 for *mm* (minutes) and a maximum value of 59 for *ss* (seconds). If you do not specify a value for this class, the time interval is set to the default value, 15 seconds.

<u>GRAF=</u>
specifies the time interval value for graphic devices (CLASGRAF). You can specify a maximum value of 99 for *mm* (minutes) and a maximum value of 59 for *ss* (seconds). If you do not specify a value for this class, the time interval is set to the default value, 30 seconds.

<u>TAPE=</u>
specifies the time interval value for tape devices (CLASTAPE). You can specify a maximum value of 99 for *mm* (minutes) and a maximum value of 59 for *ss* (seconds). If you do not specify a value for this class, the time interval is set to the default value, 10 minutes.

<u>UR=</u>
specifies the time interval value for unit record input devices (CLASURI) and unit record output devices (CLASURO). You can specify a maximum value of 99 for *mm* (minutes) and a maximum value of 59 for *ss* (seconds). If you do not specify a value for this class, the time interval is set to the default value, 1 minute.

<u>MISC=</u>
specifies the time interval value used for miscellaneous devices. Miscellaneous devices include MSS devices, CLASGRAF TYP328x and TYP1053, and CLASURO TYP3800 and TYP3289E printers. MSS devices include

CLASSPEC TYP3851 and CLASDASD FEATURE=SYSVIRT or FEATURE=VIRTUAL. You can specify a maximum value of 99 for mm (minutes) and a maximum value of 59 for ss (seconds). If you do not specify a value for this class, the time interval is set to the default value, 12 minutes.

Usage Notes

1. If you do not specify a value for a device class, CP uses the default time interval.
2. If you specify zero for a device group, monitoring is set off for that group. Specify zero for any device group that you do not use.
3. Use the SET MITIME command to change the time intervals of device classes. Use the QUERY MITIME command to determine the time intervals in effect.
4. If you specify a time interval for a device class below its default value, be careful not to shorten the time interval too much because this may cause unnecessary missing interruption handler processing for devices that are functioning properly.
5. If you specify more that one time interval for a device class, the last value coded is used.
6. If you remove module DMKDID from the load list, and later enter the SET MITIME or QUERY MITIME command, CP issues a message that missing interruption monitoring is not available.
7. If you specify an invalid time value in the SYSMIH statement, the time value is set to the default, return code 4 is generated, and the following MNOTE is issued:

   ```
   INVALID TIME VALUE SPECIFIED FOR class - TIME SET TO time
   ```

   Here, class indicates the device class that has the invalid time value and time indicates the default value for this class.

Example:

This example shows the SYSMIH macro instruction can:

- Set a 15-second time interval for direct access storage devices
- Set a 20-second time interval for graphic devices
- Disable I/O monitoring for tape devices
- Set a one-minute time interval for unit record processing
- Disable I/O monitoring for miscellaneous devices.

```
SYSMIH GRAF=00:20,TAPE=00:00,MISC=00:00
```

SYSLOCS Macro

The SYSLOCS macro generates internal pointer variables. This must be the last macro in the DMKSYS file.

The name field must not be specified for the SYSLOCS macro instruction. No operands are required for the SYSLOCS macro. If one is specified, it is ignored.

Format

SYSLOCS	

Example of Coding the CP System Control File

Figure 42 shows a sample CP system control file (DMKSYS).

```
SYS     TITLE   'DMKSYS   FOR  3380  VM/SP'
DMKSYS CSECT
       SYSOWN  VMSRES,                                                X
               VMPK01
       SYSRES  SYSVOL=VMSRES,                                         X
               SYSRES=123,                                            X
               SYSTYPE=3380,                                          X
               SYSCLR=YES,                                            X
               SYSNUC=(1,5),                                          X
               SYSWRM=(17,2,VMSRES),                                  X
               SYSERR=(19,2,VMSRES),                                  X
               SYSCKP=(346,1,VMSRES)
       SYSMON  USERID=VMMAP,                                          X
               AUTO=NO,                                               X
               BUFFS=2,                                               X
               TIME=(08:00,17:00),                                    X
               CLASS=M,                                               X
               ENABLE=(PERFORM,USER,DASTAP),                          X
               LIMIT=(50000,NOSTOP)
       SYSJRL
       SYSCOR  RMSIZE=8M,                                             X
               AP=NO,                                                 X
               MP=NO
       SYSOPR  SYSOPER=OPERATOR,                                      X
               SYSDUMP=OPERATNS
       SYSACNT USERID=DISKACNT,                                       X
               OUTPUT=READER,                                         X
               CLASS=C,                                               X
               LIMIT=100
       SYSTIME ZONE=4,                                                X
               LOC=WEST,                                              X
               ID=EDT
       SYSFORM
       SYSPCLAS
       SYSID
       SYSORD
       SYSMIH
       SYSFCN
       SYSIPL
       SYSLOCS
       END
```

Figure 42. CP System Control File Sample (DMKSYS)

Chapter 10. Preparing the System Name Table File (DMKSNT)

Contents of Chapter 10

DMKSNT Overview

The System Name Table (DMKSNT) is a facility that lets many users share a single *real* copy of program code and IPL an already-initialized, shareable nucleus. As a result, system resources are better used.

Defining Saved Systems and Saved Segments

The System Name Table is made up of three major components. Their names and definitions are:

Saved systems are systems that have been saved on a disk along with all the information you need to resume execution. Saved systems provide an efficient means of IPLing systems by bypassing many system initialization steps.

Saved segments are virtual storage areas of a virtual machine. These segments can contain read only data or reentrant code that many users can share. Saved segments provide an efficient means of getting programs by merely connecting segments to a virtual machine.

Shared segments are segments within a saved system or saved segment. These segments can contain read only data or reentrant code that many users can share. This reduces the demand for real storage for the overall system.

Saved Systems

Saved systems provide an efficient means of IPLing systems by bypassing many system initialization steps. A saved system contains a copy of virtual machine storage, register contents, PSW, and storage keys. The system is not loaded at IPL time. Instead, the system initializes page tables, and brings pages into storage as needed during execution.

Because a saved system can share segments of reenterable code, real storage is used more efficiently.

Note: You cannot IPL a saved system in a virtual=real machine.

Saved Segments

Saved segments are virtual storage areas of a virtual machine that:

- Have a name associated with them
- Contain read only data or reenterable code
- Were previously loaded and saved
- Can be shared by multiple users
- Can be loaded by a particular virtual machine in nonshared mode for testing and debugging.

Saved segment support lets a virtual machine logically attach and detach segments of storage. Many users can share these segments.

The saved segment should not overlay any other named segment that can be attached at the same time.

Installing the Newly Defined Saved Segment or Saved System

1. Be sure that the volume containing the segments is CP owned (listed in the SYSOWN macro for the DMKSYS file).

2. Reserve the allocated DASD space with a dummy minidisk in the $SAVSYS$ user ID of the CP directory. Run the CP Format/Allocate program to format the cylinders and to allocate them as PERM.

   At this point you can use the SNTMAP EXEC to determine if the defined $SAVSYS$ areas contain gaps, which can be used for new or expanded saved segment DASD areas. See "SNTMAP EXEC" on page 373 for a description of this exec.

3. Update the DMKSNT assemble file with the appropriate macros:

   - NAMESYS creates an entry in the system name table for a virtual machine or saved segment
   - NAMENCP creates an entry in the system name table for a 3704/3705 control program
   - NAME3800 creates an entry in the system name table for a 3800 printer image library
   - NAMELANG creates an entry in the system name table for a CP message repository.

   See the section "Preparing the System Name Table File (DMKSNT)" on page 377 for more details about these macros.

4. Assemble the DMKSNT file.

   If you get a clean assembly, you should run the SNTMAP EXEC again to verify there are no DASD overlays in your macro specifications.

5. Build and test the new CP nucleus that contains the new definition.

6. Load the information into virtual storage and save the virtual storage area in a saved segment.

   You can do this by one of three ways:

   Manually:

   a. Load the data into virtual storage with the LOAD command or the STORE command.
   b. Assign storage protection with the SETKEY command.
   c. Save the virtual storage area as a physical saved segment with the SAVESYS command.

   For more information about the LOAD command, see the *VM/SP CMS Command Reference.* For more information about the SETKEY command, see the *VM/SP Administration* book. For more information about the STORE command, see the *VM/SP CP General User Command Reference.* For more information about the SAVESYS command, see the *VM/SP CP System Command Reference.*

7. Use a tool that loads and saves a specific type of data in a physical saved segment:

   - SAVEFD loads and saves a shared minidisk directory.
   - LANGGEN loads and saves national language text files.
   - DOSGEN loads and saves CMS/DOS text files.

- SAMGEN loads and saves simulated VSE modules.
- VSAMGEN loads and saves VSAM or AMS files.
- DCSSGEN loads and saves execs and editor macros.

For more information about the SAVEFD command, see the *VM/SP Administration* book. For more information about the LANGGEN command, see the *VM System Facilities for Programming* book. DOSGEN, SAMGEN, VSAMGEN, and DCSSGEN are described in the *VM/SP Installation Guide*

8. Use the SEGGEN command to load and save the data in a logical saved segment contained in a physical saved segment. A logical saved segment can contain many types of data.

For more information about the SEGGEN command and defining logical saved segments, see the *VM/SP Application Development Guide for CMS.*

How to Save a System with the SAVESYS Command

To save a system, follow the following steps:

1. Load the system to be saved by device address. A system should be saved as soon after IPL as possible.
2. Make sure that the system to be saved has stopped before proceeding. The system must stop executing before the page-format image can be saved. All pages to be saved must be resident before doing the next step, issuing the SAVESYS command. Your installation programmer should determine where to stop the operating system.
3. Enter the SAVESYS command causing CP to save a copy of virtual machine storage, register contents, PSW, and storage keys. For more information on the SAVESYS command, see the *VM/SP CP System Command Reference.*
4. Now you and other users can IPL the new saved system by the system name rather than by device address. The virtual machine tries to resume execution and immediately encounters a page fault. The required page is brought into storage and execution continues. As execution continues, subsequent page faults bring the required pages into storage.

   CMS runs under and can easily be saved by CMS. CMS recognizes a special parameter on the IPL command and issues the SAVESYS command on your behalf at the appropriate point in initialization. See the section on saving CMS in the *VM/SP CMS for System Programming* and the *VM/SP Installation Guide* for more details on saving CMS.

   Note: You can save up to a 16MB virtual machine.

For information on saving IBM supplied saved systems and saved segments, see the *VM/SP Installation Guide.*

Considerations

The following might cause errors while saving a saved segment:

- Formatting DASD improperly. This can be corrected by rerunning the CP Format/Allocate program to format the cylinders to receive the saved segment.
- A mismatch between the SYSPGNM specification and the load address used when saving the system
- Failure to account for the *extra* CP information pages when allocating DASD space for the saved segment

- Forgetting to save the system before attempting to use it (some software will attempt to use the saved segment as soon as it is defined)
- Programming errors in the software saved into the saved segment.

Shared Segments

Shared segments are segments within a saved system or a saved segment that users can share. They contain read only data and reentrant code. If you designate one or more segments of a saved system as "shared," a virtual machine that loads the saved system by name can use these segments in real storage. This reduces the real storage demand for the overall system.

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

Usage Notes

- A shared segment must be reenterable.
- You must include the segment number in the SYSHRSG operand of the NAMESYS macro for the saved system.
- You cannot use shared segments in a virtual=real machine.
- The maximum number of segments you can define in a saved system or saved segment is 78.
- You must be careful when you resave a saved system that contains one or more shared segments.

  If the previous system is loaded by name and is still in use, problems can occur. If you use the *old* system and continue to reference pages already in paging storage, then there is no problem. However, if you use the old system and reference pages not previously referenced (in the newly saved system), then you receive the **new** versions of pages you reference.
- When you IPL the newly saved system, you share only the new copy of the shared segment.
- Using the SAVESYS command saves the entire segment, not just that portion the program occupies (for example, CMS). Thus, if there is any unwanted data, it is contained in that segment.

SNTMAP EXEC

The SNTMAP EXEC processes the macro definitions in a system name table (SNT) file. SNTMAP produces two CMS output files:

- DASD SNTMAP A - a saved segment DASD map
- MEMORY SNTMAP A - a virtual storage map.

Format

```
SNTMAP    ┌                             ┐
          │ fn      ┌ ft       ┐┌ fm ┐  │
          │         │ ASSEMBLE ││ *  │  │
          │         └          ┘└    ┘  │
          │                             │
          │ ┌ HELP ┐                    │
          │ │  ?   │                    │
          └ └      ┘                    ┘
```

Operands

fn

is the file name of the SNT input file. The file name must be entered, either when issuing the command or as a reply to a prompt from SNTMAP. Do not use * as the file name.

ft

is the file type of the SNT input file. Do not use * as the file type. If you do not enter the file type, SNTMAP assumes that the file type is ASSEMBLE.

fm

is the file mode of the SNT input file. If you do not enter the file mode, SNTMAP assumes that the file mode is *.

HELP
?

requests the HELP file for the SNTMAP command. This file is located on the same file mode as the SNTMAP EXEC file, rather than with the system HELP files.

Input Files

SNT source file: This is the main input from the user. The IBM-supplied SNT source file is DMKSNT ASSEMBLE. It contains the NAMESYS, NAMENCP, NAME3800, and NAMELANG macros. SNTMAP assumes that you have assembled the file and found no syntax errors.

Grouping Macros into Families: To make a NAMESYS macro a member of a family, insert a comment line in the DMKSNT file just before the NAMESYS macro. The format of the comment line is:

```
* Family = f[f[...]]
```

"f" is any single character representing a particular family. Members of the same family will use the same character. You can include a macro in more than one family. If, for example, you want to include a macro in families "A," "C," and "7," insert this line just before the NAMESYS macro:

```
* Family = AC7
```

As a result of adding this comment line, the macro will appear in the main memory map along with all the other NAMESYS macros, and it will appear in the memory maps for families A, C, and 7.

CP directory: After you invoke SNTMAP, the EXEC prompts you to enter the name of the CP directory. The directory contains a USER $SAVSYS$ entry that describes the DASD areas allocated for SNT data. Within $SAVSYS$, the MDISK statements must be listed in ascending order by cylinder or block number. If SNTMAP cannot find a $SAVSYS$ entry in the directory, or if one or more MDISK statements in $SAVSYS$ define the DASD type as FB-512, SNTMAP prompts you to enter the DASD types. In this case, SNTMAP does not check for extent exceeded conditions in $SAVSYS$.

$DASD$ CONSTS: The $DASD$ CONSTS file, which is supplied with VM/SP, contains information about DASD that SNTMAP needs for calculations and conversions. SNTMAP cannot do any processing without this file.

Output Files

When these files are created, any old files with the same file IDs are erased.

DASD SNTMAP A: This file contains a list of the physical saved segment names and DASD allocations for each pack that has physical saved segment resources defined. The list is arranged in order of occurrence on a given pack. Any gaps are noted within the list and summarized in a table at the end of the file.

MEMORY SNTMAP A: This file is a list of physical saved segment names and allocations arranged in order by storage location. Overlapping saved segments are noted within the list. Note that defined unshared segments that extend beyond the end of the specified SYSHRSG(S) and into another segment defined at an adjacent location are flagged.

If you have grouped NAMESYS macros into families, this file shows a separate storage map for each family. Only the shared segments defined by a NAMESYS macro are mapped. Unshared pages are listed at the end of the file.

Responses

After you invoke SNTMAP, the exec may ask you for more information.

1. If you did not supply a valid file name for the DMKSNT file, SNTMAP will ask for the file name with the following prompt:

```
Enter name of SNT definition file or press the ENTER key to quit.
```

2. SNTMAP will ask for the name of the CP directory file with this message:

```
Enter the name of your current CP directory file.
Default is VMUSERS DIRECT.
```

3. If one of the following is true:

   - SNTMAP cannot find a CP directory
   - There is no USER $SAVSYS$ entry in the CP directory
   - MDISK statements in the directory define the DASD type as FB-512.

   SNTMAP will prompt you to enter the DASD types. In this case, SNTMAP will not check for $SAVSYS$ extent exceeded errors.

4. When SNTMAP has finished running, it will give you the following message:

```
Results of mapping are in two CMS files:

    DASD related information is in DASD SNTMAP
    Memory related information is in MEMORY SNTMAP
```

Messages

DMSWSM006E	No read/write *filemode* file mode accessed
DMSWSM847R	Enter name of SNT definition file or press the ENTER key to quit
DMSWSM848E	Unable to proceed without $DASD$ CONSTS file information
DMSWSM849R	Enter *volid* DASD type or type QUIT to end SNTMAP processing
DMSWSM850W	*devtype* is not a valid DASD type
DMSWSM851W	Page number exceeds device limit of *limit* for *sysname*; SYSSTRT parameter = *parameter*
DMSWSM852E	SYSSTRT parameter for *sysname* is not compatible with *devtype* DASD type for *volid*; SYSSTRT parameter = *parameter*
DMSWSM853R	Enter the name of your current CP directory file (the default is VMUSERS DIRECT):
DMSWSM854W	Unable to find CP directory file *fn ft fm*; no $SAVSYS$ boundary checking will be done
DMSWSM855W	No $SAVSYS$ areas defined in the CP directory file; no $SAVSYS$ boundary checking will be done
DMSWSM884I	Results of mapping are in two CMS files: DASD SNTMAP contains DASD related information, and MEMORY SNTMAP contains memory related information.

For a complete explanation of each message, see the *VM/SP System Messages and Codes* book.

Security Considerations

Consider the following when planning to install saved systems and saved segments.

- The SAVESYS command (writes information into the saved segment) is a privileged CP command -- Class E. Privilege classes provide a means for restricting access to CP commands. Thus, giving Class E privilege to product installers gives them access to all Class E commands and the ability to save any saved segment defined in the system.
- If PROTECT=NO is specified on the NAMESYS macro, users of a saved segment may find their address space corrupted by others.

Preparing the System Name Table File (DMKSNT)

The system name table (DMKSNT) consists of entries that identify the name and location of the following:

- Systems saved by way of the SAVESYS command and IPLed by name.
- Saved segments that must consist of all reentrant code—their storage space cannot be altered.

  Language files for CMS that are in a saved segment will get saved with the LANGGEN command. (Other saved segments get saved by the SAVESYS command.)
- DASD areas for CP message repository language files. These files will also be saved with the LANGGEN command.

Note that a system programmer assembles DMKSNT (a pageable module).

Four macros generate entries for the system name table:

- The NAMESYS macro creates an entry in the system name table for a virtual machine operating system or saved segment.
- The NAMENCP macro creates an entry in the system name table 3704/3705 control program.
- The NAME3800 macro creates an entry in the system name table for a 3800 printer image library.
- The NAMELANG macro creates an entry in the system name table for a CP message repository.

The sample DMKSNT files each have the following entries: one each for saving copies of CMS, CMSVSAM, CMSAMS, CMSDOS, CMSBAM, CMSINST, GCS, HELP, CMSVMLIB, and CMSFILES. If you use all recommended labels and allocations and the starter system supplied DMKSYS, you can save these segments during the system generation procedure. For an explanation of this procedure, and for an illustration of the storage layout resulting from this suggested configuration, see information on installing saved segments in the *VM/SP Installation Guide.*

If you wish to change or add to the suggested system name table, code your own macro and create a DMKSNT file of your own. You can create one entry for each type of saved segment.

If you create your own version of the system name table, your file must have a CSECT and END statement. You must group like macros together in the DMKSNT CSECT; that is, group all NAMESYS macros together, group all NAMENCP macros together, group all NAME3800 macros together, and group all NAMELANG macros together. If you group them incorrectly (for example, a NAMENCP macro is placed between two NAMESYS macros,) the system generates an MNOTE indicating that the macros are out of sequence.

Note: You should use the DMKSNT module as a pageable module. If you make this module resident and larger than one page, you should punch an SPB (Set Page Boundary) LOADER CARD at the beginning of the DMKSNT to ensure alignment at a page boundary.

```
DMKSNT   CSECT
         NAMESYS  macros (one for each virtual machine
                  operating system or segment you wish
                  to save)
         NAMENCP  macros (one for each 3704/3705 control
                  program you create)
         NAME3800 macros (one for each 3800 printer
                  image library you create)
         NAMELANG macros (one for each language's CP
                  message repository, other than the
                  system national language
         END
```

NAMESYS Macro

The NAMESYS macro describes the name and location of the saved system or physical saved segment. Shared segments may be specified, but they must consist of reenterable code, with no alteration of its storage space permitted.

The system programmer codes this macro. When you make additions, changes, or deletions to the system name table, you must reassemble the DMKSNT file and rebuild the system.

Table 27 contains sizing information for saved systems or physical segment names available under CMS and CP. The table also gives detailed segment information for customizing a DMKSNT file. When coding the NAMESYS macro, refer to this table for information about the following:

- Page number (SYSPGNM)
- Page count (SYSPGCT)
- Shared segment number (SYSHRSG).

Table 27. Planning for SNT Input

Saved System or Physical Segment Name	Page Number (SYSPGNM)	Page Count (SYSPGCT)	Shared Segment Number (SYSHRSG)
CMS	0-13, 16-34 3584-4095	545	224-255
CMSINST	3392-3455	64	212-215
HELP	3312-3391	80	207-211
CMSDOS	3296-3311	16	206
CMSFILES	2496-2751	256	156-171
CMSBAM	3072-3119	48	192-194
CMSVSAM	2960-3071	112	185-190
CMSAMS	2816-2959	144	176-181
CMSVMLIB	2752-2815	64	172-175
GCS	0-6, 1024-1279	263	064-079

Format

```
label | NAMESYS | SYSSIZE=nnnnnK, SYSNAME=name, SYSVOL= volid,
      |         |
      |         | VSYSADR=[cuu   ], [VSYSRES=volid,] [SYSCYL=nnnn,      ]
      |         |         [IGNORE]                 [SYSBLOK=nnnnnn,   ]
      |         |
      |         | SYSSTRT={(cccc,p),}        SYSPGCT=pppp,
      |         |         {pppppp,  }
      |         |
      |         | SYSPGNM=(nn,nn,nn-nn,...),
      |         |
      |         | SYSHRSG= (s,s,s-s,...),
      |         |
      |         | PROTECT={OFF}
      |         |         {ON }
      |         |
      |         | VMGROUP={YES}
      |         |         {NO }
      |         |
      |         | PARMRGS= (m,n)
      |         |
      |         | [USERID=userid,] [RCVRID=rcvrid,] [SAVESEQ={10      }]
      |         |                                   [        {priority}]
```

(Defaults underlined: ON, NO, 10.)

Parameters

label
: is any desired user label.

SYSSIZE=
: where *nnnnnk* represents the least amount of storage you must have available to IPL the saved system. You must specify *K*. Although you must code this operand for saved segments, it is not used for them.

SYSNAME=
: is the name (one-to-eight alphanumeric characters) the SAVESYS and/or IPL commands use to identify the system or segment. The name you select must not be one that could be interpreted as a hexadecimal device address (for example, A or E).

SYSVOL=
: is the volume identifier (one-to-six alphanumeric characters) of the DASD volume you want to receive the saved system or segment. This must be a CP-owned volume.

VSYSADR=*cuu*
: is the virtual address of the minidisk that is the system residence volume of the system you want to save. This operand defaults to IGNORE if you specify USERID=. The system flags other values. You must code "VSYSADR=IGNORE" when you are defining a saved segment. It may also be used when defining a saved system to improve performance.

Note: If you are likely to have many CMS users active at one time, you should distribute CMS activity over two volumes by (1) setting up a second CMS system residence volume and dividing the users between the two CMS system residence volumes or (2) putting your licensed programs on one spindle and CMS nonresident commands on another spindle.

VSYSADR = IGNORE
indicates that the NAMESYS macro describes a system or segment that does not require a virtual system residence volume. Code VSYSADR = IGNORE when you are defining a saved segment.

VSYSRES =
is the real volume identifier (one-to-six alphanumeric characters) of the DASD volume that contains the minidisk that is the system residence volume of the system you want to save. The system ignores this operand if you code VSYSADR = IGNORE, but you must specify it as null (VSYSRES = ,). This operand is flagged and ignored if you specify USERID = .

SYSCYL =
SYSBLOK =
is the real starting location of the virtual disk that is the system residence volume for the system you want to save (as the VSYSRES and VSYSADR specifies.)

The SYSCYL or SYSBLOK operand is ignored for VSYSADR = IGNORE provided you specify SYSCYL = or SYSBLOK = as null. If you specify USERID =, the system flags and ignores this operand.

- For count-key-data DASD, specify the SYSCYL = parameter.
- For fixed-block DASD, specify the SYSBLOK = parameter.

SYSSTRT =
is the starting location on SYSVOL where you want to save this named system. SAVESYS and IPL processing use this location to generate DASD addresses for I/O operations.

- For count-key-data SYSVOL devices, specify *(cccc,p)*. The number cc is the starting cylinder address and p is the starting page address. You must express both as decimals and they must be equal to or greater than one.
- For fixed-block SYSVOL devices, specify *pppppp*. The number *pppppp* is the starting page address as a decimal number. It must be greater than one.

The number of pages written to this area is the total number you specify on the SYSPGCT operand, plus up to three information pages. The number of information pages CP needs depends on the number of virtual machine pages you save. Table 27 on page 379 shows this relationship. Also, see "Relationship of Page Numbers, Segment Numbers, and Hexadecimal Addresses" on page 383 for help converting segment and page numbers to hexadecimal addresses.

SYSPGCT =
is the exact number of virtual machine pages you want to save as the SYSPGNM operand indicates. (Information pages CP requires are not counted.) The SYSPGCT operand is optional and the value cannot exceed 4096. If you do not specify the SYSPGCT operand, the NAMESYS macro calculates the number of pages to save.

SYSPGNM =
are numbers of the pages to save. You may specify pages singly or in groups. For example, if you want to save pages 0, 4, and 10 through 13, use the format: SYSPGNM = (0,4,10-13). The total must be equal to the SYSPGCT specification.

SYSHRSG =

are the segment numbers to be shared. The segment number (a decimal) must be zero or greater. For example, you can specify the first number as 0, the next as a larger decimal number, and so on. Pages in these segments (set at IPL time) can be used by loading this name.

You may specify the segment numbers singly or in groups. For example, if you want to share segment numbers 0, 2, 4, and 64 through 79, use the format: SYSHRSG = (0,2,4,64-79).

Shared segments must be reenterable and the maximum number must not exceed 78. Note, however, that using segment numbers with three or more digits reduces the maximum number of shared segments that you can define. For example, if all segments are three-digit numbers and you separate the segments by commas, you can define less than 64 shared segments. If you specify USERID =, then the system flags and ignores this operand.

PROTECT =

indicates that VM/SP is to run either with protected (ON) or unprotected (OFF) shared segments for the named system. ON is the default. If a named system is specified as unprotected, any changes made to shared pages in the named system will not be detected by the VM/SP control program; the change will be seen by all users of the shared page.

Virtual machines in a Virtual Machine Group always run with PROTECT = OFF. If you specify PROTECT = ON with the VMGROUP = YES specification, an MNOTE is generated indicating that PROTECT = OFF is assumed. If a shared system is specified as protected, segment zero must not be shared. Page zero in this segment contains areas that may change (that is, PSWs) and unpredictable results can occur if it is shared in protect mode.

VMGROUP =

determines if the saved system being defined is to be treated as a Virtual Machine Group. Specifying VMGROUP = YES indicates that all virtual machines that access the shared segment will become members of the Virtual Machine Group. With VMGROUP = YES specified, virtual machines accessing the segment are restricted to using the IPL command. Users cannot access the saved system using the FINDSYS or LOADSYS functions of DIAGNOSE code X'64'. The VMGROUP = YES specification also forces PROTECT = OFF.

The VMGROUP = NO specification indicates that the shared segment not be treated as a Virtual Machine Group. This is the default setting.

PARMRGS =

specifies the registers of the virtual machine that are to be filled with binary zeros before moving in the IPL parameters (where m and n are decimal numbers from 0 to 15 and m < = n) Any parameters which do not fit in the specified registers will be ignored. If this parameter is not specified, IPL parameters will be moved into the virtual machine's general purpose registers. If only one register is to be used for IPL parameters, ",n" may be omitted from the PARMRGS invocation.

Note that if you want to use the NOSPROF, INSTSEG, or SAVESYS parameters of the IPL command, you must have

```
PARMRGS=(0,15)
```

in the NAMESYS macro for the CMS named system.

USERID =

is the user ID of the virtual machine that is saved in the designated area. (This user can IPL from that area.) Any value specified for USERID indicates that this is a VMSAVE target area specification.

Note: More than one target area may be specified for a single user ID by including more than one NAMESYS macro with the same USERID = parameter.

RCVRID =

is the user ID of the virtual machine authorized to access a system save area. This is an optional parameter. It defaults to the value you specify for USERID. If you do not specify USERID = and you do specify this RCVRID parameter, the system issues an MNOTE and flags and ignores the RCVRID parameter.

SAVESEQ =

specifies the order in which multiple virtual machines will be saved. Values can be from 0 to 255. The virtual machine with the lowest number is saved first. If two virtual machines have the same value, the one that enabled the VMSAVE option first is dumped first. If you do not specify USERID = and you do specify this SAVESEQ parameter, the system issues an MNOTE and flags and ignores the SAVESEQ parameter. If the SAVESEQ priority value is not supplied, the default value is 10.

Available Space for Each DASD Type: The number of 4K pages available per DASD cylinder is:

Pages/Cylinder	DASD Type
24	3340-35, 3340-70, 2305
57	3330, 3333
120	3350 (in native mode)
96	3375
150	3380

Relationship of Page Numbers, Segment Numbers, and Hexadecimal Addresses: Because the NAMESYS macro requires you to specify page and segment numbers, and the DOSGEN, SAMGEN, and VSAMGEN procedures require you to enter hexadecimal addresses, you may find the following reference information useful.

```
 1K  =  1024  = X'400'
 4K  =  4096  = X'1000'   =  1 page
64K  = 65536  = X'10000'  = 16 pages = 1 segment
```

To convert a page number to a segment number, divide the page number by 16. Because one segment is 10000 in hexadecimal, then 20000 is segment 2, 100000 is segment 16, 180000 is segment 24, and so on.

Appendix C, "Worksheet to Aid in Coding the NAMESYS Macro" on page 431 contains a worksheet to help you convert segment and page numbers to hexadecimal addresses.

More Information: Information on the following subjects can be found in the *VM/SP Application Development Guide for CMS* book.

- Saving the CMS system
- Saved system restrictions for CMS
- Saving OS.

NAMENCP Macro

The NAMENCP macro defines 3704/3705 program entries in the system name table.

This section applies only to EPs as defined, created, and loaded through VM/SP. If you have a 3725 communication controller or a 3705 that will be loaded by way of ACF/SSP, see the following manuals:

ACF/NCP-SSP, V3 Installation and Resource Definition Guide (explains how to generate and load the NCP.)

EP/3725 Installation and Resource Definition Guide and Reference

EP/3705 Generation and Utilities Guide and Reference

ACF/NCP V4, ACF/SSP V3 Diagnosis Guide (explains how to dump the contents of the 37X5 and how to run ACF/TAP and CRP).

The NAMENCP macro describes the location of the 37xx control program. You must create an entry in the system name table (DMKSNT) for each separate 3704/3705 control program that you generate. You must specify shared segments, but the segments must consist of reentrant code. If you can foresee generating many versions of the 3704/3705 control program, define extra entries in the system name table when you generate VM/SP. In this way, you do not have to regenerate the VM/SP system just to update the system name table.

The system programmer assembles this macro. When you make additions, changes, or deletions to the system name table, you must reassemble the DMKSNT file and rebuild the system.

Format

label	NAMENCP	**CPSIZE = nnnK,** **CPNAME =** *ncpname,* **CPTYPE = {EP}** **SYSPGCT =** *pp,* **SYSVOL =** *volid,* **SYSSTRT =** { *(cccc,p)* / *pppppp* }

Parameters

label
is any desired user label.

CPSIZE =
is the storage size of your 3704/3705. You can specify a maximum of 256K.

CPNAME =
is the name (one-to-eight alphanumeric characters) of the 3704/3705 control program image. You can use this name in the SAVENCP and NETWORK LOAD commands.

CPTYPE =
is the 3704/3705 control program type.

SYSPGCT =
is the total number of pages (*pp*) you want to save. This value is equal to the number of pages the CPSIZE operand implies, plus 1 additional page for CP information.

SYSVOL=
is the volume identifier (volid) of the DASD volume that receives the CP image. That volume must be CP-owned.

SYSSTRT=
is the starting location on SYSVOL where you want to save this named system. SAVESYS and IPL processing use this location to generate DASD addresses for I/O operations.

- For count-key-data SYSVOL devices, specify *(cccc,p)*. The number *ccc* is the starting cylinder address and *p* is the starting page address. You must express both as decimal numbers and they must be equal to or greater than one.
- For fixed-block SYSVOL devices, specify *pppppp*. The number *pppppp* is the starting page address expressed as a decimal number. It must be greater than one.

NAME3800 Macro

The NAME3800 macro describes the name and location of the named system that contains the 3800 character arrangement tables, graphic modifications, FCBs, and copy modifications for the 3800 printers. You can specify more than one named system. The 3800's RDEVBLOK contains a pointer to the named system currently in use for that particular 3800 printer.

Format

label	**NAME3800**	**CPNAME** = *imagelib,* **SYSPGCT** = *pppp,* **SYSVOL** = *volid,* **SYSSTRT** = $\left\{\begin{matrix} (cccc,p) \\ pppppp \end{matrix}\right\}$

Parameters

label
: is any desired user label.

CPNAME =
: is the name (one-to-eight alphanumeric characters) of the 3800 image library. The IMAGELIB or IMAGEMOD command uses this name.

SYSPGCT =
: is the total number of pages *(pppp)* you specify to save for the image library. This value is a decimal number up to four digits. To determine the number of pages you want to save, use the following steps:

1. The image library consists of one or more members. These members are core images of the TEXT decks created by GENIMAGE. The hexadecimal size of each TEXT file is contained in columns 31 and 32 of the ESD card.
2. Sum the sizes and add 16 bytes to the total.
3. Divide the total by 4096 bytes to achieve the page count *(pppp)*. Be sure to round up to the next whole page. This value is equal to the number of pages the CPSIZE operand implies, plus 1 additional page for CP information.

SYSVOL =
: is the volume identifier (volid) of the DASD volume you specify to receive the 3800 image library. The volume must be a CP-owned volume.

SYSSTRT =
: is the starting location on SYSVOL where you want to save this image library.

- For count-key-data SYSVOL devices, specify *(cccc,p)*. The number ccc is the starting cylinder address and p is the starting page address. You must express both as decimal numbers and they must be one or greater.
- For fixed-block SYSVOL devices, specify *pppppp*. The number *pppppp* is the starting page address. You must specify it as a decimal number. It must be two or greater.

NAMELANG Macro

The NAMELANG macro reserves DASD space for a compiled CP message repository. This macro describes the name and location of the DASD area that will contain the compiled CP message repository for a specific language.

You also have to code a NAMESYS macro to create a DCSS for the CMS message repository and other CMS language files. Be sure that the *langid* you specify on NAMESYS is the same *langid* you specify on NAMELANG, or you will get unpredictable results.

You do not need to code a NAMELANG or NAMESYS macro entry in DMKSNT for the system national language—only for any additional languages you want to have on your system. The system national language information is in the nucleus; CP will always use that nucleus information rather than looking on the DASD specified in DMKSNT.

Ensure that all NAMELANG macros are grouped together in the DMKSNT module.

Format

label	**NAMELANG**	**LANGID** = *langid*, **NLSPGCT** = *pagecount*, **NLSVOL** = *volid*, **NLSSTRT** = { *(cccc,p)* | *pppppp* }

Parameters

label
is any desired user label. (This is optional.)

LANGID =
identifies the language of the compiled CP message repository that you want to save. Use up to five characters to specify the *langid.*

To set this language when the virtual machine logs on, specify this *langid* in the user's directory entry.

Note: The language related information that CMS and CMS applications use are in a DCSS. You define this DCSS using the naming conventions for the NAMESYS macro. (The SYSNAME operand in NAMESYS for this DCSS has the format NLSlangid.) The system programmer must ensure that the *langid* on NAMELANG matches the *langid* specified on the NAMESYS macro; otherwise, users will have problems setting languages for CP and CMS.

NLSPGCT =
specifies the total number of pages (*pagecount*) to reserve on DASD for the compiled CP message repository. Specify this page count as a decimal number greater than one. To determine the number of pages you want to save, check the listing produced by the message compiler; the minimum number of pages required by a CP message repository is printed on the listing. The value you specify for NLSPGCT should be equal to or greater than this minimum number, plus one additional page for CP.

NLSVOL=
is the volume serial number of the DASD volume designated to contain the compiled CP message repository. Specify the volid using one-to-six alphanumeric characters. The system programmer must ensure that this volume is CP-owned and CP formatted.

NLSSTRT=
is the starting location on the NLSVOL where the CP message repository will be saved.

- For count-key-data NLSVOL devices, specify (*cccc,p*). The number *cccc* is the starting cylinder address and *p* is the starting page address. You must express the cylinder and page values as decimal numbers, and they each must be equal to or greater than one.

- For fixed-block NLSVOL devices, specify *pppppp*. The number *pppppp* is the starting page address expressed as a decimal number. It must be greater than one.

For each error, NAMELANG sets a severity code of 12. The processing of NAMELANG continues to detect any other errors that may have been made in coding this macro entry.

Example of Coding the System Name Table File (DMKSNT)

Figure 43 shows a sample system name table file (DMKSNT).

```
SNT      TITLE 'DMKSNT       VM/SP REL 6       3380 SAMPLE'
DMKSNT   CSECT
*
**********************************************************************
*     HEX LOAD ADDRESS FOR SEGMENT 224 = E00000                      *
*     THE SPACE FOR CMS IS ALLOCATED ON VMSRES, AS FOLLOWS:          *
*     ( THE ALLOCATIONS ARE BASED ON 150 PAGES/3380 CYLINDER )       *
*         CYL 6, PAGE 1   TO CYL 9, PAGE 96    (546 PAGES)           *
*         545 PAGES FOR CMS, 1 FOR CP INFORMATION.                   *
**********************************************************************
CMS      NAMESYS    SYSNAME=CMS,                                      X
               SYSVOL=VMSRES,                                         X
               SYSSTRT=(006,1),                                       X
               SYSPGNM=(0-13,16-34,3584-4095),                        X
               SYSPGCT=545,                                           X
               SYSHRSG=(224-255),                                     X
               SYSSIZE=256K,                                          X
               VSYSADR=190,                                           X
               SYSCYL=533,                                            X
               PARMRGS=(0,15),                                        X
               VSYSRES=VMSRES
         EJECT
**********************************************************************
*     THE SPACE FOR NLSENGLI IS ALLOCATED ON SNTPAK, AS FOLLOWS:     *
*     (THE ALLOCATIONS ARE BASED ON 150 PAGES/3380 CYLINDER)         *
*            CYL 307, PAGE 001 TO CYL 307, PAGE 64,                  *
*            65 PAGES = 64 PAGES FOR LANGUAGE, 1 for CP INFORMATION  *
**********************************************************************
         SPACE
NLSENGLI NAMESYS    SYSNAME=NLSENGLI,                                 X
               SYSVOL=SNTPAK,                                         X
               SYSSTRT=(307,01),                                      X
               SYSPGNM=(3696-3759),                                   X
               SYSPGCT=64,                                            X
               SYSHRSG=(231-234),                                     X
               SYSSIZE=256K,                                          X
               SYSCYL=,                                               X
               VSYSRES=,
               VSYSADR=IGNORE                                         X
         EJECT
```

Figure 43 (Part 1 of 2). System Name Table File Sample (DMKSNT)

```
***********************************************************************
*     THE SPACE FOR ENGLI IS ALLOCATED ON SNTPAK, AS FOLLOWS:         *
*     (THE ALLOCATIONS ARE BASED ON 150 PAGES/3380 CYLINDER)          *
*            CYL 308, PAGE 001 TO CYL 307, PAGE 64,                   *
*            16 PAGES = 15 PAGES FOR LANGUAGE, 1 for CP INFORMATION   *
***********************************************************************
           SPACE
ENGLISH    NAMELANG  LANGID=ENGLI,                                      X
                 NLSVOL=SNTPAK                                          X
                 NLSSTRT=(308,01)                                       X
                 NLSPGCT=16
           END
***********************************************************************
*     THE SPACE FOR VM3800 IS ALLOCATED ON SNTMAP, AS FOLLOWS:        *
*     (THE ALLOCATIONS ARE BASED ON 150 PAGES/3380 CYLINDER.)         *
*         CYL 090, PAGE 001 TO CYL 008, PAGE 003                      *
*         751 PAGES = 750 PAGES FOR 3800 CODE, 1 FOR CP INFORMATION   *
***********************************************************************
         SPACE
VM3800 NAME3800 CPNAME=VM3800,                                          X
             SYSVOL=SNTMAP                                              X
             SYSSTRT=(090,01),                                          X
             SYSPGCT=750
         EJECT
***********************************************************************
*     THE SPACE FOR VMEP01 IS ALLOCATED ON VMSRES AS FOLLOWS:         *
*         CYL 12, PAGE 101 TO CYL 12, PAGE 116   (16 PAGES)           *
***********************************************************************
         SPACE
VMEP01 NAMENCP CPNAME=VMEP01,                                           X
               CPSIZE=48K,                                              X
               CPTYPE=EP,                                               X
               SYSSTRT=(012,101),                                       X
               SYSPGCT=16,                                              X
               SYSVOL=VMSRES
         EJECT
```

Figure 43 (Part 2 of 2). System Name Table File Sample (DMKSNT)

Chapter 11. Additional System Definition Files

Contents of Chapter 11

Forms Control Buffer Load (DMKFCB)

The DMKFCB module is supplied with the product tape. This module defines a 3211, 3203-4, 3203-5, 3262-1, 3262-5, 3262-11, 3289 Model 4, 4245, or 4248-1 printer forms control buffer image. There are two names provided for an FCB image.

FCB1 controls printing at 6 lines per inch, 66 lines per page and has the following specifications:

Line Represented	Channel Skip Specification
1	1
3	2
5	3
7	4
9	5
11	6
13	7
15	8
19	10
21	11
23	12
64	9

FCB8 controls printing at 8 lines per inch, 68 lines per page and has the following specifications:

Line Represented	Channel Skip Specification
1	1
4	2
8	3
12	4
16	5
20	6
24	7
28	8
32	10
36	11
63	12
66	9

If you wish to alter the supplied buffer load, see the *VM/SP Administration* book for directions.

Universal Character Set and Font Offset Buffer

The DMKUCS, DMKUCB, DMKUCC, DMKPIA, and DMKPIB modules are supplied with the product tape. These modules correspond to the following printer types:

Printer Type	Module Name
1403	DMKUCS
3211	DMKUCB
3203	DMKUCC
3289	DMKPIA
3262	DMKPIB

If you wish to change the supplied buffer load for a particular device, see the corresponding module's prolog.

DEFNUC Macro

The DEFNUC macro defines responses to the system for prompts issued during the CMS nucleus generation procedure. The expansion of the macro produces the nonexecutable DMSNGP (CMS Nucleus Generation Profile) text file. Figure 44 on page 399 shows an example of how to code a DMSNGP text file. DMSINI reads the responses for prompts from the text file. The prompts will be issued if:

- The text deck is not present
- An operand is omitted in the macro
- A question mark is entered as the response

  -OR-

- DMSNGP contains an invalid response.

Format

```
[label]  DEFNUC  [,SYSDISK=[cuu]]  [,YDISK=[cuu]]  [,HELP=[cuu]]
                           [?  ]             [?  ]           [?  ]

                 [,LANGID=[name]]  [,DBCS=[Yes|1]]  [,LANGLEV=[level]]
                          [?   ]          [No |0]             [?    ]
                                          [?     ]

                 [,BUFFSIZ=[size]] [,USEINST=[Yes|1]] [,INSTSEG=[name]]
                           [?   ]            [No |0]            [?   ]
                                             [?     ]

                 [,SAVESYS=[Yes|1]] [,SYSNAME=[systemname]] [,REWRITE=[Yes|1]]
                           [No |0]            [?         ]            [No |0]
                           [?     ]                                   [?     ]

                 [,IPLADDR=[cuu]]  [,CYLADDR=[cuu]]  [,IPLCYL0=[Yes|1]]
                           [?  ]             [?  ]             [No |0]
                                                               [?     ]

                 [,VERSION=['string']]  [,INSTID=['string']
                           [?       ]            [?       ]
```

Parameters

label
  is any desired user label

SYSDISK =
  defines the system disk address. The response may be three characters or less. A ? indicates that the prompt should be issued. If you specify SYSDISK = without a value, a null response is issued for the prompt and the default value of 190 is used.

YDISK =
  defines the Y-disk address. The response may be three characters or less. A ? indicates that the prompt should be issued. If you specify YDISK = without a value, a null response is issued for the prompt and the default value of 19E is used.

HELP=
defines the HELP disk address. The response may be three characters or less. A ? indicates that the prompt should be issued. If you specify HELP= without a value, a null response is issued for the prompt and the default value of 19D is used.

LANGID=
specifies the language identifier of the default language. The response may be five characters or less. A ? indicates that the prompt should be issued. If you specify LANGID= without a value, a null response is issued for the prompt and the default value of AMENG (American English) is used.

DBCS=
specifies whether or not the system national language is a Double-Byte Character Set (DBCS) language. Enter YES, Y, or 1 to indicate that the default language is a DBCS language. Enter NO, N, or 0 if it is not. A ? indicates that the prompt should be issued. If you specify DBCS= without a value, a null response is issued for the prompt and the default value of NO is used.

LANGLEV=
specifies the level of the saved segment for the default language. The response may be one alphanumeric character, using A to Z or 0 to 9. A ? indicates that the prompt should be issued. If you specify LANGLEV= without a value, a null response is issued for the prompt and the default value of S is used.

BUFFSIZ=
specifies the size of the file system Read/Write cache buffers in kilobytes. The response can be any number from 1 to 28. If you specify BUFFSIZ= without a value, a null response and a default size of 12 are issued. A ? indicates that the prompt should be issued.

USEINST=
indicates whether or not the CMS installation saved segment will be created. Enter YES, Y, or 1 if you plan to create the installation saved segment. Enter NO, N, or 0 if it will not be created. If you specify USEINST= without a value, a null response is issued for the prompt and the default value of YES is used. A ? indicates that the prompt should be issued.

INSTSEG=
defines the name of the CMS installation saved segment. To name the saved segment, enter a valid name using one-to-eight alphanumeric characters, A to Z or 0 to 9. If you specify INSTSEG= without a value, a null response is issued for the prompt and the default value of CMSINST is used. A ? indicates that the prompt should be issued.

SAVESYS=
indicates whether or not the CMS nucleus is to be saved as a named, saved system. Enter YES, Y, or 1 if you plan to save the system. Enter NO, N, or 0 if it will not be saved. If you specify SAVESYS= without a value, a null response is issued for the prompt and the default value of YES is used. A ? indicates that the prompt should be issued.

SYSNAME=
specifies the name of the saved system. To specify the system name, enter a valid name using one-to-eight alphanumeric characters, A to Z or 0 to 9. If you specify SYSNAME= without a value, a null response is issued and the default system name of CMS is used. A ? indicates that the prompt should be issued.

REWRITE=
indicates whether or not the CMS nucleus should be written to disk. Enter YES, Y, or 1 if you want to write the nucleus to disk. Enter NO, N, or 0 if you do not. If you specify REWRITE= without a value, an MNOTE will appear in the text deck and the prompt will be issued. A ? indicates that the prompt should be issued.

IPLADDR=
defines the disk address on which the nucleus will be written. The response may be three characters or less. If you specify IPLADDR= without a value, an MNOTE will appear in the text deck and the prompt will be issued. A ? indicates that the prompt should be issued.

CYLADDR=
specifies the cylinder/block address at which the nucleus will be written on the IPLADDR disk. The response may be five characters or less. If you specify CYLADDR= without a value, an MNOTE will appear in the text deck and the prompt will be issued. A ? indicates that the prompt should be issued.

The nucleus resides on the last cylinder(s) or block(s) of the 190 minidisk. The following table shows the relative position from the beginning of the 190 disk, according to DASD type:

Device Type	190 Allocation	Nucleus Size	Cylinder Address
3330	173 cyl	18 cyl	155 cyl
3340	429 cyl	45 cyl	384 cyl
3350	80 cyl	10 cyl	70 cyl
3375	120 cyl	12 cyl	108 cyl
3380	78 cyl	8 cyl	70 cyl
FB-512	72,000 blk	8,256 blk	63,744 blk

No response is defined in the sample file. Enter the value from the table that is correct for your system.

IPLCYL0=
indicates whether or not the cylinder/block zero should also be IPLed. Enter YES, Y, or 1 if you want to IPL the cylinder/block zero. Enter NO, N, or 0 if you do not. If you specify IPLCYL0= without a value, an MNOTE will appear in the text deck and the prompt will be issued. A ? indicates that the prompt should be issued.

VERSION=
defines the version identification that is displayed each time a user IPLs this CMS system. To specify the version heading, enter a valid heading using 1-to-32 alphanumeric characters, A to Z or 0 to 9. If you specify VERSION= without a value, a version heading is constructed at the time of execution, such as: `VM/SP REL n mm/dd/yy hh:mm:ss`. A ? indicates that the prompt should be issued.

INSTID=
specifies the heading that appears at the beginning of each output file. To specify the installation heading, enter a valid heading using 1-to-64 alphanumeric characters, A to Z or 0 to 9. If you specify INSTID= without a value, an

installation heading is constructed at the time of execution, such as: VM/SP CONVERSATIONAL MONITOR SYSTEM. A ? indicates that the prompt should be issued.

```
NGP      TITLE 'DMSNGP     (CMS)      VM/SP - VIRTUAL MACHINE SYSTEM
                                              PRODUCT - 5664-167'
DMSNGP   CSECT
         DEFNUC SYSDISK=190,        * S-disk address                  *
               YDISK=19E,           * Y-disk address                  *
               HELP=19D,            * Help disk address               *
               LANGID=AMENG,        * Default is American English     *
               DBCS=NO,             * Default is not a DBCS lang      *
               LANGLEV=S,           * DCSS ID for multiple DCSS       *
               BUFFSIZ=12,          * R/W cache buffer size           *
               USEINST=YES,         * Using EXEC/XEDIT in DCSS        *
               INSTSEG=CMSINST,     * Name of above DCSS to save      *
               SAVESYS=YES,         * Using CMS in DCSS yes or no     *
               SYSNAME=CMS,         * Name of above DCSS to save      *
               REWRITE=YES,         * Write nucleus yes or no         *
               IPLADDR=190,         * Address of where to write       *
               CYLADDR=?,           * Cyl/Blk of where to write       *
               IPLCYL0=YES,         * write ipl text on cyl 0         *
               VERSION=,            * VM/SP REL n mm/dd/yy hh mm ss   *
               INSTID=              * VM/SP CONVERSATIONAL MONITOR SY
         END
```

Figure 44. Sample Coding for the CMS Nucleus Generation Profile (DMSNGP)

Appendix A. VM/SP Configuration Aid

The Configuration Aid shown in the material that follows consists of two tables:

- Matching Hardware to Control Units
- Matching Hardware to Shared or Nonshared Virtual Control Units.

The first lists hardware devices and control units (grouped by use) needed for VM/SP system generation. The second matches the shared or nonshared device properties with each device that can be created as a stand-alone virtual device.

Matching Hardware to Control Units

The following table shows:

- Devices that can be attached to each control unit
- Operands named for each device in RDEVICE macro
- Maximum number of devices specified in the FEATURE=operand of the RCTLUNIT macro
- Shared subchannel operation (if any).

The control units have been placed in subgroups according to the ways that they can be configured. For example, the chart of tape devices indicates that a 2401 or 2402 can be attached to a 2803 or 2804 control unit.

Type of Device	RCTLUNIT		Shared Sub-channel	RDEVICE	
	CUTYPE=	Maximum FEATURE=		DEVTYPE=	Other Operands
System Consoles*	CONS	—	—	1052	—
	CONS	—	—	3210	—
	CONS	—	—	3215	—
	CONS	—	—	2150	—
	CONS	—	—	3066	—
	CONS	—	—	3138	—
	CONS	—	—	3148	—
	CONS	—	—	3158	—
	CONS	—	—	3036	—
	CONS	—	—	3278	MODEL=2A
	CONS	—	—	3279	MODEL=2A, 2C
Trans-mission Control Units	2701	—	—	2701	ADAPTER=BSCA, IBM1, or TELE2
	2702	32-DEVICE	—	2702	ADAPTER=BSCA, IBM1, or TELE2 SETADDR=0, 1, 2, or 3
	2703	176-DEVICE	—	2703	ADAPTER=BSCA, IBM1, or TELE2
	3704 3705 3725 ICA	16-DEVICE 256-DEVICE 256-DEVICE 16-DEVICE	— no yes —	3704 3705 3725 ICA	ADAPTER=BSCA, IBM1, TELE2, TYPE1, TYPE2, TYPE3,TYPE4, or TYPE5 (for 3725) MODEL=A1 through L8 (1 or 2 for 3725) SETADDR=0, 1, 2, or 3 CPTYPE=EP CPNAME=ncpname (omitted for 3725) BASEADD=cuu ADAPTER=BSCA, IBM1, TELE2, or SDLC
	2955	—	—	2955	—

*If other devices are connected in the same range of addresses with a system console, you should use the CUTYPE that applies to the device other than the system console. For example, CUTYPE=3811 should be specified for RCTLUNIT 010 if the devices in the address range 010-017 are 3148 (console) and 3211 (printer).

Many console device types have no direct analogs in the CP DEFINE command. To emulate a 1052, 3210, 3215, 2150, or 3278 Model 2A you should define a GRAF device type 3036, 3066, 3138, 3148, or 3158.

Type of Device	RCTLUNIT		Shared Sub-chan-nel	RDEVICE	
	CUTYPE=	Maximum FEATURE=		DEVTYPE=	Other Operands
SNA Con-trollers (Note 4)	3725	256-DEVICE	yes	3725	ADAPTER=TYPE5, MODEL=(1 or 2) CPTYPE=NCP
Display Devices (Local Attached)	2848	32-DEVICE	yes	2260 1052	—
	2845	—	yes	2265	—
	2840	16-DEVICE	yes	2250	—
	3272*	32-DEVICE	yes	3277 3284 3286 3288	FEATURE=OPRDR
	3274 (Note1)	32-DEVICE	yes	3262 3277 3278 (Note 2) 3279 (Note 3) 3284 3286 3287 3288 3289 3289E 4250 6262	 FEATURE=OPRDR MODEL=2, 3, 4, or 5 MODEL=2, or 3
	3276			3262 3278 (Note 2) 3279 (Note 3) 3287 3289	 MODEL=2, 3, 4, or 5
	DPA	—	— —	3230 3262 3278 (Note 2) 3279 (Note 3) 3284 3286 3287 3288 3289 4250 6262	
	HFCU	32-DEVICE	yes	HFGD	—
	(5088)		no	(5080)	—

* If a 3287 is attached to a 3272, specify the 3287 as a 3284 or 3286.
1. CUTYPE=3274 should be used for local attached non-SNA 3174s.
2. DEVTYPE=3278 should be used for 3178, 3180, 3191, and 3192 monochrome displays. MODEL=2 should be used for 3178 and 3191 displays
3. DEVTYPE=3279 should be used for 3179, and 3192 color displays.
4. Including 3705, 3720, 3725, 3274-SNA, 3174-SNA, and 3174-Token Ring.

Type of Device	RCTLUNIT		Shared Sub-chan-nel	RDEVICE	
	CUTYPE=	Maximum FEATURE=		DEVTYPE=	Other Operands
Remote 3270 Display Devices	2701	—	—	2701	ADDRESS=cuu (line address) ADAPTER=BSCA CLUSTER=label
	2703	—	—	2703	ADDRESS=cuu (line address) ADAPTER=BSCA or SDLC CLUSTER=label
	ICA 3704 3705 3725	— — — —	— — — —	ICA 3704 3705 3725	ADDRESS=cuu (line address ADAPTER=BSCA CLUSTER=label ADDRESS=cuu (line address ADAPTER=BSCA CPTYPE=EP BASEADD=cuu CLUSTER=label
Direct Access Storage Devices	2841	—	yes	2321 2303	
	IFA	16-DEVICE	—		
	3830 3830 3880 3880 3345 ISC	32-DEVICE 160-DEVICE 32-DEVICE 32-DEVICE 16-DEVICE 64-DEVICE	— — — — — —	3330 3330 3330 3333 3333	MODEL=1, 2, or 11 FEATURE=SYSVIRT, FEATURE=VIRTUAL MODEL=1, 2, or 11 MODEL=1 or 11 MODEL=1 or 11
	3830 3880 3345 ISC IFA 3830 3880 ISC IFA 3880 3880 3990 3880 3880	64-DEVICE 64-DEVICE 16-DEVICE 160-DEVICE 160-DEVICE 64-DEVICE 64-DEVICE 64-DEVICE 16-DEVICE 64-DEVICE 64-DEVICE 64-DEVICE 16-DEVICE 16-DEVICE 32-DEVICE 64-DEVICE	— — — — — — — — — — — — — — — — — — —	3340 3340 3350 3350 3375 3380 3380 3380 FB-512 9313 9332 9335 3310 3370	 FEATURE=FH FEATURE=FH
	2820	—	yes	2301	
	2835	16-DEVICE	—	2305	MODEL=1 or 2
	FTA	16-DEVICE	—	FB-512	
Tape Devices	2803 2804	16-DEVICE 16-DEVICE	yes	2401 2402	MODEL=1, 2, 3, 4, 5, 6, or 8 FEATURE=7-TRACK, DUALDENS MODEL=1, 2, 3, 4, 5 or, 6 FEATURE=7-TRACK, CONV, DUALDENS MODEL=5 or 7

Type of Device	RCTLUNIT CUTYPE=	RCTLUNIT Maximum FEATURE=	Shared Sub-channel	RDEVICE DEVTYPE=	RDEVICE Other Operands
Tape Devices (cont.)	3411 2403 2404	— 16-DEVICE	yes yes	3410 3411 2403 2404	MODEL=1, 2, or 3 FEATURE=7-TRACK, DUALDENS MODEL=1, 2, 3, 4, 5 or 6 FEATURE=7-TRACK, CONV, DUALDENS
	2415	—	yes	2415	MODEL=1, 2, 3, 4, 5 or 6 FEATURE=7-TRACK, CONV
	FTA	16-DEVICE	—	8809 9347	
	3411	—	yes	3410 3411	MODEL=1, 2, or 3 FEATURE=7-TRACK, DUALDENS
	3422	—	yes	3422	
	3430	—	yes	3430	FEATURE=DUALDENS
	3480	16-DEVICE (Note 3)	no	3480	
	3803 9347	16-DEVICE (Note 1) —	yes —	3420 9347	MODEL=3, 4, 5, 6, 7 or 8 FEATURE=7-TRACK, DUALDENS
Unit Record Output Devices	2821	—	—	1403 2540P	CLASS=(class ,class...) FEATURE=UNVCHSET CLASS=(class ,class...)
	1443	—	—	1443	CLASS=(class ,class...)
	3811	—	—	3211	CLASS=(class ,class...)
	3262	—	—	3262 (Note 2)	MODEL=5 CLASS=(class ,class...)
	2826	—	—	1018	
	2520	—	—	2520P	CLASS=(class ,class...)
	SVPC	—	—	3262	MODEL=1 or 11
	SVPC	—	—	3289	MODEL=4
	SVPC	—	—	3289E	MODEL=4
	3203	—	—	3203	MODEL=4 or 5 CLASS=(class ,class...)
	3525	—	—	3525	CLASS=(class ,class...)

1 FEATURE=16-DEVICE should be specified for 3803 when the communicator feature is used, allowing access to a second tape control unit and eight more tape drives.
2 The RCTLUNIT, when coded as 3262, is valid for DEVTYPE=3262, MODEL=5.
3 When two 3480 Control Units are attached via Dual Control Unit Communications Feature.

Type of Device	RCTLUNIT		Shared Sub-chan-nel	RDEVICE	
	CUTYPE=	Maximum FEATURE=		DEVTYPE=	Other Operands
Unit Record Output Devices (cont.)	3800	—	—	3800	MODEL=1,3, or 8 FEATURE=4WCGMS, IMAGE=imagelib, CHARS=ffff,FCB=lpi, DPMSIZE=n CLASS=(class ,class...)
	4245	—	—	4245	MODEL=1 CLASS=(class ,class...)
	4248	—	—	4248	MODEL=1 CLASS=(class ,class...)
	6262	—	—	6262	
Unit Record Input Devices	2821	—	—	2540R	
	2520	—	—	2520R	
	3505	—	—	3505	
	2495 2822	— —	— —	2495 2671	
	2826	—	—	1017	
	2501	—	—	2501	
Special Devices	CTCA	—	—	CTCA	
	3088	64-DEVICE	—	3088	
	7443	—	—	7443	

Matching Hardware to Shared or Nonshared Virtual Control Units

The virtual control unit (VCU) is an internal model of the control unit. Each VCU in either a SHARED or NONSHARED mode controls 16 adjacent device addresses. One of the two modes must be selected when the device is attached by way of the three hex-symbol address, cuu, where:

- Channel (c)
- Control unit (first u)
- Device (second u).

If the VCUNOSHR option is enabled for a virtual machine, then every VCU created for that machine operates in NONSHARED mode. There is no restriction on combining SHARED and NONSHARED devices in this case because all devices are simulated in NONSHARED mode for virtual I/O operations.

The default operation (VCUNOSHR not present) causes CP to simulate SHARED subchannel protocol for devices designed for SHARED subchannels. This mode of operation also enforces the restriction that SHARED and NONSHARED devices cannot coexist on the same VCU.

The following table matches the shared or nonshared device properties with each device that can be created as a stand-alone virtual device. The table shows the command or directory statement (COMMAND), the proper device specification (DEVICE SPEC), and the VCU mode (VCU MODE).

Devices attached or dedicated assume the properties of the real control unit to which the real device is attached. The real control unit is defined by the RCTLUNIT macro in DMKRIO.

Temporary Disk Space (TDISK)

Command	Device Spec	VCU MODE
DEFINE	T2305	Nonshared
DEFINE	T3330	Nonshared
DEFINE	T3340	Nonshared
DEFINE	T3350	Nonshared
DEFINE	T3375	Nonshared
DEFINE	T3380	Nonshared
DEFINE	T3310	Nonshared
DEFINE	T3370	Nonshared
DEFINE	T9313	Nonshared
DEFINE	T9332	Nonshared
DEFINE	T9335	Nonshared
DEFINE	TFB-512	Nonshared

Unit Record Device

Command	Device Spec	VCU MODE
DEFINE	READER	Nonshared
DEFINE	RDR	Nonshared
DEFINE	2540R	Nonshared
DEFINE	2501	Nonshared
DEFINE	3505	Nonshared
DEFINE	PUNCH	Nonshared
DEFINE	PCH	Nonshared
DEFINE	2540P	Nonshared
DEFINE	3525	Nonshared
DEFINE	PRINTER	Nonshared
DEFINE	PRT	Nonshared
DEFINE	PTR	Nonshared
DEFINE	1403	Nonshared
DEFINE	1443	Nonshared
DEFINE	3211	Nonshared
DEFINE	3203	Nonshared
DEFINE	3262	Nonshared
DEFINE	3289E	Nonshared
DEFINE	3800	Nonshared
DEFINE	4245	Nonshared
DEFINE	4248	Nonshared
SPOOL	2540 READER	Nonshared
SPOOL	2501	Nonshared
SPOOL	3505	Nonshared
SPOOL	2540 PUNCH	Nonshared
SPOOL	3525	Nonshared
SPOOL	1403	Nonshared
SPOOL	1443	Nonshared
SPOOL	3211	Nonshared
SPOOL	3203	Nonshared
SPOOL	3262	Nonshared
SPOOL	3289E	Nonshared
SPOOL	3800	Nonshared
SPOOL	4245	Nonshared
SPOOL	4248	Nonshared

Terminals and Displays

Command	Device Spec	VCU MODE
CONSOLE	1052	Nonshared
CONSOLE	3210	Nonshared
CONSOLE	3215	Nonshared
CONSOLE	3270	Nonshared
DEFINE	CONSOLE	Nonshared
DEFINE	GRAF	Shared
DEFINE	GRAF 3270	Shared
DEFINE	GRAF 3138	Nonshared
DEFINE	GRAF 3148	Nonshared
DEFINE	GRAF 3158	Nonshared
DEFINE	GRAF 3036	Nonshared
DEFINE	GRAF 3066	Nonshared
SPECIAL	3270	Shared
SPECIAL	3138	Nonshared
SPECIAL	3148	Nonshared
SPECIAL	3158	Nonshared

Miscellaneous Virtual Devices

Command	Device Spec	VCU MODE
DEFINE	LINE	Nonshared
DEFINE	TIMER	Nonshared
DEFINE	CTCA	Nonshared
DEFINE	3088	Nonshared
SPECIAL	2701 IBM	Nonshared
SPECIAL	2701 TELE	Nonshared
SPECIAL	2702 IBM	Nonshared
SPECIAL	2702 TELE	Nonshared
SPECIAL	2703 TELE	Nonshared
SPECIAL	TIMER	Nonshared
SPECIAL	CTCA	Nonshared
SPECIAL	3088	Nonshared

Appendix B. VM/SP Restrictions

A virtual machine created by VM/SP can run an IBM System/360 or System/370 operating system as long as it does not violate certain VM/SP restrictions. This appendix lists virtual machine restrictions and certain execution characteristics.

VM/SP Overview

Two components, CP and CMS, have been extensively modified and integrated into VM/370. This collective package (CP and CMS) plus Group Control System (GCS), Interactive Problem Control System (IPCS), and Transparent Services Access Facility (TSAF), is known as VM/SP. However, there are recommended licensed programs available that have been technically advanced to provide supportive function to VM/SP. Two of these licensed programs are the Remote Spooling Communication Subsystem (RSCS) Networking Version 2 (Program Number 5664-188) and VM/Pass-Through Facility (Program Number 5748-RC1).

Restrictions - Channel Program

Looping channel programs should be avoided. Execution of a backward transfer in channel CCW to an I/O CCW that will present channel end and device end at the same time could result in locking out the device as well as the channel. Users attempting to access devices on the channel will also be locked out. To recover from this state, the CP HALT command must be issued to the device or have the operator issue a system reset.

When issuing CCWs in which a data address is specified, that address must be within the virtual machine size regardless of whether data transfer is involved or not. The use of an address above the virtual machine size will result in VM/SP forcing a channel program check. Note that one RCWTASK consisting of a header, CCWs, and IDAWS cannot exceed X'7FFF' doublewords in length. When this occurs, VM/SP will force a channel program check.

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 I/O ends, either by the channel program itself or by the processor).

Exceptions (that is, dynamically modified channel programs CP allows) are:

- 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
- Those generated by the OS/VS Telecommunications Access Method (TCAM) Level 5, with the VM/SP option
- Those containing polling sequences.

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/SP directory entry. There is no such restriction for DOS ISAM, or for ISAM if it is running in an OS/VS virtual=virtual partition. To run ISAM in an OS/VS virtual=real partition, you must specify the ISAM option in the VM/SP directory entry for the OS/VS virtual machine.

Virtual machines using OS/VS TCAM (Level 5, generated or invoked with the VM/SP 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.

When a virtual machine starts a channel program containing a polling sequence, the CCW translation sets a PCI bit in the real CCW string. Each time the real CCW string is executed, the resulting PCI interruption causes CP to examine the corresponding virtual CCW string for changes. Any changes to the virtual CCW string are also made to the real CCW string while it is executing.

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/SP 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 I/O operation.

2. In the case of read device characteristics to an FB-512 device with the skip bit off, VM/SP modifies the data in user storage at completion of the channel program so the data reflects the true minidisk size and characteristics. Therefore, the data buffer area cannot be dynamically modified during the I/O operation.

   Note: The user should not attempt to use this data during the I/O operation.

3. On a minidisk, if a CCW string uses multitrack search on I/O operations, further operations to that disk must have preceding seeks or continue to use multitrack operations. There is no restriction for dedicated disks.

4. 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 0). There is no such restriction for DOS ISAM or OS/VS ISAM running virtual=virtual.

   Note: Because a VS1 system using VM handshaking does no paging, any ISAM programs run under VS1 are treated by VM/SP as though they are running in an ADDRSPC=REAL partition.

5. If your channel program for a count-key-data minidisk does not perform a seek operation, VM/SP inserts a positioning seek operation into the program to prevent accidental accessing. Thus, certain channel programs may generate a condition code (CC) of 0 on an SIO instead of an expected CC of 1, 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, 3350, 3375, or 3380 device may give results on dedicated drives that differ from results on minidisks having nonzero 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 end the program. This is because the record 0 count fields on these devices 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 end prematurely if a real record 0 is encountered.

   Notes:

   a. Minidisks with nonzero relocation factors on 3330, 3340, 3350, 3375, or 3380 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.

   b. If the 'R' byte field of 'CCHHR' = 0 at the time a virtual SIO is issued, but the 'CCHHR' field is read in dynamically by the channel program before the Search ID CCW is executed, the real Search ID CCW will use the relocated 'CCHHR' field that was dynamically read in, causing incorrect results. To avoid this problem, the 'R' byte of 'CCHHR' should not be defaulted to binary zero by the virtual machine if the search arguments are to be read in dynamically and a Search ID on "Record R0" is not desired.

7. If the DASD channel programs directed to 3330/3340/3350/3375/3380 devices include a write record R(0), results differ depending on whether the 3330/3340/3350/3375/3380 is dedicated or nondedicated. A full-pack minidisk is treated the same as any nondedicated device. For a dedicated 3330/3340/3350/3375/3380, a write R(0) is allowed, but you must be aware that the track descriptor record may not be the same from one 3330/3340/3350/3375/3380 to another. For a nondedicated 3330/3340/3350/3375/3380, 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.

8. 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 wanted.

9. When performing DASD I/O to 2305 devices, VM will not relocate the search argument in the Search ID command. This is because the Write Record R(0) is allowed to pass through to the hardware. Therefore, for minidisks, the cylinder address in record zero will contain the virtual cylinder address after a Write R(0) has been issued.

10. On FB-512 devices, the use of the CE area is different for dedicated devices and minidisks. Any user with a dedicated device can use the CE area. However, only class F users can use the CE area for minidisks.

11. FB-512 diagnostic commands are also handled differently for dedicated devices and minidisks. Any user with a dedicated device can issue diagnostic CCWs. For minidisks, however, only users with a minidisk equal to the size of the entire pack can issue a diagnostic control command. Because diagnostic sense commands must be chained from a diagnostic control command, this restriction indirectly applies to those commands also.

12. DIAGNOSTIC READ HOME ADDRESS and DIAGNOSTIC WRITE HOME ADDRESS commands are supported only for:

    - Dedicated devices
    - Minidisks that start at cylinder 0 (real).

13. See the *VM/SP Device Support Facilities User's Guide and Reference* for procedures on formatting 3375 and 3380 direct access storage for use in an OS/VS operating system running in a virtual machine.

14. When a virtual 3330 Model 1 is mapped to a real 3330 Model 11 and a virtual machine references sense information during error recovery, incorrect results will occur. Because sense byte 6, bits 1 and 2, is referenced differently by 3330 Models 1 and 11, the results will be unexpected.

15. For FBA minidisks with nonzero relocation factors, sense data passed to a virtual machine can contain incorrect values for CCHS. EREP records for these devices can also contain incorrect CCHS and RBN information.

Timing Dependencies

Timing dependencies in I/O devices or programming do not function regularly under VM/SP:

1. The following telecommunication access methods (or the designated option) violate the restriction on timing dependency by using program-controlled interrupt techniques and/or violate 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/SP 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 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 invoked by the DIAL command
   - START I/O tracing is in effect
   - The CAW is in page 0 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/SP if issued from third level storage. For example, a reset poll macro has a NOP effect if executed from virtual=virtual storage in a VS1 system running under VM/SP.

2. Programming that uses 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/SP system processing. An example of such a device is the 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 heighten use of the real channels.

5. Devices that experience extreme performance penalties if not reinstructed within a fixed interval may experience this penalty during every I/O operation. An

example is the 8809 tape drive. Setting the mode to "streaming" may actually result in a slower data rate than running in nonstreaming mode. Execution delays, caused by normal VM/SP processing, prevent a timely reinstruct and the 8809 tape drive may sustain a 1.2 second delay on every I/O operation. You must decide (based mainly on the size of the I/O buffers) between running at 100 IPS with continuous delays and running at 12.5 IPS, and set the mode accordingly.

Processor Model-Dependent Functions

On the System/370 Model 158 only, the virtual machine assist feature cannot operate at the same time with the 7070/7074 compatibility feature (No. 7117).

Programs written for processor model-dependent functions may not run properly in the virtual machine under VM/SP. The following points should be noted:

1. Programs written to examine the machine logout area do not have meaningful data because VM/SP does not reflect the machine logout data to a virtual machine.
2. The Store CPUID instruction (STIDP) returns the real machine value for the processor identification. (An exception is if you have changed the processor's serial number using the SET CPUID command. If you have done this, STIDP will return the value you set for the serial number.) 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. No simulation of other processor models is attempted by VM/SP.
4. Because an operating system's channel error recovery procedures may be processor model and channel model-dependent, operating systems that will run in a virtual machine may have to be generated for the same model of processor that VM/SP will be running on. If you IPL a uniprocessor generated system on a hardware complex of two or more processors, the STOP and START hardware feature commands should only be used with the *online* processor. A global use of STOP and START commands causes unpredictable results.

Channel Model-Dependent Functions

Channel checks (channel data check, channel control check and interface control check) no longer cause the virtual machine to be reset[28]. They are reflected to the virtual machine as other I/O errors are. This provides the operating system or other programs in the virtual machine with the opportunity to attempt recovery or close out its operation in an orderly manner. To take full advantage of this the virtual machine should abide by the following requirement:

> Each virtual channel should map to real channels of a single type. In other words, the virtual devices on a virtual channel should all map to real devices on real channels of a single type and model. These real channels should all be the same as each other, but not necessarily the same as the virtual channel.

[28] Due to the design of channel check recovery in VM/CP, a timing hole exists where multiple errors on the same channel may occur before the channel check handler has completed processing the original error. Depending on the timing sequence, many abends can occur.

If the I/O configuration of a virtual machine does not meet the above requirement, no warning message is issued and the virtual machine will run successfully until a channel check occurs. In this case, when a channel check occurs, there is a possibility that the channel extended logout data may be inconsistent with the data provided by the store channel id (STIDC) instruction.

Note: Virtual machines running CMS do not need to abide by these requirements. Here, only unit record spooling and diagnose I/O are performed. For unit record spooling there are no channel checks and for diagnose I/O, CP attempts to perform the error recovery itself.

When the store channel id instruction (STIDC) is executed in a virtual machine, it returns information from a random channel, one of many the specified virtual channel may map to. The type, model, and logout length data returned by the STIDC are the same as the real channel except that when a real channel is a block multiplexer and the virtual channel is a selector, the type field returned by STIDC indicates a selector channel.

Because the STIDC returns identifying data from the real channel, channel model-dependent error recovery procedures can use STIDC to identify the channel.

Channel extended logouts are reflected to the virtual machine in a manner that is processor model and channel model-dependent and constant with the data returned by STIDC (provided that the virtual-to-real channel mapping abides by the requirement stated previously).

A difference in the handling of channel extended logouts occurs if the virtual machine uses the bit in control register 14 to mask out channel extended logouts. In a virtual machine, any channel extended logouts that are masked out by control register 14 are lost rather than kept pending, and the logout pending bit (bit 5) in the CSW is never set. However, channel extended logouts will not be lost when they are kept pending along with their associated I/O interrupts by the channel masks in control register 2 and the PSW. Regardless of whether or not the setting of the virtual machine's control register 14 causes it to lose the channel extended logout, CP will still successfully record the logout in its own error recording cylinders.

Virtual Machine Characteristics

Other characteristics that exist for a virtual machine under VM/SP are as follows:

1. If the virtual=real option is selected for a virtual machine, I/O operations specifying data transfer into or out of the virtual machine's page 0, 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 processor and channels operates in these cases, but is unable to provide predictable protection to other virtual machines. In addition, violation of this restriction may compromise the soundness of the system. The results are unpredictable.
2. A two-channel switch can be used between the processor running a virtual machine under VM/SP and another processor.
3. The DIAGNOSE instruction cannot be issued by the virtual machine for its normal function. VM/SP 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. See the *VM System Facilities for Programming*

book for more information about the DIAGNOSE instruction in a virtual machine.

4. A control unit, usually, 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 of dedicated unit record devices or certain self-modifying IPL sequences of tape devices.

6. VM/SP spooling does 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/SP does not support count control on the virtual 1052 operator's console.

8. Programs that use the integrated emulators function only if the real system has the appropriate compatibility feature. VM/SP does not attempt simulation. The DOS emulator running under OS or OS/VS is not supported under VM/SP.

9. The READ DIRECT and WRITE DIRECT instructions are not supported for a virtual machine.

10. The SET CLOCK instruction cannot be simulated and is ignored if issued by a virtual machine. The STORE CLOCK instruction is a nonprivileged instruction and cannot be trapped by VM/SP; it provides the true TOD clock value from the real processor.

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 through a virtual 2701, 2702, or 2703 line.

12. The pseudo-timer (usually device address 0FF, 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 overlaid page is either the result of the following formula:

```
virtual machine size
——————————————————— = starting address of the overlaid page

         2
```

or the hexadecimal value 20000, whichever is smaller.

14. When CMS is IPLed by name and the virtual storage size is too small to contain all of the saved CMS pages (as specified by SYSPGNM in DMKSNT), the CMS saved system is treated by CP as a saved segment. The virtual machine's segment table is expanded to address all of the virtual storage needed for CMS. If there are shared segments in CMS beyond the VMSIZE (field in VMBLOCK representing the storage size of the virtual machine), any subsequent IPL causes the saved CMS to be purged in its entirety. The segment table is adjusted back to the VMSIZE, and all virtual pages associated with the CMS saved system within the VMSIZE are cleared. This means that if data is stored into CMS saved pages in low storage, the pages will be cleared, even if the NOCLEAR option is specified for the IPL. To make the data available for the IPL, either define storage large enough to accommodate all of CMS or use the STOP option on the second IPL, store the date required for the IPL, and continue with the BEGIN command.

15. To maintain data 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 ended with an interrupt whose CSW 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.

16. 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/SP, the contingent connection is maintained until the sense command is executed, but I/O activity from other virtual machines or another processor can begin on the device while the sense data is being reflected to the virtual machine. Therefore, you should be aware that on a shared disk, the access mechanism may have moved during this time.

17. 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, thus changing the mode setting of all 7-track tape devices attached to that control unit.

18. A shared system or one that uses saved segments cannot be loaded (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.

20. The 3066 is supported as a 3215. It is not supported as a graphics editor; therefore, it is recommended that the NODISP option of the EDIT command be used when editing in a 3066.

21. The Program Controlled Interruption (PCI) FETCH option for load module calling is not supported for OS/MFT or VS1.

22. 3081 processors running in S-370 mode do not permit use of 1 MB segments for virtual machines. Any attempt by a relocatable virtual machine (V=V) using 1MB segments to use the DAT facility for address translation results in a translation exception.

23. The Input/Output Configuration Program must not be run while single processor mode is active on the system. Objectionable results may occur.

24. OS/VS2 is supported in uniprocessor mode only.

25. The 3800 Printing Subsystem Model 3 is compatible with the 3800 Printer Model 1, letting application programs written for the 3800 Printer Model 1 run with little or no change on a 3800 Printer Model 3. However, user designed Library Character Sets (LCSs) and Graphic Character Modification Modules (GRAPHMODs) must be rescaled to the new PEL density. Recode the LCSs and GRAPHMODs, and then use the GENIMAGE command, which invokes the IEBIMAGE utility program, to create the LCSs and GRAPHMODs. (You may want to use the Character Conversion Aid, an MVS only licensed program that converts PEL density.)

26. When a guest virtual machine, capable of using DIAGNOSE code X'98', is notified by VM of a unit check, the guest will use the SIO/SIOF interface as opposed to the DIAGNOSE code X'98' interface to issue the sense CCW. The result could be incorrect sense data being returned to the guest if the DIAGNOSE code X'98' is used.

MSS Restrictions

1. There are two OS/VS system data sets associated with a Mass Storage System; the mass storage volume inventory and the mass storage volume control journal. There is one copy of each data set per Mass Storage System; not necessarily one per operating system. If more than one OS/VS system (running in either native mode or in a virtual machine) is connected to a common Mass Storage System, then the OS/VS systems must share a common inventory and journal.

2. When a real 3330V device is dedicated to a virtual machine as a virtual 3330V, the programming support in the virtual machine must recognize and access the virtual device as a 3330V.

3. The following must be the same: the definition of 3330V addresses in the MSC tables, the DMKRIO module, and the IOGEN for any OS/VS system running in a virtual machine with a dedicated MSC port. The reason for this, and the way to ensure it, is explained in the *VM/SP Administration* book.

4. Each active volume in the MSS must have a different volume number. If you wish to have two or more user volumes having the same volume serial (such as different versions of an OS/VS2 system residence volume both having a volume serial of VS2037), then create two MSS volumes having different volume serials and allocate the user volumes as minidisks.

5. Mass Storage System volumes may not be used for VM/SP residence, paging, spooling, or temporary disk space.

6. You must not change the volume serial of a real 3330V volume (the volume serial as known by the MSC) except by using the OS/VS access method services utilities. If, for example, cylinder 0 of a 3330V is dedicated to a virtual machine and that virtual machine alters the volume serial using DDR, then the volume cannot be mounted.

7. CP commands that require action from the central server must not be issued from the central server virtual system.
8. If virtual volumes are to be shared between processors, the virtual machine must handle cylinder faulting.

CMS Restrictions

The following restrictions apply to CMS, the conversational subsystem of VM/SP:

1. CMS runs only on a virtual processor provided by VM/SP.
2. The maximum sizes (in cylinders or blocks) of CMS minidisks are as follows:

Device Type	Model(s)	CMS/VSAM	CMS 800-byte Format	CMS 512, 1K, 2K, or 4K Format
3310	-	126,016 blocks	Not Supported	126,016 blocks
3330	1 or 2	404 cyls.	246 cyls.	404 cyls.
3330	11	808 cyls.	246 cyls.	808 cyls.
3333	1	404 cyls.	246 cyls.	404 cyls.
3333	11	808 cyls.	246 cyls.	808 cyls.
3340	35	348 cyls.	348 cyls.	348 cyls.
3340	70	696 cyls.	682 cyls.	696 cyls.
3350	native mode	555 cyls.	115 cyls.	555 cyls.
3370	A1 or B1	558,000 blocks	Not Supported	558,000 blocks
3370	A2 or B2	712,752 blocks	Not Supported	712,752 blocks
3375	-	959 cyls.	182 cyls.	959 cyls.
3380	AD4 or BD4	885 cyls.	121 cyls.	885 cyls.
3380	AE4 or BE4	1,770 cyls.	121 cyls.	1,770 cyls.
3380	AK4 or BK4	2,655 cyls.	121 cyls.	2,655 cyls.
9313	A01 or B01	246,420 blocks	Not Supported	246,420 blocks
9332	400 or 402	360,036 blocks	Not Supported	360,036 blocks
9332	600 or 602	554,816 blocks	Not Supported	554,816 blocks
9335	A01 or B01	804,714 blocks	Not Supported	804,714 blocks

3. If CMS cannot calculate a true time, it will display *.** in place of n.nn or x.xx.
4. Programs that operate under CMS are encouraged to use documented interfaces. Those programs that modify DMSNUC or other CMS control blocks to accomplish their interfaces with the CMS system, may hamper the performance and reliability of the system.
5. CMS uses VM/SP spooling to perform unit record I/O. However, a program running under CMS can issue its own SIOs to attached dedicated unit record devices.
6. Only those OS and VSE tasks that are simulated by CMS can be used to run OS and VSE programs produced by language processors under CMS.
7. Many types of object programs produced by CMS (and OS) languages can be run under CMS using CMS's simulation of OS supervisory functions. Although supported in OS and VSE virtual machines under VM/SP, the writing and

updating of non-VSAM OS data sets and VSE files are not supported under CMS.

8. CMS can read sequential and partitioned OS data sets and sequential VSE files, by simulating certain OS and VSE system services.

   The following restrictions apply when CMS reads OS data sets that reside on OS disks:

   - Read-password-protected data sets are not read unless they are VSAM data sets.
   - Multivolume data sets are read as single volume data sets. End-of-volume is treated as end-of-file and there is no end-of-volume switching.
   - BDAM and ISAM data sets are not read.
   - Keys in data sets with keys are ignored and only the data is read, except for VSAM.
   - User labels in user labeled data sets are ignored.

   The following restrictions apply when CMS reads VSE files that reside on DOS disks:

   - Only VSE sequential files can be read. CMS options and operands that do not apply to OS sequential data sets (such as the MEMBER and CONCAT options of FILEDEF and the PDS option of MOVEFILE) also do not apply to VSE sequential files.
   - The following types of VSE files cannot be read:
     - VSE DAM and ISAM files
     - Files with the input security indicator on
     - VSE files that contain more than 16 extents. (*Note:* User labels occupy the first extent; therefore, the file can hold only 15 additional data extents.)
   - Multivolume files are read as single volume files. End-of-volume is treated as end-of-file and there is no end-of-volume switching.
   - User labels in user labeled files are ignored.
   - Because VSE files do not contain BLKSIZE, RECFM, or LRECL parameters, these parameters must be specified by way of the FILEDEF or DCB parameters; otherwise, BLKSIZE=32760 and RECFM=U are assigned. LRECL is not used for RECFM=U files.
   - CMS does not support the use of OS/VS DUMMY VSAM data sets at program execution time, because the CMS/DOS implementation of the DUMMY statement corresponds to VSE implementation. Specifying the DUMMY option with the DLBL command will cause an execution-time error.

9. Assembler program usage of the ISAM Interface Program (IIP) is not supported.

10. CMS/DOS support is based on the VSE/Advanced Functions program product. With VSE, prior releases of VSAM are not supported under CMS/DOS.

11. System logical units (SYSIN, SYSRDR, SYSIPT, SYSLST, and SYSPCH), are not supported for VSE formatted FB-512 devices because the SYSFIL function (SVC 103) of VSE is not supported under CMS/DOS.

12. Programs created using CMS/DOS are not recommended for transfer directly to a VSE machine because:

    - The CMS/DOS VSE linkage editor is designed to link edit VSE programs under CMS/DOS only.
    - Programs created using the CMS/DOS assembler may have incorrect ESDs. In this case, the OS assembler is used. The OS assembler is *not* compatible with VSE.
    - Some VSE macros and SVCs are simulated. The code generated is not complete under CMS/DOS.

13. Setting the PSW EC mode bit on is not recommended because CMS handles interrupts in BC mode only.

14. To ensure that the saved copy of the S-STAT or Y-STAT is current, a validity check is performed when a saved system is IPLed. (The S-STAT and Y-STAT are blocks of storage that contain the file status tables associated with the S-disk and Y-disk respectively.) This check is performed only for S-DISKS and Y-DISKS formatted in 512-, 1024-, 2048-, or 4096-byte CMS blocks. For 800-byte block disks, the saved copy of the S-STAT or Y-STAT is used. The validity checking consists of comparing the date when the saved directory was last updated with the date when the current disk was last updated. If the dates for the S-STAT are different, then the S-STAT is built in user storage. If the dates for the Y-STAT are different, then the Y-disk is accessed using the CMS ACCESS command: ACCESS 19E Y/S * * Y2[29]. This means that even when the S- and Y-disks are accessed in read/write mode and then RELEASED, the message DMSINS100W S-STAT and/or Y-STAT NOT AVAILABLE will result.

15. Programs that modify the file ID of an FST can destroy the integrity of the file system, and they are not supported by CMS. These programs may cause a "file not found" condition for the file until the disk is accessed again.

16. When loading text that contains dummy sections or defines pseudo registers, their cumulative length must not exceed 32767 (decimal) or X'7FFF'. If this limit is exceeded, the values stored by the CXD entry will not be accurate and the load map will not contain the correct cumulative length values.

Miscellaneous Restrictions

For VM in General

1. When defining virtual devices for a virtual machine, be aware that tables for virtual devices must reside in contiguous storage. Therefore, 2 contiguous pages (a page is 512 doublewords) of free storage must be available to logon a virtual machine with more than 46 virtual devices, otherwise an error message is issued. The larger the real machine size, the lesser the possibility of this occurring. Contiguous pages of free storage are sure to be available only immediately after IPL, before other virtual machines have logged on. Therefore, a virtual machine with more than 46 devices should be the first to logon after IPL.

[29] The DASD address of the Y-DISK will be whatever was specified when CMS was generated. For the standard system this will be 19E.

To calculate the number of contiguous pages of free storage needed for a given number of virtual devices multiply that number by the VDEVBLOK size (see the *VM/SP CP Data Areas and Control Blocks* for the VDEVBLOK size). If your system is generated with the CP FRET Trap option add 3 doublewords to that total.

The design of the virtual control block structure has 32767 (X'7FFF') as the maximum index range for a virtual device table. Therefore the maximum number of virtual devices allowed for a virtual machine (if enough contiguous pages of free storage are available) is determined by dividing 32767 by the VDEVBLOK size (in bytes). Thus the maximum number of virtual devices allowed for a virtual machine decreases whenever the VDEVBLOK size increases.

2. CP checks any attempts to construct a virtual device configuration that would mix SHARED and NONSHARED device types on the same virtual control unit. If a violation is encountered, CP rejects the offending command and sends an error message to the user who entered the command. This restriction affects the following situations: CP Directory Updates, Logon Processing, and certain CP console functions.

   As the CP Directory source file is processed, those statements that describe virtual devices are checked for a subchannel protocol conflict. If a conflict is detected CP sends an error message, directory processing continues but the directory cylinders are not updated. Only the effects of the T-DISK options of the MDISK statement and the CONSOLE, SPECIAL, and SPOOL statements are considered during CP Directory Update.

   As the user enters the system (through LOGON), CP checks the devices represented by each CONSOLE, DEDICATE, LINK, MDISK, SPECIAL and SPOOL statements. If a violation is encountered the command is rejected and an error message is sent.

   When the CP console commands ATTACH, DEFINE, LINK, or NETWORK ATTACH are issued to create or move a virtual device, CP checks to see if a potential conflict could occur on the virtual control unit that would support the new device. If a conflict is detected, the operation is not performed and CP sends the user an error message.

   The following is an example of a user encountering the subchannel protocol restriction.

```
cp define graf as 4A0 3270
```

```
GRAF 4A0 DEFINED
```

```
cp define printer as 4A8 1403
```

```
DMKDEF331E  4A8 NOT DEFINED - USE A NONSHARED VCU INSTEAD
```

   In the above example, the 3270 is a shared device and the 1403 printer is nonshared. Therefore, a protocol conflict was encountered by CP and the error message was sent to the user.

   See "Matching Hardware to Shared or Nonshared Virtual Control Units" on page 407 to become familiar with the different virtual devices that are available.

3. Logical device support is not designed to simulate all aspects of real device support. Some instances are:

   - Logical device support always passes channel end and device end to the virtual machine together.
   - The PCI bit in the CCW is not handled by logical device support.
   - Ending status on I/O only is passed back to the virtual machine (not initial).

   VM/SP does not support a write for position CCW chained to another write CCW. If this sequence does occur, it will be treated as two chained write CCWs.

4. A delayed or an undefined PF key used with a terminal having an inhibited (nondisplay) read up causes the input area to be rewritten without the inhibited attribute byte. At this point any data typed in is displayed. To rewrite the inhibited read, press clear after using the PF key.

5. VM stacks CPEXBLOKs to the device dependent error recovery modules to defer error processing. It is possible for more interrupts to occur before the error processing is complete. When this happens, VM may stack another CPEXBLOK to the ERP module for the same IOBLOK.

6. When you generate a new nucleus and change unit record device configurations in DMKRIO, be sure to drain all unit record devices before shutting down the system. If you do not drain all unit record devices, you will get unpredictable results when you IPL the new system. Also, unpredictable results will occur if you warmstart. This is because a warm IPL will try to recover device information that was saved at shutdown and this information may no longer be valid.

Relating to I/O

1. The number of pages used for I/O must not exceed the total number of user pages available in real storage. Violation of this restriction causes the real system to be put into an enabled wait state.

2. If an I/O device (such as a disk or tape drive) drops ready while it is processing virtual I/O activity, any virtual machine users performing I/O on that device are unable to continue processing or to log off. Also, the LOGOFF and FORCE commands are not effective because they do not complete until all waiting I/O is finished. The system operator should determine which I/O device is involved and make that device ready once more.

3. If the system terminates the Input/Output Configuration Program (IOCP) prematurely while an input/output configuration data set write is in process,the CP IOCP write lock remains set. To clear the lock, do one of the following:

   - Rerun the IOCP to completion
   - Perform a virtual system restart
   - Re-IPL the virtual machine.

4. A virtual machine should not issue a CLEAR CHANNEL (CLRCH) to any dedicated channel. If the CLRCH is issued, the results will be unpredictable.

About Specific Devices

1. When using two channel-to-channel adapters (dedicated to virtual machines), and the CTCAs are operating on the same channels on each CPU, then the virtual machines should use the control CCW to prevent locking out the channel.

2. If you are using 3031, 3032, or 3033 processor, you must dedicate the service record file (SRF) device to VM/SP. Thus, the channel on which the SRF is located cannot be dedicated to any virtual machine.

3. When using the 3081 processor, V=V users can no longer use 1MB segments for constructing shadow tables.

4. For remote 3270s, VM/SP supports a maximum of 256 binary synchronous lines minus the number of 3704/3705 Communication Controllers.

5. If a 3278 Model 4 is switched to alternate mode (43 line screen) and the terminal is then dialed to a virtual machine, the terminal will not be reset to default mode (24 line screen). The 3278 Model 4 will remain in alternate mode if alternate mode is started after the logo has been written and an erase write alternate has been issued to the screen.

6. VM/SP CMS support of the 3290 Information Panel is limited to the XEDIT function.

   The CP function requires a minimum screen size of 3 rows by 50 columns. If you define a screen size smaller than this minimum, your screen will become the default size of 24 rows by 80 columns. The CMS full screen function requires a minimum screen size of 24 rows by 80 columns. Logical screen sizes ranging from the minimum up to 62 rows by 160 columns (the maximum for the 3290 Information Panel) are acceptable for CMS.

   Many licensed programs that run under VM/SP require running in 3278 compatibility mode. For these licensed programs, you must use the screen size associated with the model number defined during system generation. If you are using one of these licensed programs and you use a different screen size, unpredictable results may occur. The 3278 model numbers and associated screen sizes are:

Model Number	**Screen Size**
Model 2A or 2C	20 rows by 80 columns
Model 2	24 rows by 80 columns
Model 3	32 rows by 80 columns
Model 4	43 rows by 80 columns
Model 5	27 rows by 132 columns

   Note: The 3380 DASD Models AE4/BE4 may exceed the capacity of the CMS file system (having 1K, 2K, or 4K blocksizes). If this DASD is established as one minidisk and 1-byte records are created, the fullword limit for fields in the file system will be exceeded.

   In the XEDIT environment, if the screen size is defined to be larger than 150 columns, the copy key will work only if the virtual printer is defined as 3800.

   Users of the 3290 Information Panel should be aware that there are maximum lengths for input and output in VM/SP. In some cases, the command line of the

3290 allows more input characters than the maximum length of a command. To avoid losing part of a command due to truncation, you should make sure that your commands do not exceed the maximum lengths. The following is a list of input and output maximum lengths in characters.

CMS ENVIRONMENT	Length
Output	130
Input command line	130
Tokenized Parameter list (66 tokens)	528

XEDIT ENVIRONMENT	
CP commands	240
XEDIT subcommand from command line	255
Input after the logical line end character in TYPEWRITER mode	130
RETRIEVE or ? redisplays last input line(s) up to the maximum	255

CP ENVIRONMENT	
CP commands	240
RETRIEVE or ? redisplays last input line(s) up to the maximum	240

7. VM support of the 3800 printer as a nondedicated virtual spooling device differs from VM support of other virtual spooling devices, such as a 3211 printer. When a virtual spool file is closed, the loaded FCB between spool files is not kept. When a spool file is opened, a default FCB is set. Hex 63 (LOAD FCB) CCW allows changes to the default.

8. If you are using an 8809 tape device, it is required to have a tape mounted with the drive ready before issuing a CP DETACH command. This lets the tape drive mode be returned to the default mode when execution of the command completes.

9. In an attached processor (AP) or multiprocessor (MP) environment, when using virtual channel-to-channel adapters or virtual 3088s, the protocol used should be such that both sides are not actively driving the channel program (when *control* or *prepare* CCWs are providing synchronization).

   If both sides are actively driving their program, it is possible to receive an attention interrupt on one of the sides and if that side issues a sense command byte (X'14') CCW, the status returned can be X'00'. This may occur more frequently on a lightly loaded system (that is, in a test environment).

   If this does occur, it is recommended to change the protocol of the applications such that one side is the driver, and the other acts only when it receives an attention interrupt. Or, use the control or prepare CCWs to establish a synchronized protocol. If the preceding occurs in a test environment, it can also be resolved by testing on a uniprocessor (UP) system.

Involving Commands or Features

1. Any modifications to OPTIONS COPY file, unless otherwise specified in existing documentation, is not supported.
2. When TERMINAL CONMODE 3270 is in effect, unpredictable results may occur:
   - With a guest SCP such as MVS unless SCRNSAVE ON is specified.
   - If tracing is done at the same console.
3. Power on and enable (NETWORK ENABLE) a device with advanced features before entering NETWORK ATTACH or the advanced features will be nonoperational.
4. In Single Processor Mode, CP-owned volumes must be on strings and control units that are not online to the MVS native side.
5. Users with the cross memory feature (No. 6850) installed and MVS Release 2 or 3, cannot use the single processor mode (SPM) or nondisruptive transition (QVM) functions of VM/SP.
6. VM/SP does not support saving the system at the VM READ. For information about saving the system, see the *VM/SP Installation Guide.*

Appendix C. Worksheet to Aid in Coding the NAMESYS Macro

Seg	Page	Hex Address	Name	Seg	Page	Hex Address	Name
0	0	0		33	528	210000	
1	16	10000		34	544	220000	
2	32	20000		35	560	230000	
3	48	30000		36	576	240000	
4	64	40000		37	592	250000	
5	80	50000		38	608	260000	
6	96	60000		39	624	270000	
7	112	70000		40	640	280000	
8	128	80000		41	656	290000	
9	144	90000		42	672	2A0000	
10	160	A0000		43	688	2B0000	
11	176	B0000		44	704	2C0000	
12	192	C0000		45	720	2D0000	
13	208	D0000		46	736	2E0000	
14	224	E0000		47	752	2F0000	
15	240	F0000		48	768	300000	
16	256	100000		49	784	310000	
17	272	110000		50	800	320000	
18	288	120000		51	816	330000	
19	304	130000		52	832	340000	
20	320	140000		53	848	350000	
21	336	150000		54	864	360000	
22	352	160000		55	880	370000	
23	368	170000		56	896	380000	
24	384	180000		57	912	390000	
25	400	190000		58	928	3A0000	
26	416	1A0000		59	944	3B0000	
27	432	1B0000		60	960	3C0000	
28	448	1C0000		61	976	3D0000	
29	464	1D0000		62	992	3E0000	
30	480	1E0000		63	1008	3F0000	
31	496	1F0000		64	1024	400000	
32	512	200000		65	1040	410000	

Seg	Page	Hex Address	Name	Seg	Page	Hex Address	Name
66	1056	420000		101	1616	650000	
67	1072	430000		102	1632	660000	
68	1088	440000		103	1648	670000	
69	1104	450000		104	1664	680000	
70	1120	460000		105	1680	690000	
71	1136	470000		106	1696	6A0000	
72	1152	480000		107	1712	6B0000	
73	1168	490000		108	1728	6C0000	
74	1184	4A0000		109	1744	6D0000	
75	1200	4B0000		110	1760	6E0000	
76	1216	4C0000		111	1776	6F0000	
77	1232	4D0000		112	1792	700000	
78	1248	4E0000		113	1808	710000	
79	1264	4F0000		114	1824	720000	
80	1280	500000		115	1840	730000	
81	1296	510000		116	1856	740000	
82	1312	520000		117	1872	750000	
83	1328	530000		118	1888	760000	
84	1344	540000		119	1904	770000	
85	1360	550000		120	1920	780000	
86	1376	560000		121	1936	790000	
87	1392	570000		122	1952	7A0000	
88	1408	580000		123	1968	7B0000	
89	1424	590000		124	1984	7C0000	
90	1440	5A0000		125	2000	7D0000	
91	1456	5B0000		126	2016	7E0000	
92	1472	5C0000		127	2032	7F0000	
93	1488	5D0000		128	2048	800000	
94	1504	5E0000		129	2064	810000	
95	1520	5F0000		130	2080	820000	
96	1536	600000		131	2096	830000	
97	1552	610000		132	2112	840000	
98	1568	620000		133	2128	850000	
99	1584	630000		134	2144	860000	
100	1600	640000		135	2160	870000	

Seg	Page	Hex Address	Name	Seg	Page	Hex Address	Name
136	2176	880000		171	2736	AB0000	
137	2192	890000		172	2752	AC0000	
138	2208	8A0000		173	2768	AD0000	
139	2224	8B0000		174	2784	AE0000	
140	2240	8C0000		175	2800	AF0000	
141	2256	8D0000		176	2816	B00000	
142	2272	8E0000		177	2832	B10000	
143	2288	8F0000		178	2848	B20000	
144	2304	900000		179	2864	B30000	
145	2320	910000		180	2880	B40000	
146	2336	920000		181	2896	B50000	
147	2352	930000		182	2912	B60000	
148	2368	940000		183	2928	B70000	
149	2384	950000		184	2944	B80000	
150	2400	960000		185	2960	B90000	
151	2416	970000		186	2976	BA0000	
152	2432	980000		187	2992	BB0000	
153	2448	990000		188	3008	BC0000	
154	2464	9A0000		189	3024	BD0000	
155	2480	9B0000		190	3040	BE0000	
156	2496	9C0000		191	3056	BF0000	
157	2512	9D0000		192	3072	C00000	
158	2528	9E0000		193	3088	C10000	
159	2544	9F0000		194	3104	C20000	
160	2560	A00000		195	3120	C30000	
161	2576	A10000		196	3136	C40000	
162	2592	A20000		197	3152	C50000	
163	2608	A30000		198	3168	C60000	
164	2624	A40000		199	3184	C70000	
165	2640	A50000		200	3200	C80000	
166	2656	A60000		201	3216	C90000	
167	2672	A70000		202	3232	CA0000	
168	2688	A80000		203	3248	CB0000	
169	2704	A90000		204	3264	CC0000	
170	2720	AA0000		205	3280	CD0000	

Seg	Page	Hex Address	Name	Seg	Page	Hex Address	Name
206	3296	CE0000		232	3712	E80000	
207	3312	CF0000		233	3728	E90000	
208	3328	D00000		234	3744	EA0000	
209	3344	D10000		235	3760	EB0000	
210	3360	D20000		236	3776	EC0000	
211	3376	D30000		237	3792	ED0000	
212	3392	D40000		238	3808	EE0000	
213	3408	D50000		239	3824	EF0000	
214	3424	D60000		240	3840	F00000	
215	3440	D70000		241	3856	F10000	
216	3456	D80000		242	3872	F20000	
217	3472	D90000		243	3888	F30000	
218	3488	DA0000		244	3904	F40000	
219	3504	DB0000		245	3920	F50000	
220	3520	DC0000		246	3936	F60000	
221	3536	DD0000		247	3952	F70000	
222	3552	DE0000		248	3968	F80000	
223	3568	DF0000		249	3984	F90000	
224	3584	E00000		250	4000	FA0000	
225	3600	E10000		251	4016	FB0000	
226	3616	E20000		252	4032	FC0000	
227	3632	E30000		253	4048	FD0000	
228	3648	E40000		254	4064	FE0000	
229	3664	E50000		255	4080	FF0000	
230	3680	E60000		256	4096	1000000	
231	3696	E70000					

Appendix D. Sample SNTMAP Output

DASD SNTMAP

```
                 S A V E D   S E G M E N T   D A S D   L A Y O U T

                  SNT file used:   DMKSNT 3380
                  Directory used:  DIRECT 3380

                 Volume Segment   Segment  Start    End      Number
                 Label  Name      Location Cyl/Pg#  Cyl/Pg#  Of Pages
                 ------ --------  -------- -------- -------- --------

                 VMPK01 -  3380
                        GCS       064-079  0012,001 0013,114   264
                        CMSFILES  156-171  0013,115 0015,071   257
                                                              7879   ** GAP **

                 VMSRES -  3380
                        CMS       224-255  0006,001 0009,096   546
                        CMSINST   212-215  0009,097 0010,011    65
                        HELP      207-211  0010,012 0010,076    65
                        CMSDOS    206      0010,077 0010,093    17
                        CMSBAM    192-194  0010,094 0010,142    49
                        CMSVSAM   185-190  0010,143 0011,105   113
                        CMSAMS    176-181  0011,106 0012,100   145
                        VMEP01    (NONE)   0012,101 0012,116    16
                                                                 1   ** GAP **
                        VMEP02    (NONE)   0012,118 0012,133    16
                                                                 1   ** GAP **
                        CMSVMLIB  172-175  0012,135 0013,049    65
                                                               551   ** GAP **


                     FREE DASD SPACE - SAVESYS AREA GAPS

                 Volume DASD       Start      End        Number
                 Label  Type       Cyl/Pg#    Cyl/Pg#    Of Pages
                 ------ --------   --------   --------   --------
                 VMSRES 3380       0012,117   0012,117          1
                 VMSRES 3380       0012,134   0012,134          1
                 VMSRES 3380       0013,050   0016,150        551
                 VMPK01 3380       0015,072   0067,150       7879
```

MEMORY SNTMAP

```
V I R T U A L   M E M O R Y   M A P

  SNT file used:  DMKSNT 3380

Volume Segment  Segment   Number of
Label  Name     Location  Segments
------ -------- --------  ---------
VMPK01 GCS      064-079      16
VMPK01 CMSFILES 156-171      16
VMSRES CMSVMLIB 172-175       4
VMSRES CMSAMS   176-181       6
VMSRES CMSVSAM  185-190       6
VMSRES CMSBAM   192-194       3
VMSRES CMSDOS   206           1
VMSRES HELP     207-211       4
VMSRES CMSINST  212-215       4
VMSRES CMS      224-255      32

-------------------------------------------------------

The following macros contain unshared pages:

Segment  Shared    Actual Page
Name     Segments  Allocation
-------- --------  -----------
GCS      064-079   0-6,1024-1279
CMSAMS   176-181   2816-2959
CMSVSAM  185-190   2960-3071
CMS      224-255   0-13,16-34,3584-4095

Segment overlays do not work in the same virtual machine.
```

Summary of Changes

How to Obtain Prior Editions of This Book

Previous editions of these books can be ordered using the pseudo-numbers found in the *VM/SP Library Guide and Master Index.*

Summary of Changes
for SC19-6201-06
for VM/SP Release 6

Integration of Between-Release Support Information to VM/SP Release 6

- *VM/SP 9370 Processors, 9332 and 9335 Direct Access Storage Devices, and 9347 Tape Drive*, GC24-5315
- *VM/SP GCS/VSAM Support for Local Shared Resources/Deferred Write*, GC24-5360
- *VM/SP Transparent Services Access Facility 9370 Local Area Network Subsystems*, GC24-5371
- *VM IBM 3380 Direct Access Storage Models AJ4/BJ4 and AK4/BK4*, GC24-5371
- *VM IBM 3990 Storage Control Models 1 and 2 and IBM 3380 Direct Access Storage Direct Channel Attach Model CJ2*, GC24-5372
- *VM Diagnose Code X'E4'*, GC24-5376
- *VM/SP Automatic Re-IPL Enhancement*, GC24-5391
- *VM/SP 9332 Direct Access Storage Device Enhancements*, GC24-5376.

New Sections Added

- Planning for APPC/VM VTAM support.
- CMS Shared File System.
- Optional CMS initialization parameters added to IPL Control Statement.
- APPCPASS control statement.
- SYSIPL macro.
- Two new execs: DISKMAP and SNTMAP.
- Two new options (DEVInfo and DEVMaint) added to the OPTION Control Statement.
- Comprehensive glossary.

Miscellaneous Changes

- Callable Services Library now on CMS system disk.
- Installation procedure for saved segment and saved system has changed.
- AVS (APPC/VM VTAM Support) is a new component of VM/SP.
- VM/SP no longer supports 3330s as starter systems.
- The 2311, 2314, and 2319 storage devices are no longer supported by VM/SP.

- New printers supported by VM/SP:
  - 3812
  - 3820
  - 6262.

**Summary of Changes
for SC19-6201-05
for VM/SP Release 5**

- New book organization:
  - Release 4 Chapters 1 through 24 have been reorganized and combined into *new* Chapters 1 through 11
  - Release 4 "Chapter 19. Defining Your Own Privilege Classes" has been deleted and moved to the *VM/SP CP for System Programming* book
  - Release 4 "Appendix A. Licensed Programs and Integrated Emulators" has been deleted
  - Release 4 "Appendix C. Compatible Devices" is now part of Chapter 2
  - A Glossary has been added.
- Major new sections added:
  - 'Alternate CP Nucleus' (See Chapter 5)
  - 'How CLUSTER and TERMINAL Macros Affect Addressing' (See Chapter 8)
  - 'Planning for GCS' (See Chapter 4)
  - 'Planning for Virtual Machine Operating Systems (Other than CMS) ' (See Chapter 5)
  - 'Planning for TSAF' (See Chapter 5)
  - 'Security Considerations' (See Chapter 7).
- New tables added:
  - 'System Directory Planning Input Information' (See Chapter 7)
  - 'Planning for SNT Input' (See Chapter 10)
  - 'Minidisks Reserved for the MAINT User ID' (See Chapter 7)
  - 'Polling Definitions for Remote Clusters' (See Chapter 8)
  - 'System Name Table File Sample (DMKSNT)' (See Chapter 10)
  - 'Sample Coding for the CMS Nucleus Generation Profile (DMSNGP)' (See Chapter 11)
  - 'User IDs Reserved for System Functions' (See Chapter 7).
- Other significant changes:
  - New information on optional directory system areas has been added (See Chapter 7)
  - A new macro entry, NAMELANG, has been added to the System Name Table file (See Chapter 10)

- A new macro, DEFNUC, which defines responses to the system for prompts during CMS nucleus generation, has been added (See Chapter 11)
- VM/SP no longer supports 3340s as starter systems
- The GENERATE EXEC for multiprocessing has been replaced by SPGEN PROFILE
- The procedure for 'Saving a System' has changed.

• The following new directory control statements have been added to Chapter 7:
  - ACIGROUP
  - INCLUDE
  - PROFILE.

Summary of Changes
for SC19-6201-04
for VM/SP Release 4

• VM/SP now supports the following hardware devices:
  - 3279 Model 2C Console, in display mode
  - 3290 Information Panel
  - 3370 Direct Access Storage Models A2 and B2
  - 3480 Magnetic Tape Unit
  - 3725 Communication Controller
  - 3800 Printing Subsystem Model 3, in Model 1 compatibility mode
  - 4248 Printer
  - 4361 Model Groups 3, 4, and 5 Processor
  - 4381 Model Groups 1, 2, and 3 Processor.

• Chapter 3, "Estimating VM/SP Storage Requirements" reflects the new DASD requirements for saved systems. The SAVESYS, VMSAVE, and IPL functions can now save or restore a page image up to 16 MB long. (Previously, the limit was 8 MB.)

• "Planning for IPCS" is new. It describes how to plan for the VM/SP Interactive Problem Control System.

• "Chapter 8. Planning for Virtual Machine Operating Systems (Other than CMS)" has been removed from the book. Most of the material from that chapter is included in the book *Virtual Machine Running Guest Operating Systems.* A new chapter, "Planning for the 3081 D16 Processor," covers the remaining topics, Monitoring and Service Support Facility and the Input/Output Configuration Program, from the old Chapter 8.

• Chapter 7, "Creating Your VM/SP Directory" and Appendix A, "VM/SP Configuration Aid" describe a new restriction: you cannot mix SHARED and NONSHARED device types on the same virtual control unit.

• "Chapter 19. Defining Your Own Privilege Classes" is also new. It explains how you can expand the number of privilege classes from the original eight to 32.

• Chapter 10, "Preparing the System Name Table File (DMKSNT)," expanded from the last edition, describes a new planning tool, SNTMAP EXEC.

Glossary of Terms and Abbreviations

A

ACF/SSP. Advanced Communications Function for Systems Support Programs.

ACF/VTAM. Advanced Communications Function for Virtual Telecommunications Access Method.

Advanced Communications Function for Systems Support Programs (ACF/SSP). An IBM program product made up of a collection of utilities and small programs. SSP is required for operation of the NCP.

Advanced Communications Function for Virtual Telecommunications Access Method (ACF/VTAM). An IBM licensed program that controls communications and flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

Advanced Program-to-Program Communications (APPC). The inter-program communication service within SNA LU 6.2 on which the APPC/VM interface is based.

Advanced Program-to-Program Communications/VM (APPC/VM). An API for communicating between two virtual machines that is mappable to the SNA LU 6.2 APPC interface and based on IUCV functions. Along with the TSAF virtual machine, APPC/VM provides this communication within a single system and throughout a collection of systems.

APPC. Advanced Program-to-Program Communications.

APPC/VM. Advanced Program-to-Program Communications/VM.

APPC/VM VTAM Support (AVS). A component of VM/SP that lets application programs using APPC/VM communicate with programs anywhere in a network defined by IBM's SNA. AVS transforms APPC/VM into APPC/VTAM protocol.

API. Application program interface.

application program interface (API). The formally defined programming language interface between an IBM system component or program product and its user.

AVS. APPC/VM VTAM Support.

AVS virtual machine. The virtual machine that manages a gateway that allows communication between VM systems and an SNA network.

B

basic control (BC) mode. A mode in which additional System/370 features, such as new machine instructions, are not operational. Contrast with *extended control (EC) mode.*

basic sequential access method (BSAM). An access method for storing or getting data blocks in a continuous sequence (using either a sequential access or direct access device).

BC mode. Basic control mode.

BSAM. Basic sequential access method.

BSC. Binary synchronous communication.

C

callable services library (CSL). A package of CMS assembler routines that can be stored as an entity and made available to application programs.

CETI. Continuously Executing Transfer Interface.

channel status word (CSW). An area in storage that provides information about the termination of I/O.

channel-to-channel adapter (CTCA). A hardware device that connects two channels on the same computing system or on different systems.

channel-to-channel (CTC) device. A hardware device that connects two channels on the same computing system or on different systems. CTC devices include both CTCAs and 3088 MCUs.

class authority. Privilege assigned to a virtual machine user in the user's directory entry; each class specified allows access to a subset of all the CP commands. See *user class restructure (UCR).*

Common Programming Interface (CPI) Communications. A set of program-to-program communication routines that let applications written in REXX and high-level languages access APPC/VM functions. These routines are part of IBM's Systems Application Architecture.

Continuously Executing Transfer Interface (CETI). An interface that uses continuously executing channel programs to transfer messages between two systems, or between an application and a control unit.

CP assist. A hardware function, available only on a processor with ECPS, that reduces CP overhead by doing the most frequently used tasks of CP routines.

CPI Communications. Common Programming Interface Communications.

D

DASD Dump Restore (DDR) program. A service program that copies all or part of a minidisk onto tape, loads the contents of a tape onto a minidisk, or sends data from a DASD or from tape to the virtual printer.

DAT. Dynamic address translation.

data control block (DCB). A control block access method routines use to store and retrieve data.

discontiguous saved segment. One or more 64K segments of storage that were previously loaded, saved, and assigned a unique name. The segment(s) can be shared among virtual machines if the segment(s) contains reentrant code. Discontiguous segments used with CMS must be loaded into storage at locations above the address space of a user's CMS virtual machine. They can be detached when no longer needed.

Disk Operating System/Virtual Storage Extended (DOS/VSE). An operating system that is an extension of DOS/VS. A VSE system consists of: (a) licensed VSE/Advanced Functions support, and (b) any IBM-supplied and user-written programs required to meet the data processing needs of a user. VSE and the hardware it controls form a complete computing system.

DOS/VSE. Disk Operating System/Virtual Storage Extended.

DPA. Dynamic paging area.

dynamic address translation (DAT). In System/370 virtual storage systems, the change of a virtual storage address to a real storage address during execution of an instruction.

dynamically modified channel program. A channel program changed by the program or by data being read in from a channel during the interval between the execution of the START I/O (SIO) instruction and the channel end interruption.

E

ECB. Event control block.

EC mode. Extended control mode.

ECPS:VM/370. Extended Control Program Support:VM/370.

environmental record editing and printing program (EREP). A program that makes the data contained in the system recorder file available for further analysis.

EREP. Environmental record editing and printing program.

escape symbol. Synonym for *logical escape symbol.*

event control block (ECB). A control block that represents the status of an event.

expanded virtual machine assist. A hardware assist function, available only on a processor that has ECPS, that handles many privileged instructions not handled by VMA, and extends the level of support of certain privileged instructions beyond that provided by VMA.

extended control (EC) mode. A mode in which all features of a System/370 computing system, including dynamic address translation, are operational. Contrast with *basic control (BC) mode.*

Extended Control Program Support (ECPS:VM/370). A hardware assist feature that improves the performance of CP by reducing CP overhead. ECPS:VM/370 consists of CP assist, expanded virtual machine assist, and virtual interval timer assist.

F

FIFO (first-in-first-out). A queuing technique in which the next item to be retrieved is the item that has been on the queue for the longest time. Contrast with *LIFO (last-in-first-out).*

file pool. A collection of minidisks managed by SFS. It contains user files and directories and associated control information. Many users' files and directories can be contained in a single file pool.

G

gateway. The LU name of a TSAF collection that is a source for communications to an SNA-defined network or the target of communications from an SNA-defined network.

GCS. Group Control System.

Group Control System (GCS). A component of VM/SP, consisting of a shared segment that the user can IPL and run in a virtual machine. It provides simulated MVS services and unique supervisor services to help support a native SNA network.

guest operating system (GOS). A second operating system that runs on the user's primary operating system.

An example of a GOS is VSE running on VM/SP to support VM/VCNA.

guest virtual machine (GVM). A virtual machine in which an operating machine is running.

H

handshaking feature. See *VM/VS handshaking feature.*

host system. A data processing system that prepares programs and the operating environments for use by another computer or controller.

I

installation verification procedure (IVP). A procedure distributed with VM/SP that exercises the newly generated VM/SP system. This procedure verifies that the basic facilities of VM/SP are correctly functioning.

Interactive Problem Control System (IPCS). A component of VM/SP that permits online problem management, interactive problem diagnosis, online debugging for disk related CP or virtual machine abend dumps or CPTRAP files, problem tracking, and problem reporting.

IPCS. Interactive Problem Control System.

IUCV. Inter-user communication vehicle.

IVP. Installation verification procedure.

L

LAN. Local area network.

LIFO (last-in-first-out). A queuing technique in which the next item to be retrieved is the item most recently placed in the queue. Contrast with *FIFO (first-in-first-out).*

load map. A map containing the storage addresses of control sections and entry points of a program loaded into storage.

local area network (LAN). A data network located on the user's premises in which serial transmission is used for direct data communication among data stations. Contrast with *wide area network (WAN).*

logical escape symbol. A special editing symbol, usually the double quotation (") symbol, that causes CP to consider the immediately following character as a data character instead of as a logical editing symbol. Synonymous with *escape symbol.*

logical saved segment. A portion of a physical saved segment that CMS can manipulate. Each logical segment can contain different types of program objects, such as modules, text files, execs, callable services libraries, language repositories, user-defined objects, or a single minidisk directory. A system segment identification file (SYSTEM SEGID) associates a logical saved segment to the physical saved segment in which it resides. See *physical saved segment* and *saved segment.*

logical unit (LU). An entity addressable within an SNA-defined network, similar to a node within a VM network. LUs are categorized by the types of communication they support. A TSAF collection in an SNA network is viewed as one or more LUs.

logical unit name (LU name). A symbolic name given to a particular LU in an SNA-defined network.

M

memo-to-users. A file provided on a service tape that contains specific service information for a product.

MIH. Missing interrupt handler.

missing interrupt handler (MIH). A VM/SP facility that detects incomplete I/O conditions by monitoring I/O activity. It also tries to correct incomplete I/O conditions without operator intervention.

N

NCCF. Network Communication Control Facility.

NCP. Network control program.

Network Communication Control Facility (NCCF). An IBM licensed program consisting of a base for command processors that can monitor, control, and improve the operation of a network.

network control program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

O

OLTSEP. Online test stand-alone executive program.

online test stand-alone executive program (OLTSEP). A program IBM uses for I/O maintenance.

Operating System/Virtual Storage (OS/VS). A family of operating systems that control IBM System/360 and System/370 computing systems. OS/VS includes VS1, VS2, MVS/370, and MVS/XA.

OS/MFT. The IBM System/360 Operating System that supports multiprogramming with a fixed number of tasks.

OS/MVT. The IBM System/360 Operating System that supports multiprogramming with a variable number of tasks.

OS/VS1 nonpaging mode. If OS/VS1 executes under the control of a VM/SP system that supports the VM/VS handshaking feature and if the OS/VS1 address space is equal to the size of its VM/SP virtual machine, OS/VS1 executes in nonpaging mode. When OS/VS1 executes in nonpaging mode, it uses fewer privileged instructions and avoids duplicate paging because paging is done only by CP.

P

physical saved segment. One or more pages of storage that have been named and retained on a CP-owned volume (DASD). Once created, it can be loaded within a virtual machine's address space or outside a virtual machine's address space. Multiple users can load the same copy. A physical saved segment can contain one or more logical saved segments. A system segment identification file (SYSTEM SEGID) associates a physical saved segment to its logical saved segments. See *logical saved segment* and *saved segment.*

preferred machine assist (PMA). The hardware feature of the IBM 308X processor complex or the IBM 3033 processor that improves MVS/SP (Release 1 enhancement, or later) V = R virtual machine performance. The MVS/SP guest virtual machine operates in supervisor state with direct control of its own I/O operations under VM/SP HPO. PMA is an extension of VMA, which eliminates CP simulation of certain instructions and interruptions.

prefix storage area (PSA). A page zero of real storage that contains machine-used data areas and CP global data.

privileged instruction simulation. The CP-incurred overhead to handle privileged instructions for virtual machine operating systems that execute as if they were in supervisor state but which are executing in problem state under VM/SP. See *virtual machine assist (VMA).*

product parameter file. A file containing installation and service parameters for a product: control options, minidisk and SFS directory assignments, and component part type/function lists.

product parameter override file. A file containing one or more component override areas.

programmable operator facility. This facility enables automatic filtering and routing of messages from a specified virtual machine (for example the operator) to a logical operator virtual machine in a local distributed or mixed environment. It also permits installation defined actions to be automatically performed.

program update tape (PUT). A tape containing a customized collection of service tapes (preventive service) to match the products listed in a customer's ISD (IBM Software Distribution) profile. Each PUT contains cumulative service for the customer's products back to earlier release levels of the product still supported. The tape is distributed to authorized customers of the products at scheduled intervals or on request.

PSA. Prefix storage area.

PVM. VM/Pass-Through Facility.

Q

QSAM. Queued sequential access method.

queue-drop. The action by the system scheduler, DMKSCH, of removing a virtual machine from the list of virtual machines that can be given control of a processor.

queue-drop elimination. A VM/SP performance option that eliminates the dropping of a virtual machine from the active queue if the virtual machine is determined to be idle.

queued sequential access method (QSAM). An extended version of BSAM. When this method is used, a queue is formed of input data blocks awaiting processing or processed output data blocks awaiting transfer to auxiliary storage or to an output device.

R

remote operator console facility (ROCF). A 4300 Series Support Processor microcode function that permits communication from a remote console for functions like IML or IPL using a switched line. The VM/Pass-Through Facility program provides a communication vehicle that lets any of its supported display stations serve as this remote console.

Remote Spooling Communications Subsystem Networking (RSCS). An IBM licensed program and special-purpose subsystem that supports the reception and transmission of messages, files, commands, and jobs over a computer network.

ROCF. Remote operator console facility.

rotational position sensing (RPS). A standard or optional feature of most IBM disk storage devices. It lets these devices disconnect from a block-multiplexer channel (or its equivalent on Model 3115/3125 processing units) during rotational positioning operations, thereby allowing the channel to service other devices.

RPS. Rotational position sensing.

RSCS. Remote Spooling Communications Subsystem Networking.

S

saved segment. A segment of storage that has been saved and assigned a name. The saved segment(s) can be physical saved segment(s) that CP recognizes or logical saved segments that CMS recognizes. The segments can be loaded and shared among virtual machines, which helps use real storage more efficiently, or a private, nonshared copy can be loaded into a virtual machine. See *logical saved segment* and *physical saved segment.*

saved system. A special nonrelocatable copy of a virtual machine's virtual storage and associated registers kept on a CP-owned disk and loaded by name instead of by I/O device address. Loading a saved system by name substantially reduces the time it takes to IPL the system in a virtual machine. In addition, a saved system such as CMS can also share one or more 64K segments of reenterable code in real storage between virtual machines. This reduces the cumulative real main storage requirements and paging demands of such virtual machines.

SCP. System control programming.

server-requester programming interface (SRPI). (1) A protocol between requesters and servers in an enhanced connectivity network. Includes the protocol to define a cooperative processing subsystem. (2) The interface that enables enhanced connectivity between requesters and servers in a network.

sever. Ending communication with another virtual machine or with the user's own virtual machine.

SFS. Shared file system.

SFS communication adapter. The part of CMS in a user machine that communicates with file pool server machines.

SFS directory. A group of files. SFS directories can be arranged to form a hierarchy in which one directory can contain one or more subdirectories as well as files.

shared read-only system residence disk. A system residence disk tailored so that most of the system residence information is read-only and accessible to all relevant virtual machines, leaving a relatively smaller private read/write system disk that must be dedicated to each virtual machine. This technique can substantially reduce the disk requirements of an installation by avoiding needless duplication of disk packs by virtual machines that use the same operating system. See *saved system.*

shared segment. A feature of a saved system or physical saved segment that lets one or more segments of reentrant code or data in real storage be shared among many virtual machines. For example, if a saved CMS system was generated, the CMS nucleus is shared in real storage among all CMS virtual machines loaded by name; that is, every CMS machine's segment of virtual storage maps to the same 64K of real storage. See *discontiguous saved segment* and *saved system.*

shared system. See *saved system* and *shared read-only system residence disk.*

SNA. Systems Network Architecture.

SRPI. Server-requester programming interface.

SSP. System service program.

supervisor call instruction (SVC). An instruction that interrupts a program being executed and passes control to the supervisor so that it can do a specific service indicated by the instruction.

SVC. Supervisor call instruction.

system control programming (SCP). IBM-supplied programming fundamental to the operation and maintenance of the system. It serves as an interface with IBM licensed programs and user programs and available without additional charge.

Systems Application Architecture. A defined set of interfaces, conventions, and protocols that can be used across various IBM systems.

system service program (SSP). In ACF/TCAM, an IBM-supplied or user-supplied program that does system-oriented auxiliary functions in support of the message control program. System service programs run under control of the initiator as attached subtasks.

T

task ID. A 2-byte field that uniquely defines a task within a GCS virtual machine. Task ID is sometimes combined with machine ID to uniquely identify a task within a virtual machine group.

Transparent Services Access Facility (TSAF). A component of VM/SP that handles communication between systems by letting APPC/VM paths span multiple VM systems. TSAF lets a source program connect to a target program by specifying a name that the target has made known, instead of specifying a user ID and node ID.

TSAF. Transparent Services Access Facility.

TSAF collection. A group of VM processors, each with a TSAF virtual machine, connected by CTC, binary synchronous lines, or LANs.

U

uniprocessor mode. This term indicates that there is only one processor in the physical configuration, or that VM/SP uses the facilities of one processor in an AP or MP system (not to be confused with *single processor mode*).

user class restructure (UCR). The extension of the class structure of CP instructions from 8 to 32 classes for each user, command, and diagnose code within the system. This extension allows the installation greater flexibility in authorizing CP instructions.

V

virtual interval timer assist. A hardware assist function, available only on a processor, that has ECPS. It provides, if desired, a hardware updating of each virtual machine's interval timer at location X'50'.

virtual machine assist (VMA). A hardware feature available on certain VM/SP-supported System/370 models, that causes a significant reduction in the real supervisor state time that VM/SP uses to control the operation of virtual storage systems such as VSE, DOS/VS and OS/VS and, to a lesser extent, CMS, DOS, and OS when running under VM/SP. VM/SP supervisor state time is reduced because the VMA feature, instead of VM/SP, intercepts and handles interruptions caused by SVCs, other than SVC 76, and certain privileged instructions. See *CP assist, expanded virtual machine assist, Extended Control Program Support (ECPS:VM/370)*, and *virtual interval timer assist.*

virtual machine communication facility (VMCF). A CP function that provides a method of communication and data transfer between virtual machines operating under the same VM/SP system.

virtual machine control block (VMBLOK). The primary control block for many activities related to a single virtual machine. This block contains, for each virtual machine, the following types of information: the dispatch and priority level of the virtual machine, the virtual machine's processor registers, preferred virtual machine options currently in effect, and information concerning all other significant activities.

Virtual Machine/VTAM Communications Network Application (VM/VCNA). A program that runs in the VTAM service machine. VM/VCNA controls the physical appearance of the screen when displaying output on a VM/SP terminal attached to an SNA network.

virtual reserve/release. A function that lets many operating systems such as MVS, SVS, VS1, and VM/SP itself all run as virtual machines under the same VM/SP operating system and have data protection on a minidisk. It prevents many users of the same data file from simultaneously accessing the same data, particularly when that data is being updated.

virtual storage access method (VSAM). An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

virtual storage extended (VSE). The generalized term that indicates the combination of the DOS/VSE system control program and the VSE/Advanced Functions program product. Note that in certain cases, the term DOS is still used as a generic term; for example, disk packs initialized for use with VSE or any predecessor DOS or DOS/VS system are sometimes called DOS disks. Also note that the DOS-like simulation environment provided under the VM/SP CMS component and CMS/DOS exists on VM/SP and VM/SP HPO program products and continues to be called CMS/DOS.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in a computer network. It provides single-domain, multiple-domain, and multiple-network capability. VTAM runs under MVS, OS/VS1, VM/SP, and VSE.

VMA. Virtual machine assist.

VMBLOK. Virtual machine control block.

VMCF. Virtual machine communication facility.

VMLIB. The name of the CSL supplied with VM/SP and that contains routines to do various VM functions.

VM/Pass-Through Facility. A facility that lets VM users interactively access remote system and processor nodes. These can be remote IBM 4300 processors, other VM systems, with or without this facility installed, or System/370- compatible non-VM systems.

VM/VCNA. Virtual Machine/VTAM Communications Network Application.

VM/VS handshaking feature. A communication interface between VM/SP and other operating systems running a virtual machine under VM/SP. These operating systems and CP make each other aware of mutual capabilities and requirements.

VSAM. Virtual storage access method.

VSCS. VTAM SNA Console Support.

VSE. Virtual storage extended.

VSM. VTAM service machine.

VTAM. Virtual Telecommunications Access Method.

VTAM service machine (VSM). A virtual machine that contains an operating system (OS/VS1 or DOS/VSE), an access method (ACF/VTAM or ACF/VTAME), and VM/VCNA. VSM forms the interface for SNA communication in VM/SP.

VTOC. Volume table of contents.

W

WAN. Wide area network.

wide area network (WAN). A network that provides communication services to a geographic area larger than that served by an LAN. Contrast with *local area network (LAN)*.

Z

zap. To modify or dump an individual text file, using the ZAP command or the ZAPTEXT EXEC.

Bibliography

Related Publications

Here is a list of other IBM books that can help you plan your system. The *VM/SP Library Guide and Master Index*, GC19-6207, describes all the VM/SP books and has an index to all of them. If you do not see the book you want in this list, you might want to check the *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001.

- **Planning**

  Virtual Machine/System Product:

  Release 6 Guide, SC24-5368
  Group Control System Command and Macro Reference, SC24-5250
  Connectivity Planning, Administration, and Operation, SC24-5378
  CMS Shared File System Administration, SC24-5367

  Virtual Machine

  Running Guest Operating Systems, GC19-6212

  Remote Spooling Communications Subsystem Networking Version 2:

  Planning and Installation, SH24-5057

  Other Non-VM/SP titles:

  IBM 3850 Mass Storage System (MSS) Introduction and Preinstallation Planning, GA32-0035

  IBM 3704 and 3705 Control Program Generation and Utilities Guide and Reference Manual (OS/VS TCAM Levels 5 and 6 in VS1; VS2 Rel 1.6, 1.7, 2, SCP 5744-BA1, GC30-3008

  IBM 3704 and 3705 Control Program Generation and Utilities Guide and Reference Manual (TCAM 10 SVS - 5742-017) SCP 5742, 5744-AN1/BA2, 5747-AG1/AJ2, GC30-3008

  ACF/VTAM General Information (for VM), GC38-0254

- **Installation**

  Virtual Machine/System Product:

  Installation Guide, SC24-5237
  Service Guide, SC24-5389

  Remote Spooling Communications Subsystem Networking Version 2:

  Planning and Installation, SH24-5057

  Other Non-VM/SP titles:

  IBM 3850 Mass Storage System (MSS) Installation Planning and Table Create, GC35-0028

  VM/VTAM Communication Network Application Installation, Operation, and Terminal Use, SC27-0502

  VM/VTAM Installation and Resource Definition, SC23-0111

  ACF/NCP-SSP, V3 Installation and Resource Definition Guide, SC30-3253

  EP/3725 Installation and Resource Definition Guide and Reference, SC30-3172

  EP/3705 Generation and Utilities Guide and Reference, GC30-3242

  ACF/NCP V4, ACF/SSP V3 Diagnosis Guide, SC30-3255

- **Administration, Operation, Programming, and Diagnosis**

  Virtual Machine/System Product:

  Operator's Guide, SC19-6202
  System Messages and Codes, SC19-6204
  Terminal Reference, GC19-6206
  Application Development Guide for CMS, SC24-5286
  Administration, SC24-5285
  Connectivity Programming Guide and Reference, SC24-5377
  Connectivity Planning, Administration, and Operation, SC24-5378
  CP General User Command Reference, SC19-6211
  CP System Command Reference, SC24-5402
  Interactive Problem Control System Guide and Reference, SC24-5260

  Virtual Machine

  System Facilities for Programming, SC24-5288

  Remote Spooling Communications Subsystem Networking Version 2:

  Operation and Use, SH24-5058

  Other non-VM/SP titles:

  3704 and 3705 Communication Controllers
  Introduction to the IBM 3704 and 3705 Communications Controllers, GA27-3051
  IBM 3704 Control Panel Guide, GA27-3086
  IBM 3705 Control Panel Guide, GA27-3087
  3800 Printing Subsystem
  Introducing the 3800 Printing Subsystem, GC26-3829
  Introducing the IBM 3800 Model 3 Printing Subsystem, GA32-0049
  Concepts of the IBM 3800 Printing Subsystem, GC20-1775
  Reference Manual for the IBM 3800 Printing Subsystem, GA26-1635
  IBM 3800 Printing Subsystem Programmer's Guide, GC26-3846
  Reference Manual for the IBM 3800 Printing Subsystem Model 3, GA32-0050
  IBM Printing Subsystem Model 3 Programmer's Guide: Compatibility, SH35-0051
  3850 Mass Storage System
  IBM 3850 Mass Storage System (MSS) Principles of Operation: Theory, GA32-0035
  IBM 3850 Mass Storage System (MSS) Principles of Operation: Reference, GA32-0036
  VM/VTAM
  VM/VTAM Communication Network Application Messages, SC27-0510
  VM/VTAM Communication Network Application Logic, LY38-3033
  EREP
  EREP User's Guide and Reference, GC28-1378
  Miscellaneous
  3270 Information Display System Library User's Guide, GA23-0058
  IBM MVS/XA Linkage Editor and Loader User's Guide, GC26-4143
  Input/Output Configuration Program User's Guide and Reference, GC28-1027
  Device Support Facilities User's Guide and Reference, GC35-0033
  VSE/VSAM Programmer's Reference, SC24-5145

VM/SP RELEASE 6 LIBRARY

Evaluation

- General Information GC20-1838
- ▨ Introduction GC19-6200
- Introduction to Security SC24-5316

Installation and Service

- ▨ Installation Guide SC24-5237
- ▨ Service Guide SC24-5389

Planning

- ▨ Planning Guide and Reference SC19-6201
- VM Running Guest Operating Systems GC19-6212
- ▨ Release 6 Guide SC24-5368

Operation

- ▨ Operator's Guide SC19-6202

Administration

- ▨ VM System Facilities for Programming SC24-5288
- ▨ Administration SC24-5285
- ▨ Connectivity Planning, Administration, and Operation SC24-5378
- ▨ CMS Shared File System Administration SC24-5367
- ▨ CP System Command Reference SC24-5402

Application Development

- ▨ Application Development Guide for FORTRAN and COBOL SC24-5247
- ▨ Programmer's Guide to the SRPI for VM/SP SC24-5291
- ▨ Application Development Guide for CMS SC24-5286
- ▨ Application Migration Guide for CMS SC24-5366
- ▨ Application Development Reference for CMS SC24-5284
- ▨ Connectivity Programming Guide and Reference SC24-5377
- ▨ GCS Command and Macro Reference SC24-5250
- ▨ **SAA Common Programming Interface Communi-cations Reference** **SC26-4399**
- ▨ Directory of Programming Interfaces for Customers GC24-5417

Index/Glossary

- ▨ Library Guide and Master Index GC19-6207
- Glossary SC24-5379

End Use

- Quick Reference SX20-4400
- ▨ CMS Primer SC24-5236
- CMS Primer for Line-Oriented Terminals SC24-5242
- ▨ CMS User's Guide SC19-6210
- ▨ CMS Command Reference SC19-6209
- ▨ System Product Editor User's Guide SC24-5220
- ▨ System Product Editor Command and Macro Reference SC24-5221
- ▨ System Product Interpreter User's Guide SC24-5238
- ▨ System Product Interpreter Reference SC24-5239
- ▨ EXEC 2 Reference SC24-5219
- ▨ CP General User Command Reference SC19-6211
- ▨ Terminal Reference GC19-6206

▨ one copy of each shaded manual automatically received with product tape

VM/SP RELEASE 6 LIBRARY

Diagnosis

- System Messages and Codes SC19-6204
- System Messages Cross-Reference SC24-5264
- Service Routines Program Logic LY20-0890
- Interactive Problem Control System Guide and Reference SC24-5260
- Diagnosis Guide LY24-5241
- CP Diagnosis Reference LY20-0892
- CP Data Areas and Control Blocks LY24-5220
- CMS Diagnosis Reference LY20-0893
- CMS Data Areas and Control Blocks LY24-5221
- VM CP Trace Table (Poster) SX24-5225
- Problem Determination Summary SX24-5224

Reference Summaries

- SP Editor Command Language Reference Summary SX24-5122
- EXEC 2 Language Reference Summary SX24-5124
- CMS Primer Summary of Commands SX24-5151
- SP Interpreter Reference Summary SX24-5126
- CMS Primer for Line-Oriented Terminals Summary of Commands SX24-5159
- HELP Facility Introduction SX24-5221
- CP General User Command Reference Summary SX24-5219
- CP System Command Reference Summary SX24-5222
- CMS Command Reference Summary SX24-5220
- VM Summary of End Use Tasks and Commands (Poster) SX24-5173

Auxiliary Communication Support

- VTAM Installation and Resource Definition SC23-0111
- VTAM Customization SC23-0112
- VTAM Operation SC23-0113
- VTAM Messages and Codes SC23-0114
- VTAM Programming SC23-0115
- VTAM Diagnosis Guide LY30-5601
- VTAM Programming for LU 6.2 SC30-3400
- VTAM Data Areas (VM) LY30-5593
- VTAM Reference Summary LY30-5600
- VM/Pass-Through Facility Overview GC24-5373
- VM/Pass-Through Facility: Managing and Using SC24-5374
- RSCS Exit Customization SH24-5197
- RSCS General Information GH24-5055
- RSCS Planning and Installation SH24-5057
- RSCS Messages and Codes SH24-5196
- RSCS Operation and Use SH24-5058
- RSCS Diagnosis Reference LY24-5228
- RSCS Reference Summary SX24-5135

Index

Program Number
5664-167

File Number
S370/4300-34

Printed in U.S.A.

VM/SP
Planning Guide and Reference
Order No. SC19-6201-06

READER'S COMMENT FORM

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? __YES __NO

Please print your name, company name, and address:

____

____

____

____

IBM Branch Office serving you: ______________________________

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

SC19-6201-06

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape — Please Do Not Staple — Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

Fold and tape — Please Do Not Staple — Fold and tape

IBM®

IBM®

Program Number
5664-167

File Number
S370/4300-34

2

SC19-6201-06

Printed in U.S.A.