File No. S370-34
Order No. GC20-1801-10

Systems

IBM Virtual Machine Facility/370: Planning and System Generation Guide

| Release 6 PLC 17

This publication is intended for system programmers responsible for the planning, installation, and updating of a VM/370 system. It includes information about:

- Planning for system generation
- Defining your VM/370 system
- Generating VM/370 (CP, CMS, RSCS, and IPCS)
- Generating a 3704/3705 control program that runs under VM/370
- Updating VM/370

A prerequisite for understanding this publication is the publication *IBM Virtual Machine Facility/370: Introduction*, Order No. GC20-1800.

IBM

Eleventh Edition (January 1980)

| This edition (GC20-1801-10), together with Technical Newsletters
| GN25-0776, dated March 3, 1980; and GN25-0837, dated April 1, 1981,
| applies to Release 6 PLC 17 (Program Level Change) of the IBM Virtual Machine Facility/370 and to all subsequent releases unless otherwise indicated in new editions or Technical Newsletters.

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

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

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

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

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

Preface

This publication is intended for system programmers and those responsible for the planning and installation of a VM/370 system. It contains information about VM/370 and the procedures used to generate and support a VM/370 system.

Note: Use this publication in conjunction with the "Memo to Users" (second file on the System Program Update Tape (PUT)) when you are installing a system PUT.

You should have a general understanding of System/370 data processing techniques and be familiar with teleprocessing techniques.

This publication has five parts, plus appendixes.

"Part 1. Planning for System Generation" describes the components, features, and options of VM/370, and tells you what you must do during system generation to support them. Part 1 includes information about CMS, RSCS, and other operating systems in a virtual machine. It also discusses performance options, remote 3270s, the 3704/3705 control program, saved systems, discontiguous saved segments, CMS/DOS, VSAM under CMS, Access Method Services, the Attached Processor System, storage requirements, and minidisks; Part 1 also lists the devices supported by VM/370.

"Part 2. Defining Your VM/370 System" tells you how to create the files that define your system; the real I/O configuration (DMKRIO), CP system control (DMKSYS), VM/370 directory (DMKDIR), system name table (DMKSNT), forms control buffer load (DMKFCB) files, and the macros and control statements needed to create them, are described.

"Part 3. Generating VM/370 (CP, CMS, RSCS, and IPCS)" describes the step-by-step procedure for installing CP, CMS, and, optionally, RSCS and IPCS. The starter systems for CP and CMS which are 2314, 3330, 3340, 3350, and FB-512 are discussed. The Installation Verification Procedure (IVP) for CP and CMS is described. Also included is a description of loading and saving discontiguous saved segments.

"Part 4. Generating the 3704/3705 Control Program" describes the step-by-step procedure for installing a 3704/3705 control program that runs under VM/370.

"Part 5. Updating VM/370" describes the procedures, programs, and EXEC procedures to update VM/370 source code and macro libraries.

The appendixes include information about:

- Program products, language processors and emulators
- Configuring VM/370
- CMS regeneration requirements
- Compatible devices
- Compatibility between VM/370 and CP-67
- VM/370 restrictions
- A sample EXEC procedure to copy DOS/VS macros into a CMS MACLIB

An expanded glossary is available in the IBM Virtual Machine Facility/370: Glossary and Master Index, Order No. GC20-1813.

In this publication, the following terms have extended meanings:

- The term "3330 series" refers to the IBM 3330 Disk Storage Modesl 1, 2, and 11; and the IBM 3333 Disk Storage and Control, Models 1 and 11.
- The term "2305 series" refers to the IBM 2305 Disk Storage, Models 1 and 2.
- The term "3262" refers to the IBM 3262 Printer, Models 1 and 11.
- The term "3289E" refers to the IBM 3289, Model 4 Printer.
- The term "3340 series" refers to the IBM 3340 Disk Storage, Models A2, B1 and B2, and the 3344 Direct Access Storage Model B2.
- The term "3350 series" refers to the IBM 3350 Direct Access Storage Models A2 and B2 in native mode.
- The term "FB-512" refers to the IBM 3310 and 3370 Direct Access Storage Devices.

- The term "3705" refers to the IBM 3705-I and 3705-II Communications Controllers, unless otherwise specified.

- The term "3270" is used in this publication to refer to all VM/370 supported virtual machine display consoles unless otherwise noted. A specific device type is used only when a distinction is required between device types.

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

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

- The term "typewriter terminal" refers to printer-keyboard devices that produce hard-copy output only (such as the IBM 2741 Communication Terminal, the IBM 3215 Console Printer-Keyboard, or the IBM 3767 Communication Terminal, Model 1 or 2, operating as a 2741).

- The term "2741" refers to the IBM 2741 Communication Terminal, and also the 3767 Communication Terminal (unless otherwise noted).

- The term "display device" refers to any VM/370 supported system console terminal that displays data on a screen.

- Unless otherwise noted, where the term "Attention key" is used in this publication, the phrase "(or equivalent)" is implied. The equivalent key on the 1050 terminal is the RESET LINE key; on the 3276, 3277, and 3278 terminal, the Enter key. Each of the terminals that can be used with the VM/370 system has a key that is the equivalent of the Attention key on the 2741 (with which you can signal an attention interrupt).

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

  ```
  set dos on
  ```

  has been previously invoked.

- The phrase "the CMS file system" refers to disk files that are in CMS's 800-byte, 1K, 2K, or 4K fixed physical block format; CMS's VSAM data sets are not included.

- Unless stated otherwise, reference to the System/370 Models 138 and 148 also apply to Models 135-3 and 145-3, respectively.

- The term "3330V" is used in this publication for both volumes and device addresses. When used with volumes, it refers to a Mass Storage System volume that has been mounted and that is directly accessible from the processor. When used with device addresses, 3330V refers to a device on which 3330V volumes may be mounted by the Mass Storage System.

- When an installation has Release 6 installed and an IBM 3850 Mass Storage System attached to the processor, references to 3330 can be thought of as meaning 3330Vs unless the reference is to VM/370 system residence, paging or spooling devices.

COREQUISITE PUBLICATIONS

IBM Virtual Machine Facility/370:

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

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

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

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

System Messages, Order No. GC20-1808

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

Operating Systems in a Virtual Machine, Order No. GC20-1821

Operator's Guide, Order No. GC20-1806

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

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

Release 6 Guide, Order No. GC20-1834

IBM 3270 Information Display System Component Design, Order No. GA27-2749.

Contents

The entries in this Table of Contents are accumulative. They list additions to this publication by the following VM/370 System Contol Program Products:

- VM/370 Basic System Extensions, Program Number 5748-XX8
- VM/370 System Extensions, Program Number 5748-XE1

However, the text within the publication is not accumulative; it only relates to the one SCP program product that is installed on your system. Therefore, there may be topics and references listed in this Table of Contents that are not contained in the body of this publication.

MISCELLANEOUS

Changed: Documentation Only

This Technical Newsletter incorporates minor technical and editorial change.

Summary of Amendments
for GC20-1801-10
as updated by GN25-0776
For Release 6 PLC 9

IBM 3101 DISPLAY TERMINAL

New: Hardware Support

VM/370 now supports the IBM 3101 Display Terminal. For programming systems which support Teletype Model ASR 33/35 teletypewriter, the 3101 can be supported as a substitute device.

3704/3705 CONTROL PROGRAM

Changed: Documentation and Program Support

Technical changes have been made to more adequately reflect the 3704/3705 Control Program support offered by VM/370. The generation procedure for loading the 3704/3705 controllers has also been updated to include the INSTEP EXEC procedure. This procedure is used to generate the 3705 Assembler and create the necessary macro and text libraries needed to generate the 3704/3705 Control Program. For more information see Part Four. "Generating the 3704/3705 Control Program."

SYSTEM PROGRAM UPDATE TAPE

Changed: Program Support

VM/370 service update tapes are now distributed as system Program Update Tapes. The system PUT is distributed regularly by IBM so that you may maintain the current service level. The system generation procedure has been updated to include this new support. Additionally, the VMSERV EXEC procedure is included on the PUT to direct the installation of service. See Part Five. "Updating VM/370" for more information.

MISCELLANEOUS

Changed: Documentation

Various technical and editorial updates have been made throughout this publication to reflect the current level of the VM/370 System Control Program.

Part 1. Planning for System Generation

Part 1 contains planning information. It describes the various components, options, and features of VM/370 and tells you what you must do to install them. Part 1 contains the following sections:

- Introduction
- Performance Guidelines
- Planning Considerations for CMS
- Planning Considerations for CMS VSAM and Access Method Services
- Planning Considerations for CMS/DOS
- Planning Considerations for Virtual Machine Operating Systems (Other than CMS)
- Planning Considerations for 3270s
- Generating a VM/370 System that Supports the 3704/3705
- Generating a VM/370 System that Supports the 3800 Image Library
- 3850 Mass Storage System
- Saved Systems
- Discontiguous Saved Segments
- Attached Processor Systems
- Estimating VM/370 Storage Requirements
- Minidisks
- Configurations

Introduction

The IBM Virtual Machine Facility/370 is a System Control Program (SCP) that manages a real computing system so that all its resources -- main processor, attached processor, storage, and input/output devices -- are available to many users at the same time. Each user has at his disposal the functional equivalent of a real, dedicated computing system. Because this functional equivalent is simulated by VM/370 and does not really exist, it is called a "virtual" machine.

The processors that VM/370 supports are listed under the heading "Processors" later in Part 1. The real System/370 must have the Dynamic Address Translation feature, a hardware facility that translates virtual storage addresses to real storage addresses, and the System Timing facility. Also, it must operate in extended control mode, a mode in which all the features of a System/370, including dynamic address translation, are operational.

VM/370 has four components:

- The control program (CP), which controls the resources of the real computer to provide multiple virtual machines.
- The Conversational Monitor System (CMS), which provides a wide range of conversational and time-sharing facilities. Using CMS, you can create and manage files, and compile, test, and execute problem programs.
- The Remote Spooling Communications Subsystem (RSCS), which transfers spool files between VM/370 users and remote locations over telecommunication lines.
- The Interactive Problem Control System (IPCS), which provides VM/370 problem analysis and management facilities, including problem report creation, problem tracking, and CP abend dump analysis.

For an overview of the functions performed by VM/370, see the _VM/370 Introduction_.

Virtual Machine Operating Systems

While the control program of VM/370 manages the concurrent execution of the virtual machines, it is also necessary to have an operating system managing the work flow within each virtual machine. Because each virtual machine executes independently of other virtual machines, each one may use 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
DOS	VM/370
DOS/VS	Time Sharing Option of OS
OS/PCP	
OS/MFT	
OS/MVT	
OS/VS1	
OS/VS2	**Conversational**
OS-ASP	CMS
RSCS	

CP provides each of these with virtual device support and virtual storage. The operating systems themselves execute as though they were controlling real devices and real storage, but they must not violate any of the restrictions listed in "Appendix F: VM/370 Restrictions."

Introduction to VM/370 System Generation

The purpose of the system generation is to create a system that meets your installation's particular needs.

The first step in the system generation procedure is to restore the starter system, a small working copy of a basic VM/370 system. Using the starter system, you tailor a VM/370 system to your own hardware configuration. You also describe your DASD volumes and define how they are to be used.

The following versions of the starter system can be ordered:

- 2314 Starter System
- 3330 Starter System
- 3340 Starter System
- 3350 Starter System

All starter systems must be restored to a compatible disk (that is, 2314 starter system to a 2314 disk), but all starter systems can then be used to build any supported system residence volume type (2314, 3330, 3340, or 3350).

Before you begin the system generation procedure, you should:

- Know which devices to include in your VM/370 system.
- Create the real I/O configuration (DMKRIO) file describing your I/O configuration. If an IBM Mass Storage System is to be attached to VM/370, you must coordinate the real I/O configuration with the Mass Storage Control's tables.

- Decide how many virtual machines to define.
- Create the VM/370 directory control statement file describing the virtual machines.
- Decide which volumes are to be owned and used by CP (for system residence, paging, spooling, and so on), the amount of real storage available to VM/370, and the user identification of the real system operator.
- Create the CP system control (DMKSYS) file describing CP-owned volumes, the real storage size, and so on.
- If you wish, you can create your own forms control buffer (module DMKFCB) and system name table (module DMKSNT). These modules are, however, supplied with the starter system.

Once you have defined your VM/370 system with these files, you can begin the system generation procedure. You should read the rest of Part 1 to be sure you have all the information you need to generate your system. Part 2 has the information you need to code the files that define your system. Part 3 describes the system generation procedure step-by-step. Before you start the system generation procedure be sure you have the following manuals available:

VM/370 CMS Command and Macro Reference

VM/370 CMS User's Guide

VM/370 CP Command Reference for General Users

VM/370 Operator's Guide

VM/370 System Messages

VM/370 Release 6 Guide

VM/370 System Programmer's Guide

If you are using the MSS support, you will also need the following:

OS/VS Message Library: Mass Storage System Messages, Order No. GC38-1000

OS/VS Mass Storage System (MSS) Installation Planning and Table Create, Order No. GC35-0068

OS/VS1 System Generation Reference, Order No. GC26-3791

or

OS/VS2 System Programming Library: System Generation Reference, Order No. GC26-3792

During the system generation procedure, you apply the system Program Update Tape (PUT) supplied with the starter system. This updates your system to the current level. Then use the Installation Verification Procedure (IVP) to verify that the VM/370 system is functioning properly.

| Note: If you are installing a system program update tape, use this
| publication in conjunction with the Memo to Users supplied on the PUT.

If you wish to install the Remote Spooling Communications Subsystem (RSCS) do so after running the IVP. After you generate VM/370 (CP, CMS, IPCS, and, optionally, RSCS), you can generate the 3704/3705 Control Program. Information about generating this program is found in Part 4 of this manual.

Performance Guidelines

The performance characteristics of an operating system when it is run in a virtual machine environment are difficult to predict. This unpredictability is a result of many factors:

- The System/370 model used
- The system environment used (uniprocessor or attached processor)
- The total number of virtual machines executing
- The type of work being done by each virtual machine
- The speed, capacity, and number of the paging devices
- The amount of real storage available
- The degree of channel and control unit contention, as well as arm contention, affecting the paging device
- The type and number of VM/370 performance options in use by one or more virtual machines
- The availability of VM/370 hardware assist
- The 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 being accessed by the operating system and shared with other systems depends on the total MSS utilization and contention.

Performance Measurement and Analysis

The VM/370 control program has two commands that measure system performance and, thus, help you identify problem areas. The MONITOR command collects system measurement data offline for the system operator or system analyst, while the INDICATE command displays system measurement data online for the system operator, system analyst or general user.

The MONITOR command controls the collection of performance data and writing it to system spool files or tapes. Both summary and trace data can be collected. The classes of data collected may be specified using either the operands of the MONITOR command or those of the SYSMON macro instruction. The classes selected depend on the nature of the analysis to be performed. The IBM Field Developed Program (FDP) VM/370: Performance/Monitor Analysis Program can be used to reduce the data collected. The guidelines for using this program provide the user with information that will aid him in determining the overall load environment and performance profile of his system. The VM/370 Performance/Monitor Analysis Program should enable him to analyze the utilization of and contention for the major system resources such as the CPU, storage, and I/O paging subsystems.

The INDICATE command displays, at the terminal, some key information about the system that shows the current performance indicators. Invoking the INDICATE command displays the system conditions existing at the time the command is issued including attached processor utilization measurement when operating in an attached processor environment. If, after using the INDICATE command, the system analyst wants more extensive data collection and reduction, he can use the MONITOR command.

The user can specify automatic data collection with the SYSMON macro in DMKSYS. Coding Considerations are contained in the section "Preparing the CP System Control File (DMKSYS)." See the _VM/370 System Programmer's Guide_ for the directions on using the MONITOR command to collect performance data on a dedicated tape drive or spool file, the format and contents of the various classes of data collection available with MONITOR, and details of the INDICATE command options.

Using the Performance Options

The performance of a specific virtual machine can be improved by assigning it one or more performance options. These include: favored execution, priority, reserved page frames, locked pages, and virtual=real.

The performance of a VM/370 system running virtual storage operating systems can be improved if you use virtual machine assist or Extended Control-Program Support. The manner in which these are supported by the various VM/370 processors is detailed below:

Virtual Machine Assist				VM/370:ECPS		
Standard Feature	Special Feature	RPQ	Not Available	Standard Feature	Special Feature	Not Available
135-3	135	168	155	135-3	4331[1,2]	135
138	145	168-3	155II	138		145
145-3	158	168AP	165	145-3		155
148	158-3	168MP	165-3	148		155II
3031	158AP	3032		3031[2]		158
3031AP	158MP	3033UP		3031AP[2]		158-3
4341[1]	4331[1]	3033AP		4341		158AP
		3033MP				158MP
						165
						165-3
						168AP
						168MP
						3032
						3033
						3033AP
						3033MP

[1]To function with the availability of the hardware feature.

[2]See the _VM/370 System Programmer's Guide_ for specific VM/370: EPCS functions available on this processor.

In order to invoke the favored execution, priority, reserved page frames, and locked pages options you must have a virtual machine defined with the appropriate command privilege classes. Usually, the operator's virtual machine has the appropriate command classes. Additional planning is needed to support the virtual=real option and virtual machine assist as well as VM/370:ECPS. All of these performance options are described in detail in the _VM/370 System Programmer's Guide_.

Specifying a Virtual=Real Machine

Although the virtual=real option eliminates paging, 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 highest page of the virtual=real area. In order for the virtual machine to perform input/output into virtual page 0, the CCW addresses must be translated.

When CP loads an operating system into a virtual=real area, it turns on CCW translation. Once the operating system is loaded, 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. Most operating systems can be generated so that they do not use this area for input/output. However, violation of this restriction may cause damage to the entire VM/370 system.

The size of the virtual=real area is specified during CP system generation. It must be large enough to contain the entire address space of the largest virtual machine that you execute in the virtual=real area.

Only one virtual=real area can be defined.

Only one virtual machine at a time can occupy the virtual=real area.

Since 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/370 initial program load (IPL). It can be released by the primary system operator to be used as part of the dynamic paging area. Once released, it cannot be reclaimed except by reloading VM/370. The virtual=real area must be released in total, that is, unused pages of the area cannot be selected for release.

Note: If a very large virtual=real area is released after VM/370 initialization, a system performance degradation may occur as more and more users log on and use the released space. The reason for this is that the number of pages allocated for CP fixed free storage during VM/370 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, and system overhead may increase as CP extends to get dynamic free storage pages.

This problem may be counteracted by using the FREE operand in the SYSCOR macro instruction in the system control (DMKSYS) file at system generation. The SYSCOR macro is described in "Part 2. Defining Your VM/370 System." The examples used in the following discussions assume that you are allowing VM/370 to determine the number of free storage pages to allocate.

To use the virtual=real option effectively on a multiport teleprocessing system with no CCW translation (SET NOTRANS ON), lines must be dedicated to that system via the ATTACH command or by VM/370 directory assignment. Conversely, on a multiport 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 issue the DIAL command while SET NOTRANS ON is in effect, CCW translation is done for I/O involving that line.

Note that you cannot execute programs with dynamic or self-modifying channel programs in a virtual=real area if you also use the DIAL command. Also, you cannot load (via IPL) a shared system into a virtual machine running in the virtual=real area. For a virtual=real machine, you must issue 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, do the following:

- Specify the VIRT=REAL option in the VM/370 directory for all the virtual machines in your installation that you plan to run in the virtual=real area.
- Reserve enough DASD space for the CP nucleus. A CP nucleus that supports a virtual=real area is larger than one that does not.
- Make sure the virtual machine you are using to generate CP has sufficient virtual storage.
- Specify the amount of storage you want reserved for a virtual=real area.

"Part 2. Defining Your VM/370 System" describes the Directory program, including information about the VIRT=REAL operand of the OPTION control statement.

RESERVING DASD SPACE FOR A CP NUCLEUS WITH A VIRTUAL=REAL AREA

A CP nucleus with the virtual=real option requires more DASD space for system residence than a CP nucleus without the option. Use the following formulas to calculate the number of cylinders needed for system residence (disregard any remainders):

```
Number of 2314/2319 cylinders = (128+(VRSIZE/4))/32
Number of 3330/3333 cylinders = (171+(VRSIZE/4))/57
Number of 2305/3340 cylinders = (144+(VRSIZE/4))/24
Number of 3350 cylinders      = (240+(VRSIZE/4))/120
```

where VRSIZE is the size of the virtual=real storage area (in K bytes). K represents 1024 bytes. This size must be at least 32K bytes.

The number of cylinders you calculate here is the number you reserve on your system residence device using the FORMAT program DMKFMT. You need this information to code the SYSRES macro of your CP system control (DMKSYS) file correctly.

If you do not reserve enough DASD space for your CP nucleus, the nucleus overlays other cylinders when it is written on the system residence volume and may itself be overlaid by other disk writes.

VIRTUAL STORAGE REQUIREMENTS

When you are generating VM/370 you have three constraints 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:

- CMS (320K)
- CP nucleus size (including the virtual=real area)

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 your VM/370 system to include a virtual=real machine, in Step 16 of 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):
```

Normally, you would not want to specify the largest virtual=real machine possible, since 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 real CPU 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 the amount of available real storage (ARS) by subtracting the amount of storage required by CP from the real machine size. Formula 1 is:

$$ARS = RM - \left[I + T + 12K + 4K \left[\frac{RM - 256K}{64K} \right] \right]$$

where:

RM is the real machine size.

I is the amount of 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 amount of 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:

$$4K \left[\frac{RM}{256K} \right]$$

If the calculation RM/256K results in a fraction, the result should be rounded upward to the next higher integer.

12K is the amount of fixed free storage allocated for the first 256K of real storage.

$$4K \left[\frac{RM - 256K}{64K} \right]$$

is the amount of 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 amount of 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 amount of real storage required by CP and subtracting that value from the real machine size. When you calculate the amount of 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 (the 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 amount of storage needed at IPL time for the dynamic paging area. After VM/370 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
384K	104K
512K	232K
768K	484K
1M	732K
2M	1744K

Note that in this table it is assumed the value of I is equivalent to 244K[1].

---

[1] Since the amount of storage required to IPL VM/370 varies with the inclusion of optional features and the number of devices in DMKRIO, this figure is used in the following example for illustrative purposes only.

Example 1

Determine the maximum size of the virtual=real area for a real machine with 768K of storage that executes a VM/370 system that requires 244K to IPL.

Formula 1

$$ARS = 768K - \left[244K + 4K\left[\frac{768K}{256K}\right] + 12K + 4K\left[\frac{768K - 256K}{64K}\right]\right]$$

$$ARS = 768K - [244K + 12K + 12K + 32K]$$

$$ARS = 768K - 300K$$

$$ARS = 468K$$

Formula 2

$$VRS = 768K - \left[244K + 12K + 12K + 4K\left[\frac{768K - 256K - 468K}{64K}\right] + 16K\right]$$

$$VRS = 768K - \left[268K + 4K\left[\frac{44K}{64K}\right] + 16K\right]$$

$$VRS = 768K - [268K + 4K[0] + 16K]$$

$$VRS = 484K$$

Note that the fraction (44/64) resulting from the

$$\frac{RM - 256K - ARS}{64K}$$

calculation in Formula 2 is truncated to zero.

Example 2

Determine the maximum size of the virtual=real area for a real machine with 384K of real storage. The VM/370 system requires 244K to IPL.

Formula 1

$$ARS = 384K - \left[244K + 4K\left[\frac{384K}{256K}\right] + 12K + 4K\left[\frac{384K - 256K}{64K}\right]\right]$$

$$ARS = 384K - [244K + 4K[2] + 12K + 8K]$$

$$ARS = 384K - [272K]$$

$$ARS = 112K$$

Note that the fraction 384/256 in the trace table calculation is rounded to the next higher integer, two.

Formula 2

$$VRS = 384K - \left[244K + 8K + 12K + 4K\left[\frac{384K - 256K - 112K}{64K}\right] + 16K\right]$$

$$VRS = 384K - \left[264K + 4K\left[\frac{16}{64}\right] + 16K\right]$$

$$VRS = 384K - [280K]$$

$$VRS = 104K$$

Note that the calculation

$$\frac{RM - 256K - ARS}{64K}$$

results in a fraction that is then replaced by zero.

Example 3

Determine the maximum size virtual=real area for a real machine with 1792K of real storage. The VM/370 system requires 244K to IPL.

Formula 1

$$ARS = 1792K - \left[244K + 4K\left[\frac{1792K}{256K}\right] + 12K + 4K\left[\frac{1792K - 256K}{64K}\right]\right]$$

$$ARS = 1792K - [244K + 4K[7] + 12K + 4K[24]]$$

$$ARS = 1792K - [244K + 28K + 12K + 96K]$$

$$ARS = 1792K - [380K]$$

$$ARS = 1412K$$

Formula 2

$$VRS = 1792K - \left[244K + 28K + 12K + 4K\left[\frac{1792K - 256K - 1412K}{64K}\right] + 16K\right]$$

$$VRS = 1792K - \left[284K + 4K\left[\frac{124}{64}\right] + 16K\right]$$

$$VRS = 1792K - [308K]$$

$$VRS = 1484K$$

Note that the fraction (124/64) resulting from the

$$\frac{RM - 256K - ARS}{64K}$$

calculation in Formula 2 is rounded to the next higher integer, two.

Example 4

Determine the maximum size virtual=real area for a real machine with 256K of real storage. The VM/370 system requires 244K to IPL.

$$ARS = 256K - \left[244K + 4K\left[\frac{256K}{256K}\right] + 12K + 4K\left[\frac{256K-256K}{64K}\right]\right]$$

$$ARS = 256K - [244K + 4K[1] + 12K + 4K[0]]$$

$$ARS = 256K - [260K]$$

$$ARS = -4K$$

Since ARS is a negative number, CP cannot IPL and the following error message informs the user of this condition:

DMKCPI955W INSUFFICIENT STORAGE FOR VM/370

Virtual Machine Assist

Virtual machine assist is a combination of a processor feature and VM/370 programming. It improves the performance of VM/370. Virtual storage operating systems that run in problem state under the control of VM/370 use many privileged instructions and SVCs that cause interrupts which VM/370 must handle. When virtual machine assist is used, many of these interrupts are intercepted and handled by the processor; and, consequently, VM/370 performance is improved. The manner in which virtual machine assist and ECPS are supported by the various VM/370 processors is detailed under "Using the Performance Options."

Certain interrupts must be handled by VM/370. Consequently, virtual machine assist is not available if it:

- Has an instruction address stop set
- Traces SVC and program interrupts

Since an address stop is recognized by an SVC interrupt, VM/370 must handle SVC interrupts while address stops are set. Whenever you issue the ADSTOP command, VM/370 turns off the SVC handling portion of the assist feature for your virtual machine. The assist feature is turned on again after the instruction is encountered and the address stop removed.

Whenever a virtual machine issues a TRACE command with the SVC, PRIV, BRANCH, INSTRUCT, or ALL operands, the virtual machine assist feature is turned off for that virtual machine. The assist feature is turned on again when the tracing is completed.

If virtual machine assist is available on a processor, the operator can turn the function off, and on again, for the entire VM/370 system. Also, if the function is available to VM/370, each virtual machine operator can turn the function off, and on again, for his own virtual machine. When you create your VM/370 directory, you can set off the SVC-handling portion of the virtual machine assist function for various virtual machines by specifying SVCOFF on the OPTION control statement.

VM/370 EXTENDED CONTROL-PROGRAM SUPPORT

VM/370 Extended Control-Program Support is a hardware assist function that provides support over and above that provided by the virtual machine assist feature described previously, and consequently reduces VM/370's real supervisor state time needed to support virtual machines. VM/370 Extended Control-Program Support provides the following functions.

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

Whenever VM/370 is loaded on one of the supported processors, all three hardware assist functions plus virtual machine assist are activated unless turned off by the system operator.

Expanded virtual machine assist includes a more comprehensive 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

If (1) CP assist is turned off, (2) hardware assist does not support the specific service required, or (3) an error condition occurs, the appropriate CP software routine is used.

Virtual interval timer assist provides for hardware updating of the location 80 interval timer for each virtual machine that has the virtual timer assist function turned on. This timer assist provides a more accurate and repeatable interval timer value for virtual machines than was previously possible through CP software.

Both virtual machine assist and expanded virtual machine assist are automatically turned off if the user invokes certain TRACE functions. In addition, virtual interval timer assist is turned off if external interrupts are traced. When the tracing function is terminated, CP automatically reactivates these VM/370 hardware assist functions.

For more details on VM/370 Extended Control-Program Support, refer to the _VM/370 System Programmer's Guide_.

Planning Considerations for CMS

The Conversational Monitor System (CMS) is a component of VM/370 that provides a comprehensive set of conversational facilities to 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.

CMS is a required component of VM/370. You must generate CMS in order to support CP.

This section contains the following information about CMS:

- Storage Requirements
- Device Support
- Libraries
- Command Language
- Program Language Facilities
- Limited Support of DOS and OS
- Disk and File Management
- Tape Support
- Unit Record Support
- Editing
- Batch Facility
- Saving CMS

CMS Storage Requirements

CMS requires virtual storage and auxiliary storage. A minimum of 320K bytes of virtual storage is required for a CMS virtual machine; this virtual storage is distributed as follows:

- CMS nucleus -- 128K
- Loader tables -- 8K (for virtual machines with up to 384K of virtual storage)
  12K (for virtual machines with more than 384K of virtual storage)
- User program area -- 184K (for application programs or CMS disk-resident commands)

AUXILIARY STORAGE

The CMS auxiliary storage requirements are:

- System residence for CMS -- 110 cylinders on a 2314 or 2319, 72 cylinders on a 3330 or 3333, 203 cylinders on a 3340 Model 35 or Model 70, or 29 cylinders on a 3350 in native mode.

- Resident disk space for application programs (CMS commands, user programs, IBM Program Products) -- the amount of space needed is program-dependent, and must be assigned by you.
- Work space for application programs (CMS commands, user programs, IBM Program Products) -- the amount of space is program-dependent, and must be assigned by you.

Device Support

CMS supports the virtual machine devices shown in Figure 2.

Virtual IBM Device	Virtual Address[1]	Symbolic Name	Device Type
3210,3215,1052	cuu[2]	CON1	System console
2314,2319,3330,3340,3350	190	DSK0	System disk (read-only)
2314,2319,3330,3340,3350	191[3]	DSK1	Primary disk (user files)
2314,2319,3330,3340,3350	cuu	DSK2	Disk (user files)
2314,2319,3330,3340,3350	cuu	DSK3	Disk (user files)
2314,2319,3330,3340,3350	192[3]	DSK4	Disk (user files)
2314,2319,3330,3340,3350	cuu	DSK5	Disk (user files)
2314,2319,3330,3340,3350	cuu	DSK6	Disk (user files)
2314,2319,3330,3340,3350	cuu	DSK7	Disk (user files)
2314,2319,3330,3340,3350	19E[3]	DSK8	Disk (user files)
2314,2319,3330,3340,3350	cuu	DSK9	Disk (user files)
1403,3203,3211,1443	00E	PRN1	Line printer
2540,2501,3505	00C	RDR1	Card reader
2540,3525	00D	PCH1	Card punch
2401,2402,2403,2415, 2420,3410,3411,3420	181-4	TAP1-TAP4	Tape drives

[1]The device addresses shown are those that are preassembled into the CMS resident device table. You can change the virtual machine addresses by using the CP DEFINE command.
[2]The virtual address of the system console may be any valid multiplexer address.
[3]The virtual device address (cuu) of a disk for user files can be any valid System/370 device address, and can be specified by the CMS user when he activates a disk. If the user does not activate a disk immediately after loading CMS, CMS automatically activates the user's primary disk (A-disk) at virtual address 191, the D-disk at 192, and the Y-disk (a read-only extension of the system disk) at 19E.

Figure 2. Devices Supported by a CMS Virtual Machine

Under CP, unit record devices and the system console are simulated and mapped to different addresses and different devices. For instance, CMS expects a 3215, 3210, or 1052 type of operator's console, but many terminals are 2741s or 3270s. Regardless of the real device type, the virtual system console is a 3215. The control program (CP) of VM/370 handles all channel program modifications necessary for this simulation. CMS virtual disk addresses are mapped by CP to different real device addresses.

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

The A-disk is a read/write disk and is the primary or first disk. Files that you wish to retain for later use are stored on one of your disks. Information stored on a disk remains there until you erase it. An exception is the temporary disk; files written on this disk are lost when you log off. In addition to the system disk (S-disk) and primary disk (A-disk), each CMS user can have up to eight additional disks.

You can enter CMS commands and input files from the terminal and direct output files, program results, and error and prompting messages back to the terminal.

The virtual card reader is used as the input medium for files, source decks, and data to be processed by your programs. The virtual card punch is used for your output files, language processor object decks, and various other types of data. The virtual printer is used for program results, storage dumps and language processor output.

Under VM/370, the unit record equipment is normally spooled. CMS supports only spooled unit record devices.

The following is an example of a VM/370 directory entry for a CMS virtual machine.

```
USER USER1 PASSWORD
 ACCOUNT NUMBER BIN7
  IPL CMS
  CONSOLE 009 3215
  SPOOL C 2540 READER A
  SPOOL D 2540 PUNCH A
  SPOOL E 1403 A
  LINK MAINT 190 190 RR
  MDISK 191 2314 71 10 UDISK1 W RPASS WPASS
```

This entry describes the configuration when you log on as USER1. The Directory program control statements are described in Part 2. Briefly, this entry describes the USER1 virtual machine: it has a console at 009, a class A reader at 00C, a class A punch at 00D, a class A printer at 00E, a link to the CMS system disk (owned by userid MAINT) at 190, and a minidisk at 191. Once you are logged on you can change the configuration and also the spooling classes of the unit record devices. You can add devices to the VM/370 directory or dynamically add to the configuration as needed.

CMS Libraries

CMS updates simulated partitioned data sets which contain:

- CMS and OS macros to be used at assembly time (macro libraries)
- Object routines to be referred to at execution-load time (text libraries)

The system macro libraries, located on the CMS system disk, are:

<u>Library</u>	<u>Contents</u>
CMSLIB MACLIB	All of the CMS macros
OSMACRO MACLIB	The selected OS macros from SYS1.MACLIB that are supported under CMS
OSMACRO1 MACLIB	The remaining distributed OS macros from SYS1.MACLIB
TSOMAC MACLIB	The OS macros distributed in SYS1.TSOMAC
DOSMACRO MACLIB	The DOS/VS macros and CMS macros that provide DOS/VS function

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 DOS programs containing DOS macros in CMS/DOS, you must first create a CMS macro library that contains all the DOS macros you need. "Appendix G: A Sample EXEC Procedure for Copying DOS/VS Macros into a CMS MACLIB" shows the procedure for copying an entire macro library. The procedure for copying individual macros is described in the <u>VM/370 CMS User's Guide</u>.

The system text libraries, also located on the CMS system disk, are:

<u>Library</u>	<u>Contents</u>
CMSLIB TXTLIB	The CMS system text library
TSOLIB TXTLIB	Selected TSO routines necessary to support certain features of the language program products
EREPLIB TXTLIB	Base text library for CPEREP
ERPTFLIB TXTLIB	Updates to CPEREP text library

Execution-time libraries are available with the program product language processors and execute under CMS.

You can generate your own libraries and add, delete, or list entries in them via the MACLIB and TXTLIB commands. You can also specify which libraries (system and user) to use for program compilation and execution via the GLOBAL command. Up to eight libraries may be specified. Although CMS library files are similar in function to OS partitioned data sets, OS macros should not be used to update them.

CMS Command Language

The CMS command language lets you converse with CMS. With this command language, you can use:

- Language compilers
- An assembler
- CMS file management system
- Context editing and line editing
- Execution control
- Debugging capability

Additionally, you can invoke the CP commands available to all virtual machines under VM/370 directly from CMS. Using these CP commands, you can send messages to the operator or to other users, dynamically change your virtual machine's configuration, and invoke spooling facilities. In CMS, the facilities of CP and CMS together appear as those of a single integrated system.

To use CMS, you must first gain access to a virtual machine via the CP LOGON command, and IPL CMS. Then you can enter commands or data from the remote terminal (virtual operator's console). Each command, upon completion, returns control to you. For information about how to use CMS and for a description of all CMS commands, see the *VM/370 CMS Command and Macro Reference* and the *VM/370 CMS User's Guide*.

CMS Program Language Facilities

The languages available under CMS include:

S/370 Assembler
VS BASIC
PL/I
OS FORTRAN IV
OS/VS COBOL
DOS PL/I Optimizer
DOS/VS COBOL
VS APL

The assembler is distributed with VM/370. The language compilers that are program products must be ordered separately. For a complete list of language processors that can be executed under CMS, see "Appendix A: Program Products, Installed User Programs, Field Developed Programs and Emulators."

CMS executes the compilers via interface modules. CMS commands are provided to invoke the compilers within the conversational environment of CMS.

OS/VS COBOL programs, using the following facilities, can be compiled under CMS, but must be transferred to a machine (virtual or real) running OS for execution.

QSAM & BDAM spanned records
ISAM
RERUN statement
label-handling options
OPEN REVERSED
Sort feature
Segmentation feature
ASCII code feature
Forced end of volume
TCAM feature

OS PL/I programs, using the following facilities, can be compiled under CMS, but must be transferred to a machine (virtual or real) running OS for execution.

Multitasking
Teleprocessing file support
ISAM
Backwards attribute

Spanned records for buffered files
Sort-merge
Checkpoint-restart
ASCII data sets
Track overflow

The DOS/VS COBOL and DOS PL/I Optimizing compilers execute in the CMS/DOS environment of CMS. The CMS/DOS environment does not support the execution of DOS programs that use:

- Sort exits. The DOS/VS COBOL and DOS PL/I Optimizer SORT verbs are not supported in CMS/DOS.
- Teleprocessing, indexed sequential access method (ISAM), or direct access method (DAM). CMS/DOS supports only the sequential access method (SAM) and virtual storage access method (VSAM).
- Multitasking. CMS/DOS supports only a single partition, the background partition.

CMS TEXT PROCESSING FACILITY

Text processing facilities that can create formatted output from one or more CMS files containing text and/or control words are available through the SCRIPT command. SCRIPT/370 is an IBM Installed User Program that must be ordered separately.

Limited Support of OS and DOS in CMS

Object programs (TEXT files) produced under CMS and under OS in real or in virtual machines can be executed under CMS if they do not utilize 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 be used to process data sets on OS disks.

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

CMS supports VSAM and access method services for DOS and OS users. CMS supports VSAM for the following compilers: OS/VS COBOL, OS PL/I, VS BASIC, DOS/VS COBOL, and DOS PL/I. CMS does not support VSAM for assembler language programs or VS APL.

The application programmer who normally uses CMS to interactively create, modify, and test his programs may require facilities not supported in CMS (for example, an OS program using ISAM). He can alternately execute CMS and another operating system in the same virtual machine.

A description of the actual processes for reading OS or DOS files is in the _VM/370 CMS User's Guide_. A description of alternating operating systems is in _VM/370 Operating Systems in a Virtual Machine_.

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, or Assembler language.

You can also execute any data base description generation and program specification block generation. The data base recovery and reorganization utilities must also be executed in a DOS/VS virtual machine.

For more information, see the VM/370 CMS User's Guide, and DL/I DOS/VS General Information, GH20-1246.

CMS Disk and File Management

CMS can manage up to ten virtual disks for each user. These disks may be minidisks or full packs. Moreover, they may be in:

- CMS format
- OS or DOS format
- VSAM format

When the VM/370 MSS support is installed, and the VM/370 processor is attached to an MSS, any CMS virtual disk can be located on an MSS 3330V volume.

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 and DOS disks or minidisks may be used in CMS. OS or DOS programs executing in CMS may read data sets or files on OS or DOS disks, but may not write or update them. OS and DOS minidisks may be formatted with the IBCDASDI service program, or with an appropriate OS/VS or DOS/VS disk initialization program, if the disk is a full pack.

VSAM disks used in CMS are fully compatible with OS and DOS VSAM disks. Minidisks for use with VSAM must be formatted with the IBCDASDI program; full disks must be initialized using the appropriate OS/VS or DOS/VS disk initialization program.

DISK ACCESS

Disks can be accessed in two ways: read-only, where files on that disk can only be read; and read/write, where files can be read and written.

Both CP and CMS can control read/write access. If a disk is designated read/write by CP, then the CMS access determines its read/write status. If a disk is designated read-only by CP, then it can only be accessed read-only in 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/370 directory entry, or it can be acquired dynamically with the CP LINK or DEFINE commands.
- Identify the disk to CMS by assigning it a filemode letter. You do this using the ACCESS command in CMS.

While you may have many virtual disks known to CP in your virtual machine configuration at one time, CMS allows a maximum of ten to be accessed, with filemode letters A through G, S, Y, and Z. The S-disk (usually at virtual address 190) is the CMS system disk. The A-disk (usually at virtual address 191) is the user's primary read/write work disk. Disks may be dynamically accessed and released during a terminal session.

FILE SHARING

CP provides for sharing of disks and minidisks among several users. The type of access (multiple users read-only or read/write) is controlled by LINK command operands. Password protection is provided. Since CMS does not provide any control for multiple writes (such as ENQ, DEQ), it is not recommended that CMS disks be used with multiple-write access.

CMS DISK FILE FORMAT

All CMS disks (that is, disks that are to contain CMS files) must be formatted before being used the first time. The CMS FORMAT command initializes disks in CMS format and writes a label on the disk. The 10-byte label (written on record 3 of cylinder 0, track 0) consists of the following:

- Four characters: CMS=
- Six characters: Desired label (blank-filled if less than 6 characters; truncated if more than 6 characters)
- The remaining bytes of the record are all binary zeros

The disks are formatted into 800-byte physical records, called blocks. Logical records, which may be fixed-length or variable-length, are imposed on constant physical blocks. Space required for files is automatically allocated by CMS. As a file grows, its space is expanded, and it is contracted as its space requirements are reduced.

Files on a CMS disk are identified by means of a file directory, called the master file directory. The file directory is updated when a command is issued that changes the status of the file on the disk.

Figure 3 compares the disk devices supported by CMS.

For more information about planning CMS minidisk requirements, see "Estimating VM/370 Storage Requirements" later in this section.

	2314/ 2319	3330	3340	3350
Maximum number of files that can be contained on the disk	3500	3400	3400	3400
Maximum minidisk size (in cylinders)	203	246	348 (model 35) 682 (model 70)	115
Number of 800-byte blocks per cylinder	150	266	96	570
Maximum data extent	65,535 records 12,848,000 bytes			

Figure 3. CMS Disk File Statistics

IDENTIFYING DISK FILES

CMS commands are provided to list the identifications of files on CMS and non-CMS formatted disks and minidisks. The LISTFILE command lists 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 for listing data spaces on VSAM volumes.

CMS Tape Support

Each CMS machine can support up to four magnetic tape units at virtual addresses 181, 182, 183, and 184. They may be 2401, 2402, 2403, 2415, 2420, 3410/3411, or 3420 drives, or a mixture of tape drives.

Three tape-handling commands (ASSGN, FILEDEF, and TAPE) allow you to specify the modeset of the tape: track (7-track or 9-track), density, and, for 7-track tape only, 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 (bits per inch), odd parity, converter on, and translate off. If the tape is 9-track, the density is assumed to be 1600 bpi (or whatever bpi the tape drive was last set at) for dual density drives; for single density drives, whatever bpi the drive is (800, 1600, or 6250 bpi) is assumed.

As an alternative to specifying mode in each command that uses the tape (for example, FILEDEF), you can issue a CMS TAPE command that sets the mode for the tape and stays in effect until reissued. You must do this if one of your programs is to use tapes in other than the default mode.

With one exception, CMS commands permit only unlabeled tapes to be read or written. However, the CMS TAPPDS command can read standard OS tape labels. Your programs executing under CMS must use unlabeled tapes or provide code to create and read their own labels as data records.

Multivolume tape files are not supported by CMS.

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

The VM/370 operator must attach tapes to your CMS virtual machine before any tape operation can take place.

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

CMS Unit Record Support

CMS supports one virtual card reader at virtual address 00C, one virtual card punch at virtual address 00D, and one virtual printer at virtual address 00E. Under VM/370, these devices are spooled. CMS does not support real or dedicated unit record devices, nor does it support a virtual 2520 Card Punch. Figure 2 lists the devices supported as virtual devices by CMS.

CARD READER

The READCARD command reads data records from the spooled card reader to a CMS disk. Input records of 151 or fewer characters are accepted. Column binary data is not acceptable. Your card decks must be read into the virtual reader before a READCARD command can be issued. Do one of the following:

- Place a card deck, containing only one file, in a real card reader and have CP read it. The card images are placed on a spool file in the specified virtual machine's virtual card reader. If you are not logged on when data is spooled to your virtual card reader, the deck remains in your virtual card reader until you log on and issue the READCARD command.

- Transfer records from your virtual card punch or printer to a virtual card reader (your own or that of another virtual machine).

For more information on reading files from the CMS card reader, see the *VM/370 CMS User's Guide*.

CARD PUNCH

The CMS PUNCH command causes the specified file to be punched to the spooled card punch. Records up to 80 characters long are accepted. Shorter records are padded to 80 characters with blanks filled in to the right. The following are not supported:

- Punch stacker select
- Punch feed read
- 3525 Multiline Card Print feature

PRINTER

The PRINT command prints the specified disk file on the spooled printer. You can specify whether the first character of each record is to be interpreted as a carriage control character or as data. Both ASA and machine code carriage control characters are supported. The file may have either fixed- or variable-length records.

Editing

Using the CMS Editor, you can create new files online or modify or display portions of existing files.

The CMS Editor operates on fixed- and variable-length records. The maximum record length accepted by the editor is 160 characters. You can specify file characteristics, such as record length, format, and tab locations. The system includes defaults for certain filetypes (such as ASSEMBLE and EXEC).

Files are edited in virtual storage. The maximum size file that can be edited is determined by the amount of virtual storage available to you, minus that used by the CMS nucleus.

If the file does not fit into virtual storage, it must be divided into smaller files, and each file edited separately. Sufficient disk space must be available to accommodate both parts of the file. Alternately, you can temporarily increase the size of your virtual storage by issuing the CP DEFINE STORAGE command.

For more information about the editing facilities of CMS, see the VM/370 CMS User's Guide.

CMS Batch Facility

The CMS Batch Facility is a VM/370 programming facility that executes under CMS. It allows you to execute jobs in batch mode. You can submit jobs from a virtual machine or from a real card reader. You do not have to be logged on to submit jobs to the batch virtual machine. If you want to use the batch facility you must define a CMS batch virtual machine when you create your VM/370 directory. You should include a read/write disk at virtual address 195 in the directory entry. Information about the CMS Batch Facility is in the VM/370 CMS User's Guide.

There must be an entry in the VM/370 directory for any user who wants to submit jobs to the CMS Batch Facility. This entry can be the minimum: userid, password, and one device. Alternatively, you can provide entries for users who will not be logging on to the system, but submitting jobs through the real card reader. For these users, you can code the password in the directory as NOLOG; these users do not need any devices defined for their virtual machines. The CMS Batch Facility uses the FOR operand of the CP SPOOL command to identify their output files.

The batch facility provides two entry points so you can add support that your installation may require. BATEXIT1 is provided so you can write your own routine to check non-CMS control statements. BATEXIT2 is provided so you can process additional information on the /JOB card. These entry points are described in the VM/370 System Programmer's Guide.

You can write EXEC procedures to control the operation of a batch facility virtual machine. See the _VM/370 CMS User's Guide_ for examples of these EXEC procedures.

Saving CMS

CMS is designed so that it can be saved easily and so that the second segment of CMS can be shared by CMS users. Also, CMS is designed so that CMS/DOS, CMS VSAM and access method services, the CMS Editor, CMS EXEC processor, and CMS OS simulation routines can be placed in discontiguous saved segments. The VM/370 starter system has entries in the system name table (DMKSNT) and CP system control file (DMKSYS) so that you can save CMS (and the CMS discontiguous saved segments) at the end of the system generation procedure for CMS.

For more information about saved systems, see the "Saved Systems" section of this manual and see the _VM/370 System Programmer's Guide_.

Planning Considerations for CMS VSAM and Access Method Services

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

CMS also supports access method services to manipulate OS and DOS VSAM and SAM data sets.

Under CMS, VSAM data sets can span up to nine DASD volumes. CMS does not support VSAM data set sharing; however, CMS already supports the sharing of minidisks or full pack minidisks. Only one user may have write access to the VSAM master catalog, but many other users may read and reference the catalog at the same time.

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. A VSAM data set created in CMS has a file format that is exactly the same as, and therefore compatible with, OS and DOS VSAM data sets. Thus a VSAM data set created in CMS can later be read or updated by OS or DOS.

Because VSAM data sets in CMS are not a part of the CMS file system, CMS file size, record length, and minidisk size restrictions do not apply. The VSAM data sets are manipulated with access method services programs executed under CMS, instead of with the CMS file system commands. Also, all VSAM minidisks and full packs used in CMS must be initialized with the IBCDASDI program or an appropriate DOS/VS 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 2314/2319 devices, or for 3340 devices that do not have the feature.

Hardware Devices Supported

Because CMS support of VSAM data sets is based on DOS/VS VSAM and DOS/VS access method services, only disks supported by DOS/VS can be used for VSAM data sets in CMS or for CMS disk files used as input by access method services. These disks are:

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

- IBM 3350 Direct Access Storage
- When VM/370 MSS support is installed, and the VM/370 processor is attached to an MSS, the CMS disk may be defined as a 3330 Model 1 which is mapped by VM/370 to all or part of a 3330V volume.

Programming Languages Supported

CMS supports VSAM for programs written for the following compilers:

Compiler	Program No.
OS/VS COBOL Compiler and Library	5740-CB1
OS COBOL Interactive Debug	5734-CB4
VS Basic Processor	5748-XX1
OS PL/I Optimizing Compiler and Libraries	5734-PL3
OS PL/I Checkout Compiler	5734-PL2
DOS/VS COBOL Compiler and Library	5746-CB1
DOS PL/I Optimizing Compiler and Library	5736-PL3

Data Set Compatibility Considerations

CMS can read and update VSAM data sets that were created under DOS/VS or OS/VS. In addition, VSAM data sets created under CMS can be read and updated by DOS/VSE or OS/VS.

However, 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. DOS/VS 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 unique data sets are defined, and space is released whenever data spaces, catalogs, and unique data sets are deleted.

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 be used to 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.

User Requirements and Considerations

Because the CMS support of VSAM and access method services is based on DOS/VS access method services, you must order a DOS/VS system (Release 31 and above and use the DOS/VS starter system when you install the CMS VSAM support. In order to install CMS VSAM from the DOS/VS Release 33 starter system, you must be sure to use the current level of VM/370 installation files (Release 3 PLC 8 or later). Also, the CMS/DOS support must be installed before you install VSAM under CMS.

Distribution of VSAM and CMS

VM/370 does not distribute the DOS/VS VSAM and access method services routines that it uses. You are responsible for ordering Release 31 and above DOS/VS systems. The VM/370 starter system has an installation EXEC procedure, VSAMGEN, that generates CMS support for VSAM and access method services using a restored DOS/VS starter system.

Planning Considerations for CMS/DOS

Installations that use CMS/DOS must also have available a DOS/VS system (Release 31 and above). Therefore, if you want to use CMS/DOS you must first order and install a DOS/VS system. Also, if you want to use the DOS/VS COBOL and DOS PL/I compilers under CMS/DOS, you must order them and install them on your DOS/VS system.

DOS/VS System Generation Considerations

The CMS/DOS support in CMS uses a real DOS/VS system pack in read-only mode. CMS/DOS provides the necessary interface and then fetches DOS/VS logical transients and system routines directly from the DOS/VS system libraries. Also, CMS/DOS fetches the DOS/VS COBOL and DOS PL/I compilers directly from the DOS/VS system or private core image libraries.

It is your responsibility to order the DOS/VS system and then generate it. Also, if you plan to use DOS compilers, you must order the DOS/VS COBOL and DOS PL/I Optimizing compilers and install them on the same DOS/VS system.

When you install the compilers on the DOS/VS system, you must link-edit all the compiler relocatable modules using the following linkage editor control statement:

```
ACTION REL
```

You can place the link-edited phases in either the system or the private core image library.

When you later invoke the compilers from CMS/DOS, the library (system or private) containing the compiler phases must be identified for CMS. You identify all the system libraries to CMS by coding the filemode letter that corresponds to that DOS/VS system disk 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 DOS/VS system and private disks must be linked to your virtual machine and accessed before you issue the commands to identify them for CMS.

CMS/DOS has no effect on the update procedures for DOS/VS, DOS/VS COBOL, or DOS PL/I. You should follow the normal update procedure for applying IBM-distributed coding changes to them. PTFs that must be applied are listed in the current VM/370 Release Guide.

When the DOS/VS System Must Be Online

Most of what you do in the CMS/DOS environment requires that the DOS/VS system pack and/or the DOS/VS private libraries be available to CMS/DOS. In general, you need these DOS/VS volumes whenever:

- You use the DOS/VS COBOL compiler or DOS PL/I compiler. The compilers are executed 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 books from your system or private source statement library.
- You invoke one of the librarian programs: DSERV, RSERV, SSERV, PSERV, or ESERV.
- You link-edit DOS programs that use LIOCS modules. CMS/DOS link edits LIOCS routines with the DOS program from DOS/VS system or private relocatable libraries.
- You execute DOS programs that fetch phases directly from DOS/VS system or private core-image libraries.

A DOS/VS system pack is usable when it is:

- Defined for your virtual machine
- Accessed
- Specified, by mode letter, on the SET DOS ON command

A DOS/VS private library is usable when it is:

- Defined for your virtual machine
- Accessed
- Identified via ASSGN and DLBL commands

The DOS/VS system pack and private libraries may reside on full packs or minidisks.

CMS/DOS Tape Handling

CMS/DOS does not process tape labels. In general, CMS/DOS either bypasses labels on input tapes or passes control to a user routine to process header labels on input tapes. CMS/DOS processes all output tapes as tapes with no labels. Trailer labels are not supported for input tapes or output tapes.

CMS/DOS passes control to user label routines, if there are any, for input tapes with standard or nonstandard labels.

If a tape which is opened as an output tape already has a header label (standard or nonstandard), CMS/DOS writes over that label when it writes data to the tape.

There is no equivalent in CMS/DOS to the DOS/VS TLBL control statement. The TLBL label function is not required in CMS/DOS.

Standard Label Cylinder

CMS/DOS does not support a standard label cylinder. If the real DOS/VS system pack used by CMS/DOS has a standard label cylinder, it is not used.

In CMS/DOS, ASSGN and DLBL commands provide functions similar to those provided by the DOS/VS ASSGN, DLBL, and EXTENT control statements. In DOS/VS 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 standard label cylinder.

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 execute a program in an EXEC file and invoke that EXEC file by its filename.

Planning Considerations for Virtual Machine Operating Systems (Other than CMS)

This section contains information about the following:

- The VM/VS Handshaking Feature
- Multiple Virtual Machines Using the Same Operating System
- The RSCS Virtual Machine
- VM/370 Using Channel Switching
- Operating Systems Using Reserve/Release

The VM/VS Handshaking Feature

The VM/VS Handshaking feature is a communication path between VM/370 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:

- Processing pseudo page faults
- Closing VM/370 spool files when the system control program's output writer operation is complete
- Providing an optional nonpaging mode for operating systems executing under the control of VM/370
- Providing miscellaneous enhancements for an operating system's virtual machine executing under the control of VM/370

A page fault is a program interrupt that occurs when a page that is marked "not in storage" is referred to by an instruction in an active page. The virtual machine requesting the page is placed in the wait state while the requested page is brought into real storage. However, with the VM/VS handshaking feature, a multiprogramming operating system executing under the control of VM/370 in a virtual machine may dispatch one task while waiting for VM/370 to honor a page request for another task. When the pseudo page fault portion of VM/VS Handshaking is active, VM/370 sends a pseudo page fault (program interrupt X'14') to the guest system. When the guest system recognizes a pseudo page fault, it places only the task waiting for the page in page wait and can dispatch any of its other tasks.

Since no paging is done by the operating system using VM/VS handshaking, ISAM programs are treated by VM/370 as if they are being run from fixed storage locations. Therefore, in order to execute the ISAM program 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 invoking the CP CLOSE command when the task or job has completed. Once these spool files are closed, they can be processed by VM/370 without operator intervention.

Operating systems using VM/VS handshaking can execute in nonpaging mode. Nonpaging mode exists when (1) the handshaking feature is active, and (2) the operating systems virtual storage size equals the virtual storage size of the VM/370 virtual machine. When the guest operating system executes in nonpaging mode, fewer privileged instructions are executed and duplicate paging is eliminated. Note that such a virtual machine may have a larger working set when it is in nonpaging mode than when it is not in nonpaging mode.

Also, there are some other enhancements for guest systems using VM/VS handshaking while executing under the control of VM/370. With the handshaking feature, the guest system avoids some of the instructions and procedures that would be inefficient in the VM/370 environment.

When the VM/VS Handshaking feature is active, the operation of a system control program closely resembles the standalone operation because much redundancy of function between VM/370 and the operating system is eliminated. For instance:

- One VS1 task can be dispatched while another is waiting for a page to be brought into real storage.
- There is less need for the virtual machine operator to intervene because output files are automatically closed and processed.

Note: Even when the handshaking feature is active for a virtual machine, the pseudo page fault portion of the handshaking feature is not available until it is set on with the CP SET PAGEX ON command; this command can set pseudo page fault handling on and off.

Multiple Virtual Machines Using the Same Operating System

In general, an operating system which is to run in a virtual machine should have as few options generated as possible. This is also true when several virtual machines share a system residence volume. Very often, options that improve performance on a real machine have no effect (or possibly an adverse effect) in a virtual machine. For example, seek separation, which improves performance on the real machine, is redundant in a virtual machine: CP itself issues a standalone seek for all disk I/O.

Sharing the system residence volume makes it unnecessary to keep multiple copies of the operating system online. The shared system residence volume should be read-only.

The CMS system residence volume, for example, is read-only, so it can be shared among virtual machines. CMS discontiguous saved segments can also be shared among all virtual machines; they are outside the virtual storage of each of the sharing virtual machines. The CMS/DOS environment of CMS simulates DOS/VS supervisor and input/output functions, thereby allowing execution 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. Refer to the VM/370 System Programmer's Guide for more details.

The RSCS Virtual Machine

The Remote Spooling Communications Subsystem (RSCS), operating in a virtual machine, handles the transmission of files between VM/370 users and remote terminals and stations. Figure 4 illustrates a typical RSCS configuration.

Three lines of the real 3705, operating in 2703 emulation mode, are shown dedicated to the RSCS virtual machine. The communication lines are shown attached to a 3780 Data Communication Terminal, a System/3, and an OS/HASP processor running in a remote System/360 or System/370.

The RSCS machine uses the spooling facilities of VM/370 as the interface between virtual users and itself. Any user who wishes to have an output file transmitted to a remote location must associate tag information (such as destination and priority) with his file via the TAG command and spool the file to the RSCS machine's virtual reader. RSCS analyzes the tag data, enables the appropriate line, and transmits the data using the line protocol required by the receiving station.

Remote locations can submit card files to the RSCS machine and address them to either a VM/370 user or to RSCS itself for transmission to another remote station. RSCS produces a VM/370 spool file by writing that data to virtual unit record devices and, if the file is destined for a VM/370 user, sends it to the user's virtual reader via the CP SPOOL command. If the file is addressed to RSCS, it is queued on RSCS's virtual reader and then handled in the same manner as a file spooled to RSCS by a VM/370 user. In this case, it is the responsibility of the remote station that originated the data to supply the tag information.

RSCS can also function as a remote workstation of a HASP/ASP type batch processor. VM/370 users and remote stations can submit jobs to RSCS for transmission to the HASP system. After processing, HASP can return printed and/or punched output to RSCS for spooling to the real system printer or punch. For more information about RSCS, see the <u>VM/370 Remote Spooling Communications Subsystem (RSCS) User's Guide</u>.

RSCS PLANNING CONSIDERATIONS

All the files you need to generate RSCS are on the RSCS/IPCS tape.

Before you perform the RSCS generation procedure, be sure you have a virtual machine defined for RSCS. The virtual machine you intend to run RSCS should have:

- 512K of virtual storage
- A reader
- A console
- A minidisk for the RSCS system residence volume. The RSCS system disk must have a write password. See the section "Defining Your RSCS Virtual Machine" in Part 3.

You can define more than one RSCS virtual machine. Also, you can have more than one RSCS virtual machine running at the same time. However, when multiple RSCS virtual machines are running at the same time, each must have a unique user identification (userid) and local location identification (ID=linkid).

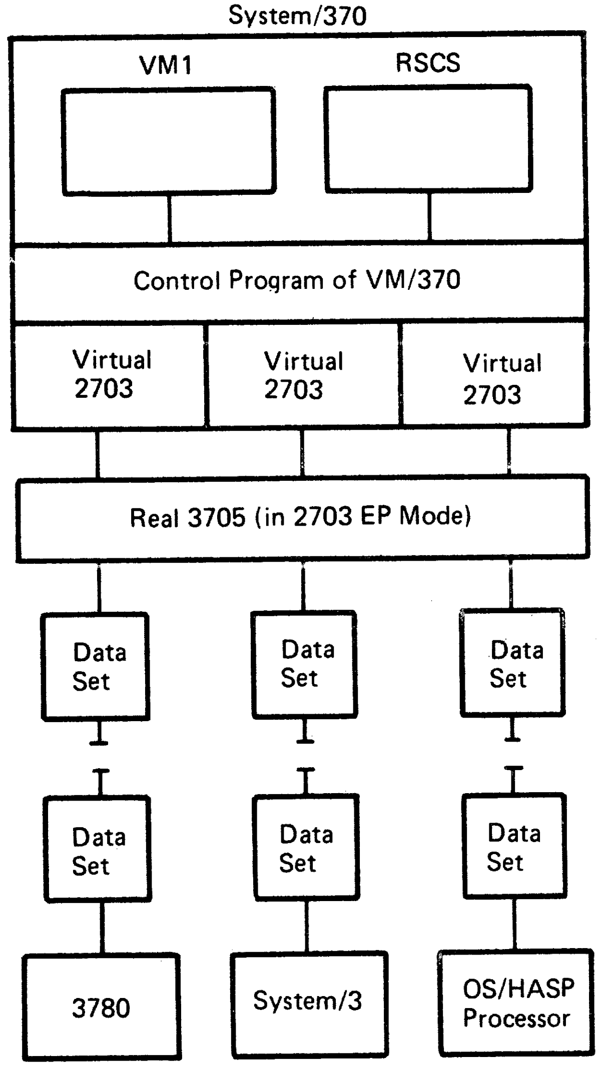


Figure 4. A Remote Spooling Communications Subsystem

When you code the GENLINK macros during the RSCS generation procedure, you must code the local location identification on the first GENLINK macro.

Information about generating and installing RSCS is in "Part 3. Generating VM/370 (CP, CMS, RSCS and IPCS)."

VM/370 Using Channel Switching

The two- or four-channel switch can be used in the following environments:

- Two processors, one running VM/370 and the other running an operating system that supports channel switching.

- Two virtual machines running under VM/370; each virtual machine operating system must support the channel switch feature (CMS does not).

- A single virtual machine running under VM/370; the virtual machine operating system must support the channel switch feature.

- A processor running VM/370 and managing multiple paths to devices through the VM/370 alternate path support. Refer to the section on "Alternate Path Support" in this document for a discussion on the VM/370 alternate path support.

You can use the two- or four-channel switch for devices attached to two processors. For example, one processor could be running VM/370 and the other could be running OS as shown in Figure 5.

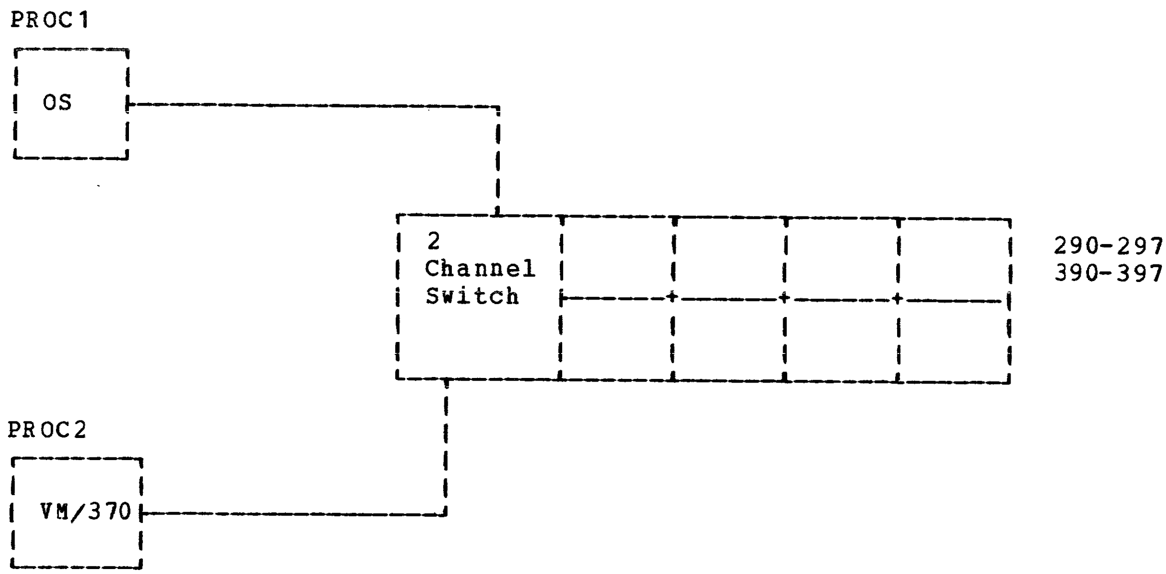


Figure 5. Channel Switching between Two Processors

VM/370 requires the following RDEVICE and RCTLUNIT macros to support this configuration:

```
RDEVICE   ADDRESS=(290,8),DEVTYPE=3330
RDEVICE   ADDRESS=(390,8),DEVTYPE=3330
RCTLUNIT ADDRESS=290,CUTYPE=3830
RCTLUNIT ADDRESS=390,CUTYPE=3830
```

These macros make it possible for you to run VM/370 on PROC1 or PROC2. If you are always going to run VM/370 on PROC2, you can eliminate one path (eliminate one set of RDEVICE and RCTLUNIT macros).

If any I/O devices controlled by VM/370 for its own exclusive use are attached to a control unit with a two- or four-channel switch, the 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 that is running VM/370. For example, one processor could be running VM/370 with OS running in a VM/370 virtual machine as shown in Figure 6. In this case, the virtual machine operating system supports channel switching.

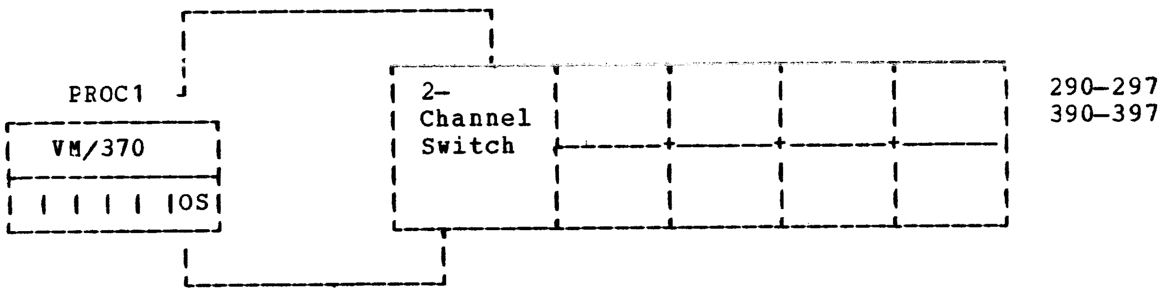


Figure 6. Channel Switching on One Processor

VM/370 requires the following RDEVICE and RCTLUNIT macros to support this configuration:

```
RDEVICE  ADDRESS=(290,8),DEVTYPE=2314
RDEVICE  ADDRESS=(390,8),DEVTYPE=2314
RCTLUNIT ADDRESS=290,CUTYPE=IFA
RCTLUNIT ADDRESS=390,CUTYPE=IFA
```

For this example, you should have all the devices associated with one path offline when you load VM/370. Otherwise, the following message is displayed:

```
DMKCPI954E DASD raddr VOLID volid NOT MOUNTED,
           DUPLICATE OF DASD raddr
```

The 2314 DASD devices can be used by the OS system running in a virtual machine if they are dedicated to that virtual machine via the ATTACH command or the DEDICATE control statement in the VM/370 directory. The device addresses generated for the virtual machine operating system 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).

- Issue the CP ATTACH command for both device addresses (580 and 680) whenever the real device is to be attached to the virtual machine.

The device addresses generated for the virtual machine operating system do not need to be the same as those on the real machine.

The devices must be used by the virtual machine as dedicated devices (attached, or defined with a DEDICATE statement in the VM/370 directory).

VM/370 alternate path logic provides support for the two channel switch, two channel switch additional feature, and the string switch feature. The purpose of alternate path support is to define alternate paths to a given device on the VM/370 processor. The virtual operating system does NOT define alternate paths. Instead, VM/370 would define alternate paths to the device by the RCTLUNIT and RDEVICE macros, respectively. VM/370 would then perform the alternate path I/O scheduling. Using Figure 6, if the installation wanted VM/370 to perform the alternate path I/O scheduling instead of the virtual operating system, 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
```

Channel-Set Switching Facility

The channel-set switching facility is a feature available on the 3033 attached processor system. 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.

The switching operating 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 attached processor environments, channel-set switching is used to continue system operation in uniprocessor mode when the main (I/O) processor is taken offline as the result of a VARY PROCESSOR OFFLINE 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 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 wait state code of X'0001'. If the error is TOD clock damage and the processor is in problem state and equipped with the channel-set switching facility, the I/O processor is taken offline. The channel-set switching feature is used to disconnect the channel-set from the failing I/O processor and to reconnect the channel-set to the attached processor. The system continues processing on the attached processor in uniprocessor mode.

The following message is issued when the channel-set is connected to the attached processor:

DMKCPU623I - CHANNEL-SET CONNECTED TO PROCESSOR nn

Operating Systems Using Reserve/Release

Shared DASD is the term used to describe 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 devices can occur when:

- A two- or four-channel switch attaches a device's control unit to two or four channels.
- String switching is utilized and the control units to which they 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 by means of the switch. Each sharing system vies 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 the neutral position, any other system, or the same one, may select the shared device and have the switch set to its interface.

It is important to note that none of the sharing systems is aware of what the other is doing with the data on the shared devices. Data integrity is the responsibility of the using program. For this reason, the hardware command, RESERVE, may be issued by a program to retain exclusive use of a shared device while a critical update to data is being performed. Device RELEASE is issued to terminate the exclusive reservation. If a shared device has been reserved for exclusive use, the system channel through which the reserve was issued will lock out any other channel, on the same or different system, from accessing the device.

Reasons for Sharing:

There are several reasons an installation 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 minimized. One update to a shared data set is needed, instead of the multiple updates that would be required if each of several systems had its own copy of the data set.
- Backup and switchover in the event of hardware failure is facilitated in a multi-system environment if the needed data is accessible to surviving systems without moving it.
- Direct access storage space may be saved, as one copy of the data is required instead of multiple copies.

Two assembler language macros, RESERVE and DEQ, are available to effect the reserving and releasing of a device. The data integrity of shared devices can be maintained by application program use of the RESERVE macro or by operating system components which 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 the OS simulation or CMS/DOS simulation packages. The SHAREOPTIONS operand which appears 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. The use of shared DASD by virtual machines should be limited to those quest operating systems which will maintain the integrity of shared data, such as catalogs, VTOCS, program libraries, etc., and will support the use of the RESERVE and DEQ macros used by application programs running under those operating systems. The only other alternative is the use of the hardware reserve or release CCWs by an application program running under CMS. In this case the application program issues the hardware reserve and release CCWs in a SIO or DIAGNOSE operation to the shared device.

VM/370 reserve/release support can be addressed in two forms: Shared DASD and Virtual Reserve/Release.

Shared DASD refers to the use of reserve/release CCW strings by virtual machine or processor operating systems for the purpose of preserving data integrity. The integrity of the data 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 via any other path.

Virtual reserve/release is the software simulation of reserve/release CCWs for minidisks. Since virtual devices associated with a minidisk all map to the same real channel interface to the device, the hardware protection is lost and a software locking structure is required to maintain the data integrity during reserve/release sequences.

The VM/370 control program and the CMS operating system do not issue reserve CCWs. The use of reserve/release remains the full responsibility of the operating system running in the virtual machine. The VM/370 initialization routine issues a release CCW to tape and DASD devices to dynamically determine whether the two-or-four channel switch feature is installed.

SHARED DASD

Operating systems that support Shared DASD use reserve/release CCWs to preserve data integrity in the following environments:

- Two virtual machines running under VM/370 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.

Reserve/Release CCWs are recognized by the VM/370 control program CCW translation routine and are executed by the hardware to preserve data integrity. In this environment 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/370 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/370, in which case the virtual machine operating system uses reserve/release CCWs, or a non-VM/370 operating system with reserve/release capability.

To support this environment, the device should be dedicated to the VM/370 virtual machine by means of the ATTACH command or DEDICATE control statement in the VM/370 directory.

In the above shared DASD environments, the use of reserve/release by virtual machine operating systems and the VM/370 alternate path support are mutually exclusive. The VM/370 control program changes a reserve CCW to a sense CCW when an alternate path has been defined for the device. The protection offered by the hardware reserve is lost. It is recommended that a single path be defined in VM/370 for devices which will be dedicated to virtual machines and then shared between other virtual machines or processors.

- A virtual machine runs under VM/370 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/370, in which case the virtual machine operating system should use reserve/release CCWs to maintain data integrity, or a non-VM/370 operating system with reserve/release capability.

The device can be defined as a minidisk, on the VM/370 processor, which begins at real cylinder 0. Again the use of reserve/release and alternate path support are mutually exclusive. It should be noted that virtual reserve/release support should not be used in this environment. The volume being shared should not contain more than one minidisk or be used for CP paging, spooling, etc., since reservation by the other processor could lock out virtual machine users or VM/370 system I/O requests to the same device.

VIRTUAL RESERVE/RELEASE

The reserve/release software simulation in VM/370 provides reserve/release protection at the minidisk level, including full volume minidisks. Virtual reserve/release is intended for use by the virtual machines that support Shared DASD (not CMS) running on the VM/370 processor. Virtual reserve/release simulation is requested by appending a character "V" to the mode operand on the MDISK statement in the directory. All subsequent links to this minidisk are subject to virtual reserve/release processing. A software locking structure is created to manage the reservation status by minidisk. The VM/370 control program then examines virtual machine channel programs and manages the reserve/release CCWs presented by the sharing virtual machines. The VM/370 control program simulates the 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 machines users who received a "device busy" indication. Diagnose users can also issue reserve/release CCWs. However, no "device busy" or "device end" status 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, at which time the Diagnose request will be redriven.

VM/370 CONTROL PROGRAM HANDLING OF A RESERVE CCW

VM/370 reserve/release support and the VM/370 alternate path support are mutually exclusive. The VM/370 CCW translation routine changes a reserve CCW to a sense CCW when alternate paths have been defined to the device from the VM/370 processor. Data integrity is not preserved when sharing a device between processors or virtual machines and alternate paths are defined. When using virtual reserve/release to share a minidisk between virtual machines on the VM/370 processor, VM/370 still changes a reserve CCW to a sense CCW when alternate paths are defined to the real device.

However, since the hardware reserve/release is simulated when virtual reserve/release is being used, the data integrity is preserved when alternate paths are defined. The chart below identifies those situations when the VM/370 control program 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)	CCW Comnd sent by VM/370 to Device	Note
Dedicated DASD or Tape	Not defined	Not applicable	Not applicable	Reserve	1
	Defined	Not applicable	Not applicable	Sense	2
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/370 system has been generated with alternate path support for those devices, it prevents the devices from being reserved. This action causes VM/370 to avoid a possible channel lockout. VM/370 does not return any indication of this action to the operating system issuing the CCW command that the device was not reserved.

[3] Without the two-channel switch special feature, VM/370 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, VM/370 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/370 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/370 processor. This path allows VM/370 virtual reserve/release processing to send a sense CCW to the device, although the reserve CCW command would be 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), VM/370 changes reserve/release CCW commands to sense CCW commands to prevent a possible channel lockout.

Figure 7. Summary of VM/370 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/370, the shared volume cannot contain more than one minidisk. The single minidisk may encompass the entire volume or a small portion of the volume and the remainder of the volume must not be referenced by CP for use as paging, spooling, etc., or by any virtual machine.

- Devices shared between processors must not be generated in VM/370's DMKRIO as having alternate paths. If there are multiple paths from the VM/370 processor to the shared devices, as well as a path from the same devices to another processor, the paths from the VM/370 processor cannot be generated in DMKRIO as alternate paths via the ALTCH or ALTCU macro operands. <u>This means that the definition of alternate paths in DMKRIO and the use of real reserve/release are mutually exclusive</u>.

RESTRICTIONS: DEVICE/MINIDISK SHARING ON A SINGLE PROCESSOR

- If more than a single 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 issued by such users preserve the data integrity. In this case, the integrity is preserved by the hardware, not by the software reserve/release support. <u>Again, the definition of alternate paths in DMKRIO and the use of real reserve/release are mutually exclusive</u>.

- 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 the 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 will still be changed to a sense CCW but the integrity will be preserved by the virtual reserve/release code.

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/370 system. The VMCF external interrupt masking is controlled by PSW bit 7 and CR0 bit 31. It is to a user's advantage to always have CR0 bit 31 set to 1 (while VMCF is in use) and control the interrupts with PSW bit 7 only. This reduces the number of LCTL instructions.

Messages and data directed to other virtual machines are logically identified via the virtual machine's userid. 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 sizes of virtual machine storage of the respective virtual machines.

Use of real storage is minimal. Only one real storage page need be locked during the data transfer. A special interrupt is used to notify one virtual machine of a pending transfer of data; this interrupt is also used to synchronize sending and receiving of data.

Under the Special Message Facility, CP acts as a virtual machine in behalf of a virtual machine that issues the CP command SMSG. The receiving virtual machine, properly programmed to accept and process special messages, authorizes itself to CP. Data (message) transfer is from CP, via the message and VMCF modules.

SUMMARY OF VMCF FUNCTIONS

VMCF functions include five data transfer operations and seven control functions. Figure 8 contains a brief summary of VMCF functions. A more detailed description of these functions and how they are implemented in VM/370 is contained in the _VM/370 System Programmer's Guide_.

The data transfer operations involve the movement of data from one virtual machine's storage to another virtual machine's storage. The VMCF control functions allow a user to manage data transfer operations from the virtual machine console.

VMCF is implemented via the CP DIAGNOSE instruction in VM/370; it is not hardware dependent. VMCF is available to any logged-on, authorized VM/370 virtual machine.

Function	Code[1]	Comments
AUTHORIZE	Control	Initialize VMCF for a given virtual machine. Once AUTHORIZE is executed, the virtual machine can execute other VMCF functions and receive messages or requests from other users.
UNAUTHORIZE	Control	Terminates VMCF activity.
SEND	Data	Directs a message or block of data to another virtual machine.
SEND/RECV	Data	Provides the capability to send and receive data on a single transaction.
SENDX	Data	Directs data to another virtual machine on a faster but more restrictive protocol than the SEND function.
RECEIVE	Data	Allows you to accept selective messages or data sent via a SEND or SEND/RECV function.
CANCEL	Control	Cancels a message or data transfer directed to another user but not yet accepted by that user.
REPLY	Data	Allows you to direct data back to the originator of a SEND/RECV function, simulating full duplex communication.
QUIESCE	Control	Temporarily rejects further SEND, SENDX, SEND/RECV, or IDENTIFY requests from other users.
RESUME	Control	Resets the status set by the QUIESCE function and allows execution of subsequent requests from other users.
IDENTIFY	Control	Notifies another user that your virtual machine is available for VMCF communication.
REJECT	Control	Allows you to reject specific SEND or SEND/RECV requests pending for your virtual machine.

[1]Data indicates a data transfer function.
Control indicates a VMCF control function.

Figure 8. Virtual Machine Communication Facility (VMCF) Functions

Planning Considerations for 3270s

VM/370 attachments can be considered either local or remote. Local attachments are all devices that do not require telecommunication lines. However, many devices that are supported for local attachment are also supported for remote attachment. Remote attachments are devices that are attached to binary synchronous lines. Such configurations usually include:

- a designated channel
- a designated communication controller or transmission control unit
- the device or control unit (for terminal attachment and/or RJE systems).

Remote Attachments

Both remote cluster and standalone configurations are supported. 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 activate and deactivate the teleprocessing lines, display stations and printers.
- CMS Editor support.
- The recording of MDR (Miscellaneous Data Recorder) records and OBR (Outboard Recording) records on the VM/370 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 allows the user to assign a screen copy function to a 3270 program function key. Pressing that key transfers the current display image, in its entirety, to an available printer attached to the same control unit. If the printer is busy or otherwise not available when the copy function is invoked, the virtual machine user receives a NOT ACCEPTED message in the screen's status area.

The following restrictions apply to VM/370's support of remote 3270s:

- 3270 terminals cannot be used as primary or alternate VM/370 system consoles.

- The number of binary synchronous lines supported by VM/370 for 3270 use is 16 minus the number of 3704/3705 Communications Controllers in NCP mode minus one (if there are any 3704/3705 Communication Controllers in EP mode).
- The CP DIAL command is not supported.

3270 Support on Binary Synchronous Lines

The supported display devices on binary synchronous lines have the same flexibility and usefulness as locally attached 3270 devices, except for the following limitations:

- Display Information Inquiry and Retrieval Speed -- Because the 3270 remote stations are subject to (1) relatively slow teleprocessing transmission speeds and (2) the mechanics of polling operations, screen display and data entry are not as rapid for remote 3270 devices as they are for locally attached 3270s.
- CP DIAL and ATTACH Commands -- The CP DIAL and ATTACH commands are not supported for remote 3270 stations.
- Hard Copy of 3270 Screen Image -- Just as users of locally attached 3270 display devices can spool their virtual console input and output to the system printer, so can the users of remote 3270 display devices. However, for remote 3270 users, and those local 3270 users whose terminals may be physically distant from the system printer, VM/370 provides a limited hard-copy facility at the local and remote locations.
- TEST REQUEST and SYSTEM TEST Keys -- These keys on the 3270 terminal are not supported for remote 3270s. The Test Request function on locally-attached 3277s is supported by the TEST REQUEST key; it is supported on locally-attached 3278s by the SYSTEM TEST key.

Remote Hardware Configurations Supported

VM/370's support of remote 3270s requires:

- 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. The transmission control units supporting remote 3270s on binary synchronous lines are:

- Integrated Communications Adapter (ICA).
- IBM 2701 Data Adapter Unit with Synchronous Data Adapter Type II.
- IBM 2703 Transmission Control with Synchronous Data Adapter Type II.
- IBM 3704 and 3705 Communications Controllers in emulation mode. A 3704/3705 line is in emulation mode when it is controlled by the Emulation Program (EP).

Cluster and standalone control units are supported for remote 3270s. The IBM 3271 Control Unit Model 2, and IBM 3274 Control Unit Models 1B and 1C support clusters of up to 32 display stations and/or printers. The IBM 3276 Control Unit Display Station Models 2, 3, and 4 support clusters of up to 8 display stations and/or printers. The IBM 3275 Display Station, Model 2, is the standalone 3270 device that can be remotely attached.

Note: The 3276, with a minimum configuration can also be considered a standalone 3270 device.

The following devices are supported when attached to the 3271 control unit:

- IBM 3277 Display Station, Model 2
- IBM 3284 Printer, Model 2
- IBM 3286 Printer, Model 2
- IBM 3287 Printer, Models 1 and 2
- IBM 3288 Line Printer, Model 2

Note: The 3271/3272 Attachment Feature #8330 is a prerequisite when the IBM 3287 Printer is attached to the 3271 control unit.

The following devices are supported when attached to the 3274 Control Unit Model 1C:

- IBM 3277 Display Station Model 2
- IBM 3278 Display Station Model 2, 3 and 4[1]
- IBM 3284 Printer Model 2
- IBM 3286 Printer Model 2
- IBM 3287 Printer Models 1 and 2
- IBM 3289 Printer Models 1 and 2

The following devices can be attached to the 3276 Control Unit Display Station Models 2, 3, and 4:

- IBM 3278 Display Station, Models 2, 3, and 4[1]
- IBM 3287 Printer, Models 1 and 2
- IBM 3289 Printer, Models 1 and 2

The IBM 3275 Display Station, Model 2, is a standalone control unit and display station. You can attach the IBM 3284 Printer, Model 3, to the 3275. In addition, you can attach the IBM 3286 Printer, Model 3, to the 3275 if RPQ MB4317 is installed. With the 3275 Dial Feature #3440, you can connect the 3275 to a computer over switched lines by using a | Western Electric[2] or equivalent data set. The 3275 Dial Feature does | not support full screen read/write.

System Generation Requirements for Remotely Attached Display Systems

When you generate VM/370 you must code the appropriate CLUSTER, TERMINAL, and RDEVICE macros and assemble them as part of the DMKRIO (real I/O configuration) file. Then, after the DMKRIO file assembles successfully, you must make a list of the resource identification codes of all the remote 3270 lines and terminals. Give the list to the operations group at your installation; the members of that group need this information when they issue the CP commands that control the operation of remote 3270 lines and devices.

----

[1]Models 3 and 4 default to the 1920 character screen size and are functionally equivalent to the Model 2.
[2]Trademark of Western Electric Co.

THE CLUSTER MACRO

Code one CLUSTER macro for each 3271 and 3274 control unit Model 1C, each 3276 control unit display station, and each 3275 display station. Only a maximum of 16 CLUSTER macros is allowed. Each CLUSTER macro must have a unique label. The CLUSTER macro is described in the "Preparing the Real I/O Configuration File (DMKRIO)" section of "Part 2. Defining Your VM/370 System."

THE TERMINAL MACRO

Code one TERMINAL macro for each display station and printer that is attached to the clustered 3271.

Code one TERMINAL macro for each display station and printer that is attached to the clustered 3274 control unit, Model 1C. For a 3274 control unit Model 1C, the TERMINAL macros for 3278s, 3287s on a Type A adapter, and 3289s must precede the TERMINAL macros for 3277s, 3284s, 3286s, 3287 on a Type B adapter and 3288s.

Code one TERMINAL macro for each display station and printer that is attached to a clustered 3276. Since the 3276 contains one integral display unit, code one TERMINAL macro for the 3276 as a 3277. Code each 3278 attached to the 3276 as a 3277, and each 3287 attached to the 3276 as a 3284 or 3286. Code each 3289 as a 3288.

Code one TERMINAL macro for the 3275 display station. If a 3284 or 3286 printer is attached to the 3275, code MODEL=3 on the TERMINAL macro. The TERMINAL macro is described in the "Preparing the Real I/O Configuration File (DMKRIO)" section of "Part 2. Defining Your VM/370 System."

THE RDEVICE MACRO

Code one RDEVICE macro for each binary synchronous line used for remote 3270s (code one RDEVICE macro for each CLUSTER macro you code). A maximum of 16 RDEVICE macros for lines to support remote 3270s is allowed.

The RDEVICE macro is described in Part 2. However, the format of the RDEVICE macro for the binary synchronous line and transmission control unit for remote 3270s is included here to help you code the macro correctly. The format of an RDEVICE macro for a communication line that supports remote 3270s is:

Name	Operation	Operands
	RDEVICE	ADDRESS=cuu,
		DEVTYPE= {2701 | 2703 | ICA | 3704 | 3705},
		ADAPTER=BSCA
		[,BASEADD=cuu]
		,CLUSTER=label

where:

ADDRESS=cuu
is the real I/O address of the binary synchronous line to be used by remote 3270s.

The address, cuu, is three hexadecimal digits from 000 to FFF.

DEVTYPE= {2701 | 2703 | ICA | 3704 | 3705}
is the device type of the transmission control unit that controls the line.

Note: The lines attached to the 3704/3705 must be controlled by the Emulation Program (EP).

ADAPTER=BSCA
is the terminal adapter for the transmission control unit. BSCA represents the:

- IBM Binary Synchronous Terminal Adapter Type II for a 2701
- IBM Binary Synchronous Terminal Control Type II for a 2703
- Integrated Communications Adapter (ICA) on the System/370 Models 135, 135-3, and 138
- IBM 3704 and 3705 Communications Controllers

BASEADD=cuu
is the native subchannel address (load address) of the 3704/3705 that controls the physical line or lines. This operand is required for correct operation of VM/370 recovery management for emulator lines on a 3704/3705. Specify this operand only for 3704 and 3705.

CLUSTER=label
is the label of a CLUSTER macro that defines the cluster or standalone station attached to this line.

Examples

The following examples are RDEVICE macros describing a nonswitched point-to-point communication line connected to a 2701, 2703, and 3705.

Example 1:

```
RDEVICE ADDRESS=078,DEVTYPE=2701,ADAPTER=BSCA,CLUSTER=CLUST001
```

The cluster station that is connected to this line is defined by the CLUSTER macro labeled CLUST001. The line at address 078 is controlled by a 2701 transmission control unit.

Example 2

```
RDEVICE ADDRESS=080,DEVTYPE=2703,ADAPTER=BSCA,CLUSTER=CLUST020
```

The line at address 080 is controlled by a 2703 transmission control unit and the corresponding CLUSTER macro is labeled CLUST020.

Example 3

```
RDEVICE ADDRESS=0B8,DEVTYPE=3705,ADAPTER=BSCA,                    X
        BASEADD=0B0,CLUSTER=CLUST030

RDEVICE ADDRESS=0B0,DEVTYPE=3705,ADAPTER=TYPE1,MODEL=B4           X
        CPTYPE=EP,CPNAME=CEP0B0
```

The line at address 0B8 is controlled by a 3705 Communications Controller and the corresponding CLUSTER macro is labeled CLUST030.

Note: Failure to code the CPNAME operand on the RDEVICE macro instruction for the 3704/3705 base address causes VM/370 to mark the device "not operational" at IPL time. The cluster on that 3704/3705 is therefore unusable.

THE RESOURCE IDENTIFICATION CODES

After the real I/O configuration file (DMKRIO) assembles successfully, generate a list of the resource identification codes that correspond to each line address. Give the list to the operations group at your installation. The operator needs to know the resource identification code when he issues the commands to control the operation of the remote 3270 lines and terminals.

The resource identification code is a four-digit hexadecimal code. The low-order three digits of the resource identification code are the resource address. The high-order digit is the line code.

The resource address is generated by VM/370; the order in which the TERMINAL macros appear in the real I/O configuration file (DMKRIO) determines the resource addresses of the terminals defined. Each CLUSTER macro defines a 3270 control unit; each 3270 control unit has a resource address of X'00'. The device defined by the first TERMINAL macro after the CLUSTER macro (in the DMKRIO file) has a resource address of X'01', the second has a resource address of X'02', up to the maximum of X'20'. This resource address makes up the low-order three digits of the resource identification code.

The line code is also generated by VM/370. Refer to the assembly listing for DMKRIO to determine the line code (the high-order digit of the resource identification code). Locate the label DMKRIORN near the end of the DMKRIO assembly listing. This label identifies a list of all the lines used by remote 3270s and by 3704/3705 Communications Controllers in NCP mode. The high-order digit is the line code and is assigned according to the order in which the line addresses appear in the list. The first line address is assigned a line code 0 to complete its resource identification code, the second is assigned 1, and so on up to the last line. VM/370 supports a maximum of 16 binary synchronous lines for use by remote 3270s; thus, the maximum value of the high-order digit is F. Figure 9 shows you a sample DMKRIO assembly listing and the corresponding line codes.

Sample of DMKRIO Assembly Listing	Line Code (in hexadecimal)
DMKRIORN DC F'4'	
DC AL2((RDV078-DMKRIODV)/8)	
DC XL2'078'	0
DC AL2((RDV07A-DMKRIODV)/8)	
DC XL2'07A'	1
DC AL2((RDV079-DMKRIODV)/8)	
DC XL2'079'	2
DC AL2((RDV07B-DMKRIODV)/8)	
DC XL2'07B'	3

Figure 9. Example of Determining Line Code for Remote 3270 Resource Identification Codes

Once you determine the resource identification codes for the devices in your remote 3270 configuration, generate a list for operations. The list should include the following information:

- Line address
- Line code
- Resource address
- Label of plug on control unit panel
- Resource Identification code
- Device type

Note: The plug panel of the 3271 control unit and 3274 control unit Model 1C have up to 32 ports where terminals and printers can be attached. The 3276 has up to 8 ports where the 3276 integrated display is attached and where up to 7 additional terminals or printers can be attached.

AN EXAMPLE OF A REMOTE 3270 CONFIGURATION

This example shows you the contents of the real I/O configuration file to define the following remote 3270 configuration:

- A clustered 3271 control unit with eight ports
- A standalone 3275 display station

The macros are coded so that the 3271 clustered control unit can support eight display devices, or six display devices and two printers. To define such a configuration, you must code 2 CLUSTER, 16 TERMINAL, and 2 RDEVICE macros defining the 2 separate clusters. A 3275 standalone control unit, with one display and one printer, is also supported by the following macros. To define it, you must code one CLUSTER, one TERMINAL, and one RDEVICE macro.

The real I/O configuration file for this example is:

```
DMKRIO    CSECT
CLUST078  CLUSTER   CUTYPE=3271,GPOLL=407F,LINE=078
          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
          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
```

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. Display devices can be interchanged among resource addresses allocated to display devices and printers can be interchanged among resource addresses allocated to printers; but a printer cannot be attached at an address defined for a display device, and vice versa.

After the DMKRIO file assembles successfully, you should generate a list of resources for the operations group. Your list should be similar to the list shown in Figure 10; it corresponds to the preceding example.

Line Address	Line Code	Resource Address	Label of Plug in Control Unit Panel	Resource Identi- fication Code	Device Type
078	0	000	--	0000	cluster
		001	0	0001	display
		002	1	0002	display
		003	2	0003	display
		004	3	0004	display
		005	4	0005	display
		006	5	0006	display
		007	6	0007	printer
		008	7	0008	printer
079	2	000	--	2000	cluster
		001	0	2001	display
		002	1	2002	display
		003	2	2003	display
		004	3	2004	display
		005	4	2005	display
		006	5	2006	display
		007	6	2007	display
		008	7	2008	display
07A	1	000	--	1000	cluster
		001	--	1001	display
		002	--	1002	printer

Note: The line code is determined by referring to Figure 9; it corresponds to this example.

Figure 10. Sample List of Resource Identification Codes for Operations

Local Attachments

Those control units that are attached directly to the processor channels include:

- IBM 3272 Control Unit
- IBM 3274 Control Unit, Model 1B

The display stations and printers that are attached directly to the control unit include:

- IBM 3277 Display Station
- IBM 3278 Display Station
- IBM 3284 Printer
- IBM 3286 Printer
- IBM 3287 Printer
- IBM 3288 Printer
- IBM 3289 Printer (will not attach to a 3272)

Note: The printers listed above are supported for the PFnn copy function only.

The 3272 control unit can support the following devices:

- IBM 3277 Display Station
- IBM 3284 Printer
- IBM 3286 Printer
- IBM 3287 Printer
- IBM 3288 Printer

The 3274 control unit, Model 1B can support the following devices:

- IBM 3277 Display Station
- IBM 3278 Display Station
- IBM 3284 Printer
- IBM 3286 Printer
- IBM 3287 Printer
- IBM 3288 Printer
- IBM 3289 Printer

Local Hardware Configurations Supported

CONTROL UNITS

The following control units are supported when locally attached on a byte multiplexer, block multiplexer, or selector channel to support 3270 devices:

- IBM 3272 Control Unit Model 2 for attachment of up to 32 3277 Display Stations Model 2, 3284 Printers Model 2, 3286 Printers Model 2, 3287 Printers, Models 1 and 2, and 3288 Line Printers Model 2. To support this configuration, the following may be required:

  --Device Adapter feature (#3250) is required if more than four devices are attached to the 3272. Up to four additional devices can be attached with each device adapter.

  --A 3271/3272 Attachment (#8330) is required to attach each 3287 Printer.

- IBM 3274 Control Unit Model 1B (supported as a 3272) for attachment of up to 32 display stations and printers. All of the 32 devices can be 3278 Display Stations Models 2, 3, and 4[1] (supported as 3277s), 3287 Printers Models 1 and 2 (supported as 3284s or 3286s), and 3289 Line Printers Models 1 and 2 (supported as 3288s). A maximum of 16 of the 32 devices can be 3277 Display Stations Model 2, 3284 Printers Model 2, 3286 Printers Model 2, 3287 Printers Models 1 and 2, and 3288 Line Printers Model 2. To support this configuration, the following are required:

  --The basic 3274 Control Unit permits attachment of up to 8 devices (3278, 3287, and 3289). At least one 3278 is required.

  --Each of Terminal Adapter Types A1, A2, and A3 (#6901, #6902, and #6903) permits the attachment of up to 8 additional devices per adapter (types 3278, 3287, and 3289). Only 1 of each type terminal adapter is permitted. Terminal Adapter Type A1 is a prerequisite to Type A2, and Type A2 is a prerequisite to Type A3.

---

[1]Models 3 and 4 are supported as Model 2 via hardware default.

--Terminal Adapter Type B1 (#7802) permits the attachment of up to 4 additional devices (types 3277, 3284, 3286, 3287 and 3288). Only 1 adapter is permitted.

--Each of the Terminal Adapter Types B2, B3, and B4 (#7803, #7804, and #7805) permits the attachment of 4 additional devices per adapter (types 3277, 3284, 3286, 3287, and 3288). Only 1 of each type terminal adapter is permitted. Terminal Adapter Type B1 is a prerequisite to Type B2, Type B2 is a prerequisite to Type B3, and Type B3 is a prerequisite to Type B4. Terminal Adapter Types A1, and B3 are mutually exclusive.

--A 3274/3276 Attachment (#8331) is required for each 3287 that attaches to the basic 3274 Control Unit, or that attaches to Terminal Adapter Types A1, A2, or A3. A 3271/3272 Attachment (#8330) is required for each 3287 that attaches to Terminal Adapter Types B1, B2, B3, or B4.

And only required if a Type B adapter is used:

--Control Storage Expansion feature (#1801) provides the ability to access control unit storage addresses above 64K.

--Extended Function Storage feature (#3622) provides additional storage required to support particular attachments or configurations. Control Storage Expansion feature (#1801) is a prerequisite.

TERMINALS

The IBM 3277 Display Station Model 2 requires one of the following features:

--66 Key EBCDIC Typewriter Keyboard (#4630)
--66 Key EBCDIC Data Entry Keyboard (#4631)
--78 Key Operator Console-Keyboard (#4632)
--78 Key EBCDIC Typewriter Keyboard (#4633)

The following features, while not required, enhance the convenience, security and usability of this terminal:

--Keyboard Numeric Lock (#4690)
--Audible Alarm (#1090)
--Operator Identification Card Reader (#4600)
--Lowercase Character Display (RPQ #8K0366)
--Security Keylock (#6340)

Note: Unless a 3270 terminal is dedicated to a virtual machine, it is supported only as a virtual 3215; that is, it is not supported by VM/370 as a real local or remote 3270.

The 3278 Display Station requires one of the following features:

--75 Key EBCDIC Typewriter Keyboard (#4621)
--75 EBCDIC Data Entry Keyboard (#4622)

Note: A prerequisite is the EBCDIC Character Set feature (#9082) on the 3278.

The following features, while not required, enhance the convenience, security, and usability of this terminal:

--Keyboard Numeric Lock (#4690)
--Audible Alarm (#1090)
--Magnetic Slot Reader (#5005) with Magnetic
  Reader Control (#4999)
--Security Keylock (#6340)

Note: Lowercase Character Set is standard on this terminal.

(The Test Request Function on locally-attached 3277s is supported by the TEST REQUEST key; it is supported on locally-attached 3278s by the SYSTEM REQUEST key.)

PRINTERS

The following printers are supported:

- IBM 3284 Printer, Model 2.
- IBM 3286 Printer, Model 2.
- IBM 3288 Printer, Model 2.
- IBM 3287 Printer, Models 1 and 2.
- IBM 3289 Printer, Models 1 and 2.

System Generation Requirements for Locally Supported Display Systems

System generation requirements for locally-supported terminals and control units are no different than are the requirements for DASD-supported or unit record devices. The channel, control unit, and devices are handled by their respective RCHANNEL, RCTLUNIT, and RDEVICE macros. See "Part 2. Defining Your VM/370 System" for further details.

Generating a VM/370 System that Supports the 3704/3705

The generation of a 3704 or 3705 Communications Controller control program that runs under the control of VM/370 is normally done after the VM/370 system generation is completed. However, when a 3704 or 3705 is to be generated, the following preparations must be made:

- An RDEVICE macro instruction for the 3704 or 3705 must be included in the real I/O configuration (DMKRIO) file.

- 3704/3705 control programs that are to be used by VM/370 must be stored on a CP-owned volume in the page format that is currently used for saved virtual machine systems (that is, those created by the SAVESYS command). Each 3704/3705 control program image to be saved must be defined by a NAMENCP macro instruction in the system name table (CP module DMKSNT) and saved with the SAVENCP command.

- Enough space to contain the 3704/3705 control program image must be allocated on the CP-owned volume specified in the NAMENCP macro instruction.

Note: The alternate console for VM/370 must not be on a telecommunication line on a real 3704/3705, unless the 3704/3705 is loaded by another operating system (OS/VS1, OS/VS2, or DOS/VS) before VM/370 is loaded.

Part 4 contains a complete discussion on generating a 3704 or 3705 control program. It describes the support provided with the Emulation Program (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 Part 2. However, the format of the RDEVICE macro for a 3704/3705 is included here to help you code the macro correctly. The format of the RDEVICE macro for an IBM 3704/3705 is:

```
| Name   |Operation   | Operands
|--------|------------|------------------------------------------------
|        | RDEVICE    |    ADDRESS={ cuu      }
|        |            |            { (cuu,nn) } ,
|        |            |
|        |            |    DEVTYPE={3704}
|        |            |            {3705} ,
|        |            |
|        |            |    ADAPTER={TYPE1}
|        |            |            {TYPE2}
|        |            |            {TYPE3}
|        |            |            {TYPE4}
|        |            |            {IBM1 }
|        |            |            {TELE2}
|        |            |            {BSCA }
|        |            |
|        |            | [,MODEL=ab]       [,SETADDR={ 0 }]
|        |            |                   [         { 1 }]
|        |            |                   [         { 2 }]
|        |            |                   [         { 3 }]
|        |            |                   [         { 4 }]
|        |            |
|        |            | [,CPTYPE=EP ]   [,CPNAME=ncpname]
|        |            |
|        |            | [,BASEADD=ccu]
```

where:

ADDRESS={cuu | (cuu,nn)}

is the real device address (cuu) of the 3704/3705. Use the (cuu,nn) form to generate multiple (nn) real device blocks (RDEVBLOKs) when CPTYPE=EP is specified.

DEVTYPE={3704 | 3705}

is the device type. You should specify 3704 or 3705 instead of 2701, 2702, or 2703 when CPTYPE = EP.

ADAPTER={TYPE1 | TYPE2 | TYPE3 | TYPE4 | IBM1 | TELE2 | BSCA}

identifies either the channel adapter accessed by the specified real address (TYPE1, TYPE2, TYPE3, or TYPE4), or a line adapter if this is an emulator line group (IBM1, TELE2, or BSCA). Only TYPE1 is valid for a device type of 3704. For DEVTYPE=3705, TYPE1 or TYPE4 may be coded. In identifying the line adapter, IBM1, TELE2, or BSCA can be specified only in relation to another RDEVICE macro that has ADAPTER=TYPE1, TYPE2, TYPE3, or TYPE4.

MODEL=ab

is the 3704/3705 model letter and number, respectively. The model number determines the size of the 3704/3705 storage. See Figure 11 for a list of the model numbers and the corresponding storage sizes.

You should enter both a letter and a number. However, if only a single numeric character is entered, an MNOTE is issued and the number is treated as model data for a 3704 or 3705-I, depending on the value of DEVTYPE.

SETADDR= {0 | 1 | 2 | 3 | 4}

indicates which set address (SAD) command must be issued prior to enabling the emulator lines.

CPTYPE={EP}

indicates which type of 3704/3705 control program is run in this 3704/3705.

The valid adapter types are as follows:

ADAPTER=	CPTYPE= EP
TYPE1	Yes
TYPE2	No
TYPE3	No
TYPE4	Yes

CPNAME=ncpname

is the one- to eight-character name of the control program to be automatically loaded in the 3704/3705 at system IPL time. If an automatic load is not desired, omit this operand.

Note: Failure to code the CPNAME operand on the RDEVICE macro for the 3704/3705 base address causes VM/370 to mark the device "not operational" at IPL time. The cluster on that 3704/3705 is therefore unusable.

BASEADD=ccu

is the native address (load address) of the 3704/3705 that controls the physical line(s). This operand is required for correct operation of VM/370 recovery management for emulator lines based on a 3704/3705.

This operand is valid only if ADAPTER=IBM1, TELE2, or BSCA.

IBM 3704 Communications Controller Model	Storage		IBM 3705 Communications Controller Model	Storage
A1	16K	3705-I	A1, B1, C1, D1	16K
A2	32K		A2, B2, C2, D2	48K
A3	48K		B3, C3, D3	80K
A4	64K		B4, C4, D4	112K
			C5, D5	144K
			C6, D6	176K
			D7	208K
			D8	240K
		3705-II	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

Figure 11. IBM 3704/3705 Models

SPECIAL CONSIDERATIONS FOR CODING THE RDEVICE MACRO

The 3704/3705 Communications Controllers have varied uses. Consequently, the control program generation for the 3704/3705 is complex.

EP-Only Control Programs

If the 3704/3705 is to be run in emulation mode:

- Use the (cuu,nn) form of the ADDRESS operand to generate multiple RDEVBLOKs.
- Specify the appropriate name for CPNAME.

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 also specify the CPNAME and ADAPTER=TYPE1 operands on the RDEVICE macro, the additional RDEVBLOKs are generated as ADAPTER=IBM1 and SETADDR=4.

Summary of RDEVICE Macro Coding Considerations

For each physical 3704/3705, there should be only one RDEVICE macro which specifies the ADAPTER=TYPE1, TYPE2, TYPE3, or TYPE4, MODEL, CPTYPE, and CPNAME operands. This RDEVICE macro defines the base address of the 3704/3705 (that is, the real address used to perform the load and dump operations). If the physical device is a 3705 with two channel adapters installed, there may be a second RDEVICE macro that specifies the ADAPTER=TYPE1, TYPE2, TYPE3, or TYPE4, MODEL, and CPTYPE operands. There must never be a second use of the CPNAME operand. Even if CPTYPE=EP is specified, the 3704/3705 base address cannot be used as a telecommunication line; its function is only to load and dump the 3704/3705, and 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 deck all of the RCTLUNIT macros required to specify those real addresses which the EP control program may use.

If you have a 3704/3705 and a 2701/2702/2703 on the same VM/370 system, the virtual addresses for the 3704/3705 must not be the same as any of the real 2701/2702/2703 addresses.

Examples

Examples of RDEVICE macro specifications follow. For convenience, the continuation character in position 72 is not shown.

Example 1

```
RDEVICE ADDRESS=(020,16),
        DEVTYPE=3704,
        MODEL=A2,
        ADAPTER=TYPE1,
        CPNAME=CEP020
```

This describes a 32K 3704 at address X'020', with 15 emulator lines addresses 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 'CEP020'.

Example 1a

```
RDEVICE ADDRESS=(030,16),
        DEVTYPE=3704,
        ADAPTER=IBM1,
        SETADDR=2,
        BASEADD=020
```

This describes an additional 16 emulator lines on the same 3704 specified by Example 1.

Creating an Entry in the System Name Table

It is necessary to create an entry in the system name table (DMKSNT) for each unique 3704/3705 control program that you generate. If you can foresee generating several versions of the 3704/3705 control program, define extra entries in the system name table when you generate VM/370. In this way, you need not regenerate the VM/370 system just to update the system name table. If you should have to regenerate the VM/370 system just to add a new entry to the system name table, see the discussion about the GENERATE EXEC procedure in "Part 5. Updating VM/370."

The NAMENCP macro is described in Part 2.

Reserving DASD Space for the 3704/3705 Control Program Image

DASD space to contain the 3704/3705 control program image must be reserved on a CP-owned volume. The DASD space reserved should be sufficient to contain the number of pages specified in the SYSPGCT operand of the NAMENCP macro, plus one or more for system use, as follows:

- If CPTYPE=EP, allow only one extra page.

These additional pages are used to store the reference table information provided by the SAVENCP program.

Alternate Path Support

Alternate path logic provides support for the two channel switch and two-channel Switch Additional Feature and the String Switch Feature by VM/370. This support allows up to four channels on one control unit to be attached to VM/370 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, VM/370 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 initiated 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/scheduled paths and 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, etc.

2. No single user is penalized more than any other user.

3. The first in, first out (FIFO) principle is adhered to.

An example of alternate path usage is shown in the section "3850 Mass Storage System" later in Part 1.

Generating a VM/370 System that Supports the 3800 Image Library

The generation of a 3800 image library that runs under the control of VM/370 is normally done after the VM/370 system generation is completed. However, when a 3800 image library is to be generated, the following preparations must be made:

- An RDEVICE macro instruction for the 3800 printer must be included in the real I/O configuration (DMKRIO) file.
- The 3800 image libraries that are to be used by VM/370 must be stored on a CP-owned volume in the page format that is currently used for saved virtual machine systems (that is, those created by the SAVESYS command). All 3800 image libraries in the system name table (CP module DMKSNT) and saved with the IMAGELIB command.
- Enough space to contain the 3800 image library must be allocated on the CP-owned volume specified in the NAME3800 macro instruction.

Coding the RDEVICE Macro

The RDEVICE macro is described in Part 2. However, the format of the RDEVICE macro for a 3800 is included here to help you code the macro correctly. The format of the RDEVICE macro for an IBM 3800 is:

Name	Operation	Operands
	RDEVICE	ADDRESS=ccu, DEVTYPE=3800, [FEATURE=4WCGMS,] [IMAGE=imagelib,] [CHARS=ffff,] [FCB=lpi,] [DPMSIZE=n,]

ADDRESS=cuu is the real device address (cuu) of the 3800.

DEVTYPE=3800 is the device type.

FEATURE=4WCGMS is a 3800 device with 4 Writeable Character Generation Modules.

IMAGE=imagelib is the image library to be used by the 3800 printer device after a cold start if none is specified on the START command.

CHARS=ffff is the character arrangement table for the 3800 printer device to be used after a cold start if none is specified on the START command.

FCB=lpi is the FCB to be used for the page separator for the 3800 printer device after a cold start if none is specified on the START command.

DPMSIZE=n is the maximum size of the delayed queue for the 3800 printer device.

Hardware Supported

As a VM/370 real spooling device, the following hardware features of the 3800 are supported:

- Automatic loading of character arrangement tables and graphic modifications
- Full support of the additional storage character generation feature
- Forms overlay feature (flashing)
- Copy modifications

The use of multiple character arrangement tables for printing use within one spool file (TRC support) is _not_ supported.

Related Publications

The _Concepts of the IBM 3800 Printing Subsystem_ manual, Order No. GC20-1775 is intended as a first reader for those users of printers who wish to take a quick look at the non-impact IBM 3800 Printing Subsystem, at its basic concepts and at how these concepts lead to new functions that may offer different options in planning and operations.

The _Reference Manual for the IBM 3800 Printing Subsystem_, Order No. GA26-1635 provides information on the functions and features of the IBM 3800 Printing Subsystem relating to channel commands, sense bytes, and error detection, recovery, and recording. Specific information and examples are given of copy modification and control and graphic character modification.

The _IBM 3800 Printing Subsystem Programmer's Guide_, Order No. GC26-3846 provides planning and conversion information for the IBM 3800 Printing Subsystem and information on how to use the 3800.

Creating and Updating a 3800 Named System

A named system must be established (via the NAME3800 macro) in DMKSNT for each system data set capable of image library activation. The purpose of the named system is to contain the 3800 character arrangement tables, copy modifications, graphic modifications, and FCBs. They can then be referenced by name and the data for them obtained from this named system when the file referencing them is about to print on a 3800. The active named system for a particular 3800 is in its RDEVBLOK and can be changed by the START command. See the NAME3800 macro description in "Part 2. Defining Your VM/370 System."

Programs exist to enable you to dynamically change the character arrangement tables, graphic modifications, copy modifications, and FCBs available. With these programs (GENIMAGE and IMAGELIB), and the named system support discussed above the installation can make these changes dynamically, without a VM/370 system load. GENIMAGE and IMAGELIB are described in detail in the _VM/370 Operator's Guide_.

3850 Mass Storage System

Generating a VM/370 System that Supports a 3850

The 3850 Mass Storage System (MSS) supplies large amounts of data online under system control. Up to 472 billion bytes of data space becomes available, allowing the user to place significant amounts of tape and DASD shelf data under direct control of the system. 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 Mass Storage System.

HARDWARE SUPPORTED

Support for the 3850 is available on the following processors supported by VM/370: System/370 Models 145, 145-3, 148, 155II, 158 (attached processor and uniprocessor mode), 165II, the 168 (attached processor and uniprocessor mode), the 3031 (attached processor and uniprocessor), 3032 and 3033 processors and the 4331 and 4341 processors.

The major hardware components of MSS are as follows:

- The 3851 Mass Storage Facility (MSF)
- The 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
- The 3333 Disk Storage and Control (Models 1 or 11)
- 3330 Disk Storage Drives (Models 1, 2, or 11)
- 3350 Disk Storage Drives (Real Only)

The Mass Storage Control (MSC) is a microprogrammed processor that provides the operational control for the components of the Mass Storage System. It is physically housed in the 3851 Mass Storage Facility. The MSC may have four System/370 channel interface positions, referred to as A, B, C, and D. A host system attaches to one of these through a control unit position of either 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 the 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 the MSC. These operating systems may be running in a virtual machine under VM/370, or in a real processor, connected to the same MSC as VM/370. 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 Mass Storage System.

The Mass Storage System uses the 3333 control unit and the 3330 Model 1, 2, or 11 for staging data and for holding the tables it requires for its operation. These units connect to the Mass Storage Facility and to the processor through a Staging Adapter. The several models of the 3330 may be intermixed on the Staging Adapter. The 3330 disk drives can be one of the following:

1. Real
2. Staging
3. Convertible

Real DASD drives are not available to the Mass Storage System for any activity. They are physically part of the system in that they have a data and control path through a Staging Adapter, but real drives are not logically connected to the Mass Storage System. Staging drives are used to hold data staged from mass storage volumes to be available for processing by the processor. Staging packs are divided into pages of storage. Each page consists of eight cylinders. The term virtual volume is used to refer 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 either real or staging drives, but not both at the same time. If the drive is to be made 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 and 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 the MSC. The transfer of data between the staging space and the Mass Storage Facility, is also under the control of the MSC. The MSC 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/370 system, the addresses known to VM/370 are the MSC's channel interfaces and the device addresses to the channel interface positions on the Staging Adapter. The MSC is supported in VM/370 only as a dedicated device. For a virtual machine to access the 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 referred to as 3330V device addresses. There are 64 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, and with the proper programming support, may be used for all purposes that a 3330-1 volume is used except VM/370 system residence, paging, and spooling.

3330V devices may be used in three different ways in VM/370:

- Mounted on the device and used as VM/370 system volumes (excluding system residence, paging and spooling) under the control of the control program.
- 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/370 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 the MSC. The MSC is supported as a dedicated device under VM/370 and full MSC support is contained in OS/VS1 and MVS. Therefore, to mount and demount 3330V volumes for VM/370 use, the control program 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. One or all CMS users may access CMS minidisks on MSS volumes. One MSS 3330V volume may contain the 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/370 system volumes is accomplished by the control program communicating with an OS/VS system in a virtual machine. There is an MSS communication program named DMKMSS which is part of the VM/370 system, but which runs in supervisor state in an OS/VS1 or MVS system. This DMKMSS program is the interface between the VM/370 control program and the MSC support contained in OS/VS. The steps to install DMKMSS in an OS/VS system are listed in the section "Generating CP and CMS Using the Starter System" later in this publication.

It is not necessary to generate a VS operating system specifically for the virtual machine environment. Any OS/VS1 or MVS system that supports the MSS can utilize VM/370 MSS support, and can act as the host for the communicator program. There is, however, a requirement for the 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 that are not online. When one is found, an order is issued to the MSC for demount. In essence, the 3330V address is passed to the MSC and the order tells the MSC to demount any volumes currently mounted on that 3330V.

A 3330V may be offline to a virtual machine because none of VM/370's 3330Vs were allocated to the virtual machine at that virtual address. However, the 3330V may be a valid address to the MSC. If the virtual machine issues a demount order to one of these 3330V devices, a volume in use by VM/370 or another virtual machine MSC can be demounted.

Therefore, the following rule must be used when defining (via 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/370 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 (via IOGEN) in the OS/VS system running in CPUID 22222.

SPECIAL CONSIDERATIONS FOR THE VS1/VS2 CENTRAL SERVER VIRTUAL MACHINE

At detach time, the VM/370 control program destages changed cylinders on a volume when the use count for the entire volume reaches zero. The destage function is accomplished by a relinquish order to the MSS through the central server. A relinquish order is issued at detach time for volume-IDs mounted on SYSVIRT and VIRTUAL virtual unit addresses which have had a volume mounted on them by VM/370 on behalf of the guest operating system. No data are destaged for VIRTUAL units that were not mounted by VM/370.

The following VS1/VS2 APARs must be applied to the central server virtual machine operating system when VM APAR 11344 (relinquish function) is applied to the VM/370 control program. The following APARs should be applied regardless of whether the new function is desired:

	COMPONENT			
	SC1BZ	SC1CI	SC1DP	SC1DR
VS1 APAR	OX27455	OX27456	OX27453	OX27454
VS1 SPE BASE	UX90058	UX90059	UX90054	UX90056
VS1 SPE MSSE	-	-	UX90055	UX90057
VS2 APAR	OZ49650	OZ49655	OZ49642	OZ49643
VS2 SPE BASE	UZ90134	UZ90135	UZ90130	UZ90132
VS2 SPE MSSE	-	-	UZ90131	UZ90133

VM/370 APAR 11342 permits general use volume sharing on 3330 virtual unit addresses between a VM/370 system and a native VS1/VS2 system when the unit control blocks are not generated in the VS1/VS2 central server virtual machine. The following VS1/VS2 APARs must be applied to the central server virtual machine operating system when this function is desired:

VS1	APAR	OX24117
	PTF	UX15678
VS2	APAR	OZ48289
	PTF	UZ33530

Note: If general use volume sharing is not desired, these APARs do not have to be applied.

DEFINING THE MSS COMMUNICATION DEVICE

The VM/370 control program initiates 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
```

THE MASS STORAGE CONTROL TABLES

This topic is provided for those installations that 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 the MSS only upon request from VM/370, 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 intend to run more than one MSS virtual machine or to run VS MSS jobs in the MSS communication virtual machine).

The MSC is driven from tables that reside on DASD. These tables are used, among other things, to define the MSS configuration. This configuration includes such items as the addresses to be used for all components of the system, and the available paths from all connected hosts to all these component devices. Thus, 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 S/370 channel attachment to the Staging Adapter).

When a virtual machine is given access to the MSS, one interface to the MSC is dedicated to that virtual machine. To the MSC, this is the same as having that interface connected to a native processor. Thus, the MSC tables must be constructed so that the MSC can process requests from the virtual machine. The MSC must treat the requests as if they came from a native processor, controlling the other components of the MSS such that MSS activity, as seen by VM/370 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/370's 3330V 250 is dedicated to the virtual machine as virtual device address 150. When virtual CPUID 12345 issues an order to the 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. Since that device is known by VM/370 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/370 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. Thus, interruptions received from the virtual machine's 150 are received on VM/370'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/370's 250 is the volume demounted.

Two different virtual machines, having the same virtual device addresses can run concurrently under VM/370. 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 multiplexor 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/370 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 or accessed as VM/370 system volumes, they must be initialized as the image of a 3330-1 disk pack. This initialization is accomplished by the use of an OS/VS access method services command called CREATEV. CREATEV is one of several commands that are part of the MSS component of the access method services, which in turn is a standard component of OS/VS1 and OS/VS2. CREATEV can run either under VS running on a native processor, or 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.

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 may be executed from either a native processor or a VS virtual machine. This 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 be used to 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/370 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/370 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.

THE VM/370 RDEVICE MACRO

The 3330V device addresses generated in the VM/370 control program can be used for two purposes: they can have 3330V system volumes containing minidisks mounted on them, or they can be dedicated to a virtual machine. In either case, the control program can dynamically select a specific device to satisfy a request. You must divide the pool of available 3330V devices into two types, one for system volumes and one for dedicated volumes. The FEATURE= operand of the RDEVICE macro is used to first indicate that a device address is a 3330V as opposed to a 3330-1, and second, to indicate the type of 3330V -- system or dedicated.

When coding the RDEVICE macro for a 3330V device address, either FEATURE=VIRTUAL or FEATURE=SYSVIRT must be coded, where:

- VIRTUAL defines a 3330V that may not be used for system volumes. It may be dedicated or attached to virtual machines as a 3330-1 or 3330V.

- SYSVIRT defines a 3330V that is used for VM/370 system volumes. It cannot be dedicated or attached to a virtual machine. MSS volumes that are 3330V, can be mounted on SYSVIRT 3330V devices but cannot be dedicated to a virtual machine by address, nor attached to other than the system.

To specify an alternate control unit on the RDEVICE macro, code:

```
RDEVICE ADDRESS=cuu,DEVTYPE=nnnn,MODEL=n,ALTCU=cuu
```

Figure 12 shows how the real I/O control block structure is coded and logically appears when an alternate control unit is specified.

```
RDEVICE ADDRESS=(340,32),DEVTYPE=3330,MODEL=1,ALTCU=250
RCTLUNIT ADDRESS=340,CUTYPE=3830,FEATURE=32-DEVICE
RCTLUNIT ADDRESS=250,CUTYPE=3830,FEATURE=32-DEVICE
```

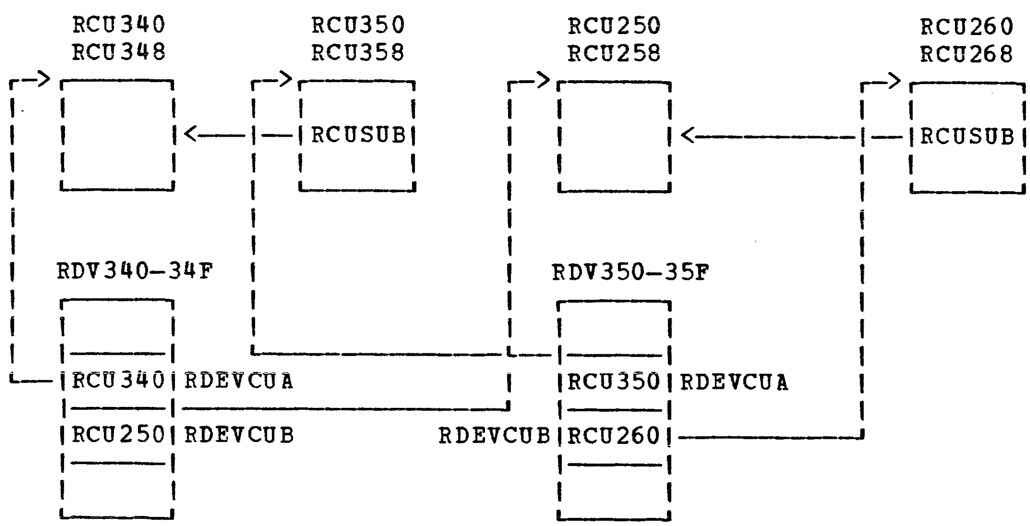


Figure 12. Real I/O Control Block Structure for Alternate Control Unit Specification

To specify alternate channel addresses on the RCTLUNIT macro, code:

```
RCTLUNIT ADDRESS=cuu,CUTYPE=nnnn,FEATURE=xxx-DEVICE,
ALTCH=(1,2,4)
```

Figure 13 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.

```
RCTLUNIT ADDRESS=340,CUTYPE=3830,FEATURE=32-DEVICE,ALTCH=(1,2,4)
RCHANNEL ADDRESS=1,CHTYPE=MULTIPLEXOR
RCHANNEL ADDRESS=2,CHTYPE=MULTIPLEXOR
RCHANNEL ADDRESS=3,CHTYPE=MULTIPLEXOR
RCHANNEL ADDRESS=4,CHTYPE=MULTIPLEXOR
```

```
      RCHAN1          RCHAN2          RCHAN3          RCHAN4
┌──>┌────────┐   ┌──>┌────────┐   ┌──>┌────────┐   ┌──>┌────────┐
│   │        │   │   │        │   │   │        │   │   │        │
│   │        │   │   │        │   │   │        │   │   │        │
│   └────────┘   │   └────────┘   │   └────────┘   │   └────────┘
│                │                │                │
│                │                │                │
│                │                │                │
│                │   RCU340       │   RCU350       │
│                │   RCU348       │   RCU358       │
│                │   ┌────────┐   │   ┌────────┐   │
│                │   │        │   │   │        │   │
│                │   │────────│───┘   │RCUSUB  │   │
│                │   │RCHAN3  │RCUCHA │        │   │
│                │   │────────│       │────────│   │
└────────────────+───│RCHAN1  │RCUCHB │RCU340  │   │
                 │   │────────│       │────────│   │
                 └───│RCHAN2  │RCUCHC │        │   │
                     │────────│       │        │   │
                     │RCHAN4  │RCUCHD │        │   │
                     └────────┘───┐   └────────┘   │
                                  │                │
                                  └────────────────┘
```

Figure 13. Real I/O Control Block Structure for Alternate Channel Specification

Restrictions

The following restrictions apply directly to Alternate Path processing:

- VM/370 does not support alternate paths for devices that issue attention interrupts 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.

Saved Systems

Saved systems are described in detail in the VM/370 System Programmer's Guide. If you plan to save core-image copies of virtual machine operating systems you should do the following when you generate VM/370.

- Create an entry in the system name table for each system you wish to save.
- Reserve space on a CP-owned volume for each system you wish to save.

You create entries in the system name table by coding NAMESYS and NAMENCP macros and assembling the system name table (DMKSNT) file during system generation. You specify which volumes are to be owned by CP by coding the SYSOWN macro and assembling the CP system control (DMKSYS) file during system generation. These macros and files are described in Part 2.

If you decide to add entries to the system name table after you have installed VM/370, you must code the appropriate NAMESYS or NAMENCP macros, reassemble the system name table module (DMKSNT), and reload the CP nucleus. Likewise, if you must add a CP-owned volume after system generation, you must recode the SYSOWN macro, reassemble the CP system control module (DMKSYS), and reload the CP nucleus. Use the GENERATE EXEC procedure to reassemble DMKSNT and/or DMKSYS and to reload the CP nucleus. GENERATE is described in "Part 5. Updating VM/370."

Discontiguous Saved Segments

VM/370 supports discontiguous shared segments and provides shared segment protection.

With discontiguous saved segment support, you can attach and detach segments of storage to and from your virtual machine. These segments may contain reenterable code that can be shared by many users. Thus, programs that are required sometimes, but not all the time, can be saved and only loaded when they are needed. Also, discontiguous saved segments can be attached to your virtual machine in nonshared mode for testing and debugging.

When in attached processor mode, all shared segments are duplicated. Sufficient storage is obtained to construct duplicate page and swap tables in contiguous storage. This additional storage space should be planned for, when running in attached processor mode.

The SHRTABLE SHRPAGE pointer points to the page and swap tables for the main processor, and the page and swap tables for the attached processor will be at a fixed offset from the page and swap tables for the main processor. DMKCFG initializes both sets of page and swap tables. At first, the swap tables for the main processor and attached processor will point at the DASD locations specified in DMKSNT. However, as the pages are read into storage and then stolen, each shared page is allocated its own DASD slot and pointed to by only one swap table entry. The last user to purge a shared system causes both sets of page and swap tables to be freed. See the _VM/370 System Programmer's Guide_ for a description of shared segments.

Segments that are to be saved in this manner must be loaded at an address within your virtual machine and then must be saved. To do this in CMS (following CMS conventions) you must define your virtual machine size large enough to contain the discontiguous segments, loader tables, and CMS control block storage at the end of virtual storage; load the segments; save the segments; then reduce the virtual storage to its normal size. When you attach these segments, they are attached beyond the end of your virtual machine. The procedures for loading and saving discontiguous segments are similar to the procedure that already exists for loading and saving systems.

CMS has three EXEC procedures to help you place portions of CMS in discontiguous saved segments:

- DOSGEN, which loads and saves CMS/DOS support
- VSAMGEN, which loads and saves CMS/VSAM and Access Method Services support
- CMSXGEN, which loads and saves the CMS Editor, EXEC processor, and OS simulation routines

See the section "Loading and Saving Discontiguous Saved Segments" in "Part 3. Generating VM/370 (CP, CMS, RSCS, and IPCS)" for descriptions of how the DOSGEN and VSAMGEN EXEC procedures are used. The CMSXGEN procedure is described in Step 24 of the system generation procedure in Part 3.

CP checks to see whether a virtual machine has altered any shared segments before it dispatches the next virtual machine. When a shared segment is found to have been modified as a result of a CP STORE, ADSTOP, or TRACE command, CP issues a message to indicate that the shared copy has been replaced by a nonshared copy. Execution continues in the virtual machine with the nonshared copy. However, if a protected shared segment is found to be altered by any other means and segment protection is on, CP sends a message to the current virtual machine to identify the altered page. The altered page is made inaccessible and the virtual machine's execution is stopped by placing it into console function mode.

Saved systems must be named and may be shared. The discontiguous saved segment support is similar to saved system support. Therefore, you should understand saved systems before you read this section; see the VM/370 System Programmer's Guide for a description of saved systems.

A discontiguous saved segment is a segment that:

- Has a name associated with it
- Was previously loaded and saved
- May or may not be shared by multiple virtual machines
- Can be loaded by a particular virtual machine in nonshared mode for testing and debugging

A discontiguous saved segment can be logically attached to a virtual machine when it is needed and detached when it is not needed. The attaching and detaching is done by the name associated with the segment. The virtual machine attaching and detaching discontiguous saved segments must issue CP DIAGNOSE instructions to perform the proper linkage. Discontiguous saved segments are loaded at the same address at which they were saved: this address must be higher than the highest address of the virtual machine that is attaching it. A discontiguous saved segment cannot be attached by a virtual machine executing in the virtual=real area.

An example of a discontiguous saved segment is the segment of CMS that supports DOS program development and testing under CMS. This segment is reentrant and is named CMSDOS. The starter system includes an EXEC procedure, DOSGEN, that helps you load and then save this segment. CMS contains all the necessary linkage to load the CMSDOS segment when it is needed.

The main advantage of placing the CMS support for DOS in a discontiguous saved segment is that it conserves real storage. Not all CMS users need the DOS support, and those who do need it probably do not need it all the time. CP keeps the segment tables in real nonpageable storage. These segment tables have an entry for each segment (whether it is saved or nonsaved) of virtual storage available to each active virtual machine. By putting the DOS support in a discontiguous saved segment (called CMSDOS), real nonpageable storage is conserved. Your segment table has entries for the CMSDOS segment (and all segments up to it) only when the CMSDOS segment is attached to your virtual machine.

Using Discontiguous Saved Segments

To use discontiguous saved segments you must:

- Allocate permanent space on a CP-owned volume to contain the saved segment.

- Assign a name to the segment and specify where it is to be stored on disk. To do this, define an entry in the system name table (DMKSNTBL) with the NAMESYS macro. See "Coding the NAMESYS Macro" in "Part 2. Defining Your VM/370 System." Or you can use the entries in the DMKSNT module supplied with the starter system.

- Load and save the segment, using the appropriate EXEC procedure (CMSXGEN, DOSGEN, or VSAMGEN).

- Be sure that the proper linkage for attaching and detaching discontiguous saved segments is in the operating system that needs the segment. CMS contains the linkage necessary to attach and detach the discontiguous saved segments it supports.

Usually, the direct access storage space is allocated and the system name table entries are created during system generation. You allocate DASD space as permanent (PERM) by executing the Format/Allocate program. This program is executed during system generation, but it is a standalone program that can be executed at any time. During system generation, you designate the CP-owned volumes by coding the SYSOWN macro of the DMKSYS file. The system name table (DMKSNT) is also created during system generation. If, at some time after system generation, you wish to change the DMKSYS or DMKSNT files, you can do a partial system generation and reassemble those files using the GENERATE EXEC procedure. GENERATE is described in "Part 5. Updating VM/370."

You can load and save a discontiguous saved segment any time after system generation.

Special Considerations for Using the Editor, EXEC Processor, and OS Simulation Routines

By the time you complete the VM/370 system generation procedure, the CMS editor, EXEC, and OS simulation load modules exist on the CMS S-disk. Also, if you have followed VM/370 recommendations, you have created a discontiguous saved segment, called CMSSEG, that contains the CMS editor, EXEC, and OS simulation routines (you save CMSSEG in Step 24).

During virtual machine execution, CMS handles a call to the editor or EXEC processor as follows:

- CMS first searches for editor or EXEC load modules on all accessed CMS disks, except the S-disk.

  Note: If you wanted to test changes made to the editor or EXEC processor, you could place the load modules on a disk other than the S-disk (that is available only to your virtual machine) and test those changes.

- CMS next attempts to attach the shared segment. If you have not reset the name of the shared segment by issuing a SET SYSNAME command, CMS attempts to attach the CMSSEG segment. If you wish to use an alternate segment, indicate the alternate segment on a SET SYSNAME command and issue that command before the segment is attached. If you do not want CMS to attach a shared segment when editor, EXEC, or OS simulation routines are needed; issue a SET SYSNAME command specifying as the segment name any name that does not correspond to a named saved segment.

- Last, CMS attempts to load the appropriate modules from the CMS S-disk.

CMS handles a call for OS simulation routines in a similar manner. CMS first attempts to attach the named saved segment. Again, you can indicate an alternate segment for loading or avoid loading a named saved segment by specifying a nonexistent segment as the alternate. If a named saved segment is not available, CMS searches all accessed disks for the OS simulation load modules and loads them into high user storage when they are found. The routines are kept in storage until CMS is reloaded or until a SET SYSNAME command is issued for CMSSEG.

Note that there is overhead associated with controlling saved segments and ensuring their integrity. In small systems, the overhead associated with using the CMSSEG saved segment may not be offset by the benefits of sharing storage among users. Therefore, each installation must decide whether the use of CMSSEG is appropriate for its own environment.

Attached Processor Systems

To produce an attached processor system it is necessary to reply "YES" when asked

ARE YOU GENERATING AN AP SYSTEM?--RESPOND (YES|NO)

by the GENERATE or VMSERV EXECs. A response of "YES" will cause DMKRnA CNTRL and APLOAD (or APVRLOAD for a system with a virtual=real area) to be used in place of DMKRn0 CNTRL and CPLOAD (or VRLOAD) EXECs.

DMKAMAC MACLIB

DMKAMAC MACLIB contains one member, OPTIONS COPY, which is identical to OPTIONS COPY in DMKMAC MACLIB except that the variable "&AP" is set to 1, causing AP support to be included in the module you are assembling. The DMKRnA CNTRL file uses this MACLIB and creates a TXTAP rather than the usual TEXT, if necessary; that is, the module is affected by attached processor support.

Modules Containing AP Support

| To find the modules that have attached processor support TXTAP decks, use the following procedure:

| 1. List all modules from the Release 6 source tape that have a filetype of TXTAP.

| 2. List all modules from the system PUT that have a filetype of TXTAP.

| 3. Combine the lists. You should then have a complete list of the TXTAP decks which include all modules containing AP support.

Six modules have been created for AP support. The nucleus-resident modules are DMKEXT, DMKIOK, and DMKMCT. The pageable modules are DMKAPI, DMKCLK, and DMKCPU. These modules have only a TXTAP and their names are contained only in the AP loadlists (APLOAD and APVRLOAD).

Estimating VM/370 Storage Requirements

This section contains information about:

- Estimating real storage requirements for VM/370
- Reducing the size of the CP nucleus
- Estimating direct access storage requirements
- Estimating storage requirements for CMS minidisks

The "Specifying a Virtual=Real Machine" section includes information about estimating real storage requirements for a virtual=real machine.

Note: The requirements specified here are applicable only to the SCP, not to its extensions.

Real Storage Requirements for CP

Figure 14 lists the various CP requirements and the amount of real storage required for each.

CP Requirement	Real Storage Allocated
Resident nucleus	Approximately 152K
Internal trace table	Conventionally, 4K of storage is allocated for each 256K of real storage. This storage is set aside at IPL time. See "SYSCOR Macro" in Part 2 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: • 88 bytes/real device • 72 bytes/real control unit • 96 bytes/real channel • 24 bytes for each remote 3270 or real 3704/3705
Permanently allocated free storage (virtual control blocks and tables). For installation control of free storage, use the SYSCOR macro. See "Part 2. Defining Your VM/370 System."	The default value is a minimum of 12K, plus an additional 4K for each 64K of real storage above 256K.[1] 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.

Figure 14. Real Storage Requirements for CP Requirements (Part 1 of 2)

[1]An additional 25% of free storage is allocated in AP mode.

CP Requirement	Real Storage Allocated
	The storage required is: • 504 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 • 56 bytes/virtual device • 40 bytes/virtual control unit • 40 bytes/virtual channel

Figure 14. Real Storage Requirements for CP Requirements (Part 2 of 2)

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
320K of virtual storage

you would estimate CP real storage requirements as follows.

152K	for the CP resident nucleus
16K	for the CP internal trace table (see "SYSCOR Macro")
4K	for the real control blocks, calculated as follows: 88 X 29 = 2552 bytes for the real devices 72 X 6 = 432 bytes for the real control units 96 X 3 = 288 bytes for the real channels the sum is: 2552 + 432 + 288 = 3272 bytes (approximately 4K)
60K	for permanently allocated free storage (default value)
232K	real storage required

Also, as each of the 12 virtual machines defined logs on, approximately 2K of real storage is allocated to each from the permanently allocated free storage.

```
 504 bytes for a VMBLOK
  64 bytes for the SEGTABLE
 200 bytes for the PAGTABLE
 680 bytes for the SWPTABLE
  56 bytes for a virtual reader
  56 bytes for a virtual printer
  56 bytes for a virtual punch
 168 bytes for three virtual disks
 120 bytes for three virtual channels
  56 bytes for a virtual machine console
 120 bytes for three virtual control units
----
2080 bytes for each of the logged-on users defined
```

The number of virtual devices for a virtual machine cannot exceed the value determined by (7FFFF/VDEVSIZE), where VDEVSIZE is the size of the VDEVBLOK. If a greater number of virtual devices is specified, results may be undesirable.

See the "Specifying the Amount of Virtual=Real Space" section for an example of estimating storage requirements and determining the maximum size of the virtual=real area.

Reducing the Size of the CP Nucleus

Support for the 3340, 3704, 3705, 3066, 3850, and 3270 increases the size of the CP nucleus. 3340 support is handled by the module DMKTRK. The 3704/3705 is primarily handled by the module DMKRNH. 3850 Mass Storage System support is provided in module DMKSSS. The graphic device support for locally attached terminals is handled by the module DMKGRF while the remote 3270 support is handled by the DMKRGA, DMKRGB, and DMKBSC modules. Each of these modules occupies space in the system nucleus.

This nucleus area can be reclaimed by deleting DMKTRK, DMKRNH, DMKMSS, DMKGRF, DMKRGA, DMKRGB, and DMKBSC from the system loadlist EXEC file. Caution should be exercised before deleting them from the loadlist. If you generate a system which includes 3340 disks in the I/O configuration, you cannot delete the module DMKTRK. If you generate any type of locally attached graphic device in the DMKRIO assemble file, you cannot delete the module DMKGRF. Or, if you generate remote 3270s in the DMKRIO assemble file, you cannot delete the DMKRGA, DMKRGB, and DMKBSC modules. Module DMKSSS cannot be deleted if you are using a 3850. In addition, if you generate the 3704/3705 in the DMKRIO assemble file, you cannot delete the DMKRNH module.

The following names are undefined during the VMFLOAD procedure if DMKTRK is deleted from the loadlist:

```
DMKTRKIN    DMKTRKFP    DMKTRKVA
```

The following names are undefined during the VMFLOAD procedure if DMKGRF and DMKRNH are deleted from the loadlist:

```
DMKGRFEN    DMKRNHCT    DMKRNHND
DMKGRFIC    DMKRNHIC    DMKRNHTR
DMKGRFIN    DMKRNHIN
```

The following names are undefined during the VMFLOAD procedure if DMKBSC, DMKRGA, and DMKRGB are deleted from the loadlist:

DMKBSCER   DMKRGBIC
DMKRGBEN   DMKRGAIN

The following names are undefined during the VMFLOAD procedure if DMKSSS is deleted from the loadlist:

DMKSSSHV   DMKSSSMQ   DMKSSSI1   DMKSSSEN   DMKSSSVA
DMKSSSAS   DMKSSSL1   DMKSSSL2   DMKSSSL3   DMKSSSDE

If you generate your system without the V=R option, the module DMKVSC can be deleted from the loadlist with no undefined symbols.

Direct Access Storage Requirements for CP

Figure 15 shows how much DASD space CP requires by DASD type. The following paragraphs describe in detail how you determine the amount of DASD space CP requires for the nucleus, error recording, warm start data, checkpoint data, directory, saved systems data, paging, and spooling space.

	2319 2314	3330	2305	3340	3350
CP Nucleus	varies	varies	varies	varies	varies
Error Recording[1]	2	2	2	2	2
Warm Start	1	1	1	1	1
Checkpoint Start	2	1	3	3	1
Directory	2	2	2	2	2
Saved Systems	varies	varies	varies	varies	varies
Paging Space	32	18	40	40	10
Spooling Space	50	30		70	15
Total System[2]	93 cyl	57 cyl	53 cyl	123 cyl	33 cyl

Figure 15. DASD Space Requirements by DASD Type

CP NUCLEUS DASD REQUIREMENTS

The CP nucleus (without a virtual=real area) currently requires about 115 pages of disk space for resident and pageable functions.

To determine the number of cylinders required for the CP nucleus, refer to the load map produced during system generation. One DASD page is required for each page of fixed and pageable nucleus (for a CP nucleus without a virtual=real area). The calculations for the amount of DASD space needed for a CP nucleus with a virtual=real area are in the "Specifying a Virtual=Real Machine" section.

----

[1]The default is 2 cylinders but up to 9 cylinders may be specified via the SYSERR operand of the SYSRES macro.
[2]These figures do not include space for the nucleus or saved systems.

For example, if the last module entry in the load map is at page 55 (hexadecimal), 85 pages of disk space are required for CP nucleus residence. 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.

Normally, the number of cylinders required for CP nucleus residence is:

6 cylinders on a 2305 or 3340
5 cylinders on a 2314 or 2319
3 cylinders on a 3330 or 3333
2 cylinders on a 3350

ERROR RECORDING DASD REQUIREMENTS

Error recording space is variable (from 2 to 9) and is established by the SYSERR operand of the SYSRES macro instruction.

WARM START DATA DASD REQUIREMENTS

Formulas for calculating the amount of warm start space needed are in "Part 2. Defining Your VM/370 System" under the discussion of the SYSWRM operand of the SYSRES macro.

CHECKPOINT START DATA DASD REQUIREMENTS

The amount of space required for the dynamic checkpointing of the VM/370 spool file system is discussed in "Part 2. Defining Your VM/370 System" under the description of the SYSRES macro.

VM/370 DIRECTORY DASD REQUIREMENTS

The VM/370 directory normally requires two cylinders so that it can be rewritten without disturbing the active directory and swapped after a successful update. Equations for computing directory sizes are found in the "Allocating DASD Space for the VM/370 Directory" section of Part 2.

SAVED SYSTEM DASD REQUIREMENTS

Saved systems require one page for each page saved, plus an additional information page. However, a 3704/3705 may require up to four additional information pages.

To save one copy of the CMS system requires two cylinders on a 2314, 2319, or 3340, or one cylinder on a 3330, 3333, or 3350.

PAGING AND SPOOLING DASD REQUIREMENTS

Paging and spooling space requirements are installation-dependent. (The values shown in the preceding list are for average systems.) Paging space is allocated at a rate of:

24 pages/cylinder on a 2305 or 3340
32 pages/cylinder on a 2314 or 2319
57 pages/cylinder on a 3330 or 3333
120 pages/cylinder on a 3350 in native mode
(The 2305 is normally used for paging only.)

Spooling data is placed in a 4K-byte buffer with the necessary channel programs required for each record. Data capacity of spooling cylinders thus varies with the data and CCWs used.

The primary system operator is warned when the paging/spooling space becomes 90% full. The VM/370 System Messages manual tells the operator what he should do if this warning occurs.

VSAM AND ACCESS METHOD SERVICES REQUIREMENTS

The VSAM and access method services support in CMS requires both DASD space and virtual storage.

The amount of DASD space needed is listed in Part 3, in the section, "Loading and Saving the CMSVSAM and CMSAMS Segments."

The VSAM and access method services support adds approximately 2K to the size of the CMS nucleus. In addition, this support uses free storage to execute the DOS/VS logical transients and for buffers and work areas. VSAM issues a GETVIS macro to request free storage.

If the CMS/DOS environment is invoked with the VSAM option

```
set dos on (vsam
```

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

IPCS Requirements

IPCS supports the same basic VM/370 processor configurations that are supported by other components of VM/370 with a minimum of 384K of real storage. This is the basic VM/370 requirement.

EXTERNAL STORAGE

The disk storage needed by IPCS is divided into two parts. The first part does not vary greatly (only problem reports and symptom summary are affected). It contains the IPCS command modules, the current NUC MAP, problem reports, and the symptom summary. These files occupy less than 5 cylinders on a 3330, allocated as shown:

- 100 problem reports plus symptom summary 20%
- NUC MAP 45%
- IPCS modules 35%

The second part contains the dumps. The size of a dump depends mainly on the size of the system being dumped, and the operand of the CP SET DUMP command, either ALL or CP. The table below shows typical space usage by device type for the fixed area and for one dump.

Files	Cylinders			
	3330	2319	3340	3350
IPCS modules NUC MAP 100 problem reports symptom summary	5	9	14	3
512K CP	1.5	3	4	1
512K ALL	2.5	5	7	1.5
1024K CP	3	6	8	1.5
1024K ALL	6	11	16	3

REAL STORAGE

The real storage requirement is the normal CP requirement of approximately 2K for control blocks while the IPCS virtual machine is logged on. Other real storage usage is controlled by the VM/370 demand paging implementation.

Estimating DASD Storage Requirements for CMS

The following information is intended to help you allocate sufficient direct access storage space for CMS minidisks.

A 2314 cylinder formatted by the CMS FORMAT command contains 150 800-byte blocks, which can contain approximately 1300 80-byte lines of source programs and data.

A 3330 cylinder formatted by the CMS FORMAT command contains 266 800-byte blocks, which can contain approximately 2300 80-byte lines of source programs and data.

A 3340 cylinder formatted by the CMS FORMAT command contains 96 800-byte blocks, which can contain approximately 960 80-byte lines of source programs and data.

A 3350 cylinder formatted by the CMS FORMAT command contains 570 800-byte blocks, which can contain approximately 5000 80-byte lines of source programs and data.

Each 800-byte block contains file control information as well as your data. A given amount of data requires more file 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:

- 7 cylinders of 2314 space (for approximately 9100 80-byte lines of source programs and data)
- 4 cylinders of 3330 space (for approximately 9600 80-byte lines of source programs and data)
- 11 cylinders of 3340 space (for approximately 10560 80-byte lines of source programs and data)
- 2 cylinders of 3350 space (for approximately 9120 80-byte lines of source programs and data.

Minidisks

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

Therefore, if you do not require the full capacity of a real DASD, you can be assigned one or more minidisks instead. A minidisk is a logical subdivision of a physical disk pack with its own virtual device address, virtual cylinders (starting with 0, 1, 2, and so on) and a VTOC (volume table of contents or disk label identifier). Each of your minidisks is preallocated the number of contiguous full cylinders that were specified in the VM/370 MDISK directory record, and that space is considered to be a complete virtual disk device.

Minidisks are controlled and managed by CP. If a virtual machine attempts to use DASD space beyond the boundaries defined for its minidisks, CP presents a command reject (seek check) to the virtual machine. If a system is to be run on both a virtual and a real machine, minidisks for that system must start at real cylinder zero. For a detailed list of minidisk restrictions, see "Appendix F: VM/370 Restrictions."

The remainder of this section describes the following characteristics of minidisks:

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

Defining Minidisks

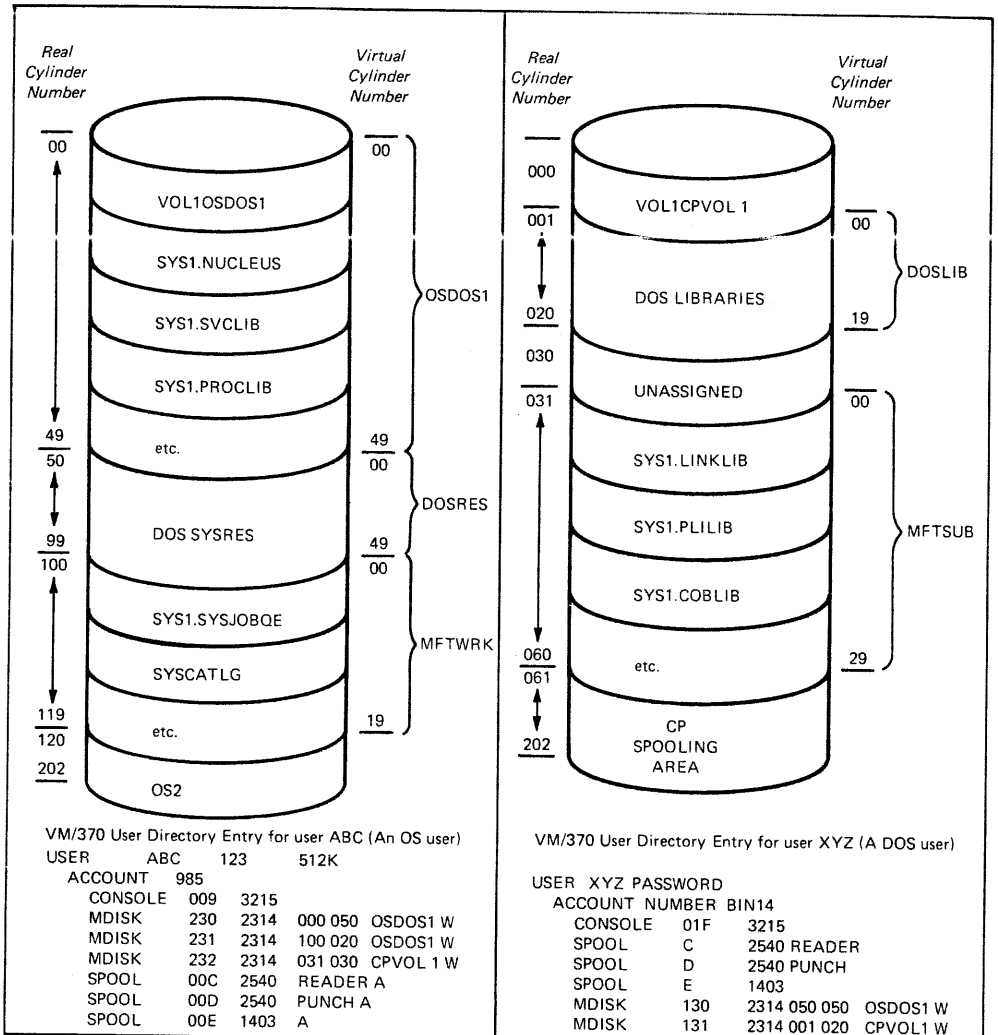
Permanent minidisks are defined in the VM/370 directory entry for a virtual machine. A minidisk defined in the directory via an MDISK statement is a permanent part of the virtual machine configuration and the data on the minidisk is available to the user from session to session.

If any virtual machine has a temporary requirement for direct access space, this can be filled from a pool of T-disk space. You specify the size of the T-disk pool when you allocate disk space with the standalone Format/Allocate program. Minidisks created from the T-disk area must be initialized and are available to the virtual machine for the duration of one terminal session. When the virtual machine logs off or issues a CP command to release the temporary minidisk, the area is returned to CP.

It is up to you to allocate minidisks on VM/370 disks in a manner that minimizes arm contention and physical overlap. Information about minimizing arm contention is found in the "Preparing the CP System Control File (DMKSYS)" section of Part 2.

<u>Note</u>: The VM/370 directory function neither checks nor flags overlapped or duplicate minidisk extents. Nor does the function provide DASD space records for unused space or used space.

Figure 16 illustrates the use and definition of minidisks. The disk labeled OSDOS1 contains several minidisks, some formatted to OS requirements and others to DOS requirements. OSDOS1 is a 2314 volume. The directory entry for userid ABC (an OS user) describes the virtual device 230 as a 50-cylinder area, and the virtual device 231 as a 20-cylinder area on real volume OSDOS1. The directory entry for userid XYZ (a DOS user) describes the virtual device 130 as a 50-cylinder disk area on a real volume OSDOS1.



Note: VM/370 allows cylinders 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 16. Use and Definition of Minidisks

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

Note: On a 3330, 3340, or 3350, an OS/VS, or OS minidisk must start at real cylinder 0 unless the VTOC is limited to one track. See the list of restrictions in "Appendix F: VM/370 Restrictions" for more information and explanation of 3330/3340/3350 restrictions.

Minidisk Space Allocation

OS bases all of its space allocation parameters on the volume table of contents (VTOC) label written on each disk; it determines the amount of space available on that volume from the format-5 (space accounting) data set control block (DSCB). Thus, for OS to support minidisks, a VTOC must be written whose format-5 DSCB reflects the desired size of the minidisk. The remainder of the disk space on the real disk appears to OS to be permanently dedicated, and not assignable by the OS space accounting routines. The IBCDASDI service program should be used to format minidisks for use by OS or DOS.

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

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

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

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

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

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

Track Characteristics

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

- For all CMS disks except CMS/VSAM disks, the CMS FORMAT command formats the specified tracks into 800-byte blocks or physical records.
- For CP disks, the standalone CP Format/Allocate program must be used to format specified tracks into 4096-byte blocks.
- For OS, DOS, and CMS/VSAM minidisks the IBCDASDI service program writes read-only track descriptor records for each track, and clears the remaining portion of each track to binary zeros. It also writes a format-5 DSCB whose contents reflect the minidisk size (the amount of free space available for allocation). Any disk initialization program that supports the operating systems use of the DASD device type may be used if you are initializing full disks.

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

Alternate Tracks

3330/3350 DISKS

Alternate tracks assigned at the factory or by IBCDASDI in the field are automatically handled on the 3330 or 3350 by the control unit. Minidisks on the 3330 Model 1 or 2 should be specified on cylinder 0 through cylinder 403 only. The remaining cylinders (404 to 411) are automatically used by the 3830 Control Unit for alternate tracks. Minidisks on the 3330 Model 11 can be specified on cylinder 0 through cylinder 807. Minidisks on the 3350 should be specified on cylinder 0 through cylinder 554 only. The remaining cylinders (555 to 559) are automatically used by the 3830 control unit for alternate tracks.

3340 DISKS

The 3340 DASD device uses a hardware logic that lessens the dependence on alternate track usage. The 3340 can bypass the defective portion of a data track and write the balance of the record in the space remaining. In the case where an alternate track is required, the alternate track can be assigned by IBCDASDI standalone using a dedicated 3340 device. Cylinder 348 on the 3348 Data Module, Model 35 and cylinders 696 and 697 on the 3348 Data Module, Model 70 are reserved for this purpose. Once IBCDADSI has assigned the alternate track, the disk, including the cylinder containing the defective track, may be used for any purpose whatever, including CP system residence, CMS minidisks, and so forth. There are only two restrictions:

- A minidisk should not be located where its track 0, cylinder 0 falls on a defective track because then it will be impossible for the CP IPL command to function for that minidisk.
- Any operating system doing SIO to this disk must be capable of doing the normal alternate track error recovery.

<u>Note</u>: CMS qualifies here because it uses DIAGNOSE in place of SIO.

Error Recovery Support

When an attempt to do I/O on a defective 3340 or 3344 track results in a track condition check, software error recovery procedures provide for switching to an alternate track. For CP I/O and for diagnose I/O issued from a virtual machine, the switching is fully automatic and the issuer of the I/O request is not aware of it. For SIO issued from a virtual machine, a track condition check is reflected to the virtual machine so that the operating system in the virtual machine will run its own error recovery procedures.

Since alternate tracks are assigned from the high-order cylinders at the end of the real 3340, the virtual machine will attempt to seek outside of the minidisk to recover. The VM/370 CCW translation process allows seeks outside of the minidisk to an alternate track provided that the particular alternate track is assigned to a defective track within that minidisk. After seeking to the alternate track, any attempts at head switching to an unowned track in this cylinder are prevented.

3340 Cylinder Assignments

On 3340-35 devices, the primary data area is cylinder 0-347. Cylinder 348 is reserved for alternate tracks. On 3340-70 devices or 3344 devices, the primary data area is cylinder 0-695. Cylinders 696-697 are reserved for alternate tracks.

Allocation Conversion at Release 5 PLC 6

Previously, the "alternate tracks" cylinders of 3340/3344 devices were often used as primary data cylinders, but now these cylinders must be reserved exclusively for alternate track use. Therefore, when changing from an old system (prior to Release 5 PLC 6) to a current system, it is necessary to revise the space allocation and minidisk layouts on any 3340/3344 disk where the "alternate tracks" cylinders had been used as a primary data area.

System Residence Devices: If the system residence device contains "alternate tracks" cylinders that have been used as the primary data area, the files of existing control statements should be revised prior to generating a new system. In particular, the allocate function performed on the system residence disk and other CP-owned disks may have to be revised, and subsequent to this revision, the specification of the SYSRES, NAMESYS, and NAMENCP macros should be reviewed.

Minidisk Devices: If any minidisks on a 3340/3344 extend into the alternate tracks cylinders, they can be copied to another area of the disk or to another disk using the DASD dump restore (DDR) utility. In the past, when a 3340/3344 had a defective track, the cylinder with the bad track was unusable and minidisks would be allocated adjacent to that cylinder, but not including it. In this case, all cylinders of the real disk should be dumped to tape using any version of the DDR utility.

If you use the new version of the DDR utility and the alternate tracks cylinders have been used as a primary data area, make sure that you specify the cylinder range explicitly. For example, enter:

```
DUMP 0 TO 697
```

rather than specifying ALL, which no longer dumps anything from the final cylinders except tracks that have been assigned as alternates. Then you can execute the IBCDASDI utility to assign alternate tracks to the defective tracks so that all cylinders become usable. Subsequently, the new DDR utility can be used to restore minidisks from the tape, possibly reordering them into the previously unusable cylinders.

Note: Whenever a minidisk is moved to a new location or its size is changed, the corresponding MDISK statements in the system directory must be revised.

Only the new versions of the DDR, DIR, and FMT utilities should be used with 3340/3344 devices after alternate tracks have been assigned.

Starter System Changes: In release 6 the starter system has been changed to reserve cylinder 348 for alternate track use. Therefore, the 3340 starter system can be restored to a disk that has defective tracks (provided that alternate tracks have already been assigned by IBCDASDI).

2314/2319 DISKS

On 2314 and 2319 devices, CP and CMS (except CMS/VSAM) do not recognize or support alternate track techniques for their own use. DOS, OS, and CMS/VSAM minidisks, however, do recognize and support alternate tracks on these types of DASD. The IBCDASDI service program automatically assigns the last cylinder in any minidisk as an alternate track cylinder. When you initialize 2314/2319 devices, you can assign all 203 cylinders for virtual machine and system use.

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

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

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

- Set up the entire volume containing the defective track as an OS, DOS, or CMS/VSAM volume and format it with either IBCDASDI or IEHDASDR for OS or CMS/VSAM disks, or with the DOS Initialize Disk utility program (INTDK) for DOS disks. Alternate tracks are assigned in the standard manner.

Labels

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

```
VOL1xxxxxx

 -- or --

CMS=xxxxxx
```

where xxxxxx is a 6-character volume label.

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

VOL1xxxxxx

where xxxxxx is a 6-character volume label.

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

In Figure 16, the volume indicated as OSDOS1 has its real cylinder 0 allocated to a minidisk that is formatted for use by OS. The volume serial number of that minidisk must be OSDOS1, the label that is associated with the real volume. Since the minidisk label identifies the physical volume, changing it affects the directory entries of all users who have minidisks on that volume.

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

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

Sharing Minidisks

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

For example, assume a virtual machine called USERA owns a minidisk at address 150. The VM/370 directory entry for USERA contains the following statement:

```
MDISK 150 3330 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, or has the following LINK statement in its VM/370 directory entry, can read the 150 minidisk belonging to USERA.

```
LINK USERA 150 cuu RR
```

The cuu is the virtual device address at which the 150 minidisk belonging to USERA is linked to another virtual machine. If you define another virtual machine, USERB, with the following statement in its VM/370 directory entry:

```
LINK USERA 150 151 RR
```

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

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

Minidisks may be shared in the following ways:

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

Note: See the description of the CP LINK command in the *VM/370 CP Command Reference for General Users* for more information about linking to minidisks.

Configurations

Before you begin the system generation procedure, make sure your installation has the minimum configuration supported by VM/370 and the features and facilities required by VM/370.

VM/370 Minimum Configuration

The minimum configuration supported by VM/370 is:

One	Processor (393,216 bytes of storage)
One	System Console device
One	Printer
One	Card Reader
One	Card Punch
Two	Spindles of Direct Access Storage
One	Nine-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 "Estimating VM/370 Storage Requirements."

| A representative VM/370 configuration is:

IBM	4341	2Mb/4Mb Storage
IBM	3278	Display Console Model 2A
IBM	3203	Printer Model 5 -- Two
IBM	3350	Direct Access Storage Model A2 -- Four drives attached to a 3880 Storage Control Model 1
IBM	2305	Fixed Head Storage Facility, Model 2
IBM	3420	Magnetic Tape Units -- Two
IBM	3705	Communications Controller
IBM	3277	Display Stations (as needed) with the 3272 Control Unit (local attachment) or with the 3271 Control Unit (remote attachment)
IBM	3278	Display Stations (as needed) with 3274 Control Unit (local attachment or remote attachment).

Configurations Supported by CMS

CMS supports the following configurations:

- Virtual storage size: minimum of 320K bytes, up to 16 million bytes in multiples of 4K.
- Virtual console: any terminal supported by VM/370 as a virtual machine operator's console.
- The same unit record devices (card readers, punches, and printers) supported by VM/370 as spooling devices, except the 2520 Punch. See "Unit Record Devices".
- Up to ten logical 2314, 2319, 3340, 3330 Model 1, 2, or 11, 3333 Model 1 or 11, or 3350 direct access storage devices. The maximum size of a CMS minidisk is:

No. of Cylinders		
CMS/VSAM	Non-VSAM	Device Type(s)
200	203	2314/2319 (the entire disk)
404	246	3330/3333 Model 1 or 2
808	246	3330/3333 Model 11
348	348	3340 Model 35
696	682	3340 Model 70
555	115	3350 (in native mode)

- Up to four 2400, 2415, 2420, 3410 (9 track only), or 3420 (7 or 9 track) Magnetic Tape Units.

Configurations Supported by RSCS

RSCS supports the following configurations:

- Virtual storage size: Minimum of 512K, up to 16 million bytes in multiples of 4K.
- Virtual console: any terminal supported by VM/370 as a virtual machine operator's console.
- Any virtual card readers, punches, and printers supported by VM/370 as spooling devices.
- One logical 2314, 2319, 3330 Model 1, 2, or 11, 3333 Model 1 or 11, 3340, or 3350 direct access storage devices.
- Transmission Control Units: 2701 Data Adapter Unit; 2703 Transmission Control Unit; or 3704 or 3705 Communications Controllers in EP mode only. Only binary synchronous communication transmission is supported.

The minimum configuration supported by RSCS is:

- 512K virtual storage
- One console
- One Reader
- One Transmission Control Unit
- One or more binary synchronous lines dedicated to the RSCS virtual machine

Devices Supported by VM/370

The following devices are supported by VM/370 except as otherwise noted. The devices are listed by device type:

- Processors
- Direct access storage devices
- Magnetic tapes
- Unit record devices (printers, readers, and punches)
- Terminals
- Transmission control units and communications controllers
- Remote spooling devices
- Other devices

Processors

VM/370 supports the following processors:

- IBM System/370 Model 135 Submodel 3
- IBM System/370 Model 138
- IBM System/370 Model 145
- IBM System/370 Model 145 Submodel 3
- IBM System/370 Model 148
- IBM System/370 Model 155 II
- IBM System/370 Model 158 UP/AP/MP[1]
- IBM System/370 Model 158 Submodel 3
- IBM System/370 Model 165 II
- IBM System/370 Model 168 UP/AP/MP[1]
- IBM System/370 Model 168 Submodel 3
- IBM 4331 processor
- IBM 4341 processor
- IBM 3031 processor UP/AP
- IBM 3032 processor UP
- IBM 3033 processor UP/AP/MP[1]

PROCESSOR REQUIRED FEATURES AND FACILITIES

The processor features and facilities required by VM/370 are listed below. Only the features and facilities that are not standard on a particular processor are described. 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.

- The System Timing facility (#2001), which includes the clock comparator and the processor timer, on the Models 135 and 145.

- The clock comparator and processor timer (#2001) on the Model 145-3.

- The Floating-point feature

  --For the Model 135, feature #3900
  --For the Model 145, feature #3910

- The Extended Precision Floating feature (#3840) on the Model 135-3.

- The Channel Indirect Data Addressing feature on each of the 2860, 2870, and 2880 standalone I/O channels on the Model 165 II or 168.

  --For the 2860, features #1861, 1862, and 1863

  --For the 2870, feature #1861

  --For the 2880, features #1861 and 1862

----

[1]This System/370 model is supported when running in uniprocessor mode or with an asymmetric I/O configuration. In an asymmetric configuration, all I/O devices attached to the system must be attached to one processor.

Note: The standalone 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 for proper operation of the input/output channels in a Dynamic Address Translation environment.

- The Word Buffer feature (#8810) is required on the System/370 Model 145-3. It is also required on the Model 145 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 also recommended for selector channels if 2314, 3330, 3340, or 3350 devices are attached.

DESIRABLE FEATURES

The following processor features are desirable for VM/370:

- Virtual machine assist improves the performance of VM/370 systems that run virtual storage operating systems in virtual machines. The manner in which virtual machine assist and VM/370:ECPS (see below) are supported on the various VM/370 processors is detailed under "Using the Performance Options."

- Extended Control - Program Support improves the performance of VM/370 through CP assist and expanded virtual machine assist capabilities.

- The Extended Floating-point feature, although not required, improves the execution of programs that use Extended Floating-point instructions under VM/370 on Models 135, 155 II, and 158.

  --For the Model 135, feature #3840
  --For the Model 155 II, feature #3700
  --For the Model 158, feature #3700

- The APL Assist feature provides performance assistance when used with the VS APL program product. It is available as hardware feature #1005 on the System/370 Models 135 and 145.

- The Conditional Swapping feature provides additional instructions required for the execution of VTAM programs. It is available as feature #1051 on the System/370 Models 135 and 145.

- The Advanced Control Program Support feature is available only on the System/370 Model 145 as feature #1001. It provides additional instructions required for the execution of MVS (OS/VS2 Release 2 and above) and/or VTAM.

  Note: The Conditional Swapping feature and the Advanced Control Program Support feature are mutually exclusive.

Direct Access Storage Devices

The following direct access storage devices and control units are supported by VM/370.

The direct access storage devices supported by VM/370 are:

| • IBM 2305 Fixed Head Storage, Models 1 and 2.

- IBM 2314 Direct Access Storage Facility.
- IBM 2319 Disk Storage.
- IBM 3330 Disk Storage, Models 1, 2, and 11.
- IBM 3333 Disk Storage and Control, Models 1 and 11.
- IBM 3340 Direct Access Storage Facility, Models A2, B1, and B2; the 3348 Data Modules, Models 35, 70, and 70F; and the 3344 Direct Access Storage, Model B2.
- IBM 3350 Direct Access Storage, Models A2 and B2.

All of these direct access devices are supported as VM/370 system residence, paging and spooling devices and as virtual devices for use by virtual machines. All are supported as dedicated devices. All except the 2305 are supported by CMS.

The following direct access control units are supported by VM/370:

- IBM 3345 Storage and Control Frame Models 3, 4, and 5 on the Models 145, 145-3, and 148 with the standard ISC for:

  --3330 Models 1 and 2
  --3333 Models 1 and 11
  --3340 Model A2 and 3344 Model B2
  --3350 Model A2

| • IBM 2835 Storage Control Model 1 for 2305 Model 1.

- IBM 2835 Storage Control Model 2 for 2305 Model 2.
- IBM 2844 Auxiliary Storage Control for 2314 and 2319.
- IBM 3830 Storage Control Model 1 for 3330 Models 1 and 2 only.
- IBM 3830 Storage Control Model 2 for 3333 Models 1 and 11, 3340 Model A2, and 3350 Model A2.
- IBM 3830 Storage Control Model 3 for 3330 Models 1 and 11, and 3333 Models 1 and 11.
- IBM 3880 Storage Control Model 1 for 3330 Models 1, 2, and 11, 3333 Models 1 and 11, and 3350 Models A2 and A2F.
- IBM Integrated File Adapter (#4650) on System/370 Models 135 and 145 for 2319.

- IBM Integrated File Adapter (#4655) on the System/370 Models 135, 135-3, and 138 or the IBM Integrated Storage Control #4660) on the System/370 Model 145, 145-3, and 148 for:

  --3330 Models 1 and 2
  --3333 Models 1 and 11
  --3340 Model A2 and 3344 Model B2
  --3350 Model A2

  Note: VM/370 does not support any Integrated File Adapters (IFAs) that support more than 64 addresses.

- IBM Integrated Storage Control on the System/370 Model 158 for:

  --3330 Models 1 and 2
  --3333 Models 1 and 11
  --3340 Model A2 and 3344 Model B2
  --3350 Model A2

- IBM Integrated Storage Control on the System/370 Model 168:

  --3330 Models 1 and 2
  --3333 Models 1 and 11
  --3340 Model A2 and 3344 Model B2
  --3350 Model A2

- IBM 3333 Disk Storage, Models 1 and 11 for the 3330 Models 1, 2, and 11.

- IBM 3340 Disk Storage, Model A2, and 3344 Model B2.

- IBM 3350 Disk Storage, Model A2.

Special Features Required with the 3350

Expanded Control Store special feature (#2151) provides additional control storage for microprogramming use and is a prerequisite 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.

The Control Store Extension feature (#2150) is a prerequisite for feature #2151.

Notes:

1. The IBM 3330 Model 11 can be used as a system generation device in the same way as the 3330 Models 1 and 2, since the starter system does not use cylinders 404-807.

2. The System/370 Models 145, 145-3, and 148 must have the Word Buffer feature (#8810) installed in order to attach a 3330, 3340, 3350 or 2305 Model 2.

Magnetic Tapes

The following magnetic tape devices and control units are supported by VM/370.

The magnetic tape devices supported are:

- IBM 2401, 2402, and 2403 Magnetic Tape Units
- IBM 2415 Magnetic Tape Units, Models 1, 2, 3, 4, 5, and 6
- IBM 2420 Magnetic Tape Units, Models 5 and 7
- IBM 3410/3411 Magnetic Tape Unit, Models 1, 2, and 3
- IBM 3411 Magnetic Tape Unit and Control, Models 1, 2, and 3
- IBM 3420 Magnetic Tape Units, Models 3, 4, 5, 6, 7, and 8

The magnetic tape control units supported are:

- IBM 2803 Tape Control
- IBM 2804 Tape Control
- IBM 3411 Magnetic Tape Unit and Control
- IBM 3803 Tape Control

Unit Record Devices

VM/370 supports the following printers, readers, punches, and unit record control units as system spool devices.

VM/370 supports the following printers:

- IBM 1403 Printer Models 2, 3, 7, and N1
- IBM 1443 Printer Model N1 (with 144 print positions)
- IBM 3203 Printer Model 4 (available on processor Models 138 and 148 only) and Model 5
- IBM 3211 Printer (Right Indexing only)
- IBM 3800 Printer (complete dedicated device support; limited spool device support)

VM/370 supports the following readers/punches:

- IBM 2501 Card Reader Models B1 and B2
- IBM 2520 Card Punch Models B2 and B3
- IBM 2540 Card Read Punch Model 1
- IBM 3505 Card Reader Models B1 and B2
- IBM 3525 Card Punch Models P1, P2, and P3

VM/370 supports the following unit record control units:

- IBM 2821 Control Unit
- IBM 3811 Printer Control Unit
- IBM Integrated Printer Adapter (IPA) (on the System/370 Model 135.
- IBM Integrated Printer Adapter Basic Control (#4670), and one of the following on the models 135-3 and 138:

  --1403 Printer Models 2 or N1 Attachment (#4672)
  --1403 Printer Model 7 Attachment (#4677)

- IBM Integrated 3203 Model 4 Printer Attachment, first printer (#8075) and optionally, second printer (#8076) on the Model 138 and 148.

Terminals

The following system consoles are supported by VM/370 as virtual system consoles (simulated as 3215 consoles):

- IBM 2150 Console with 1052 Printer-Keyboard Model 7
- IBM 3066 System Consoles Models 1 and 2 for the System/370 Models 165 II and 168
- IBM 3210 Console Printer-Keyboard Models 1 and 2
- IBM 3215 Console Printer-Keyboard Model 1
- IBM System Console for the System/370 Models 138 and 148 in printer-keyboard mode (3286 printer required) or display mode
- IBM System Console for the System/370 Model 158 in printer-keyboard mode (with the 3213 Printer Model 1 required,) or in display mode
- IBM 7412 Console (via RPQ AA2846) with 3215 Console Printer-Keyboard Model 1
- IBM 3036 Console with the 3031, 3032 or 3033 processor
- IBM 3278 Model 2A Console with the 4331 or 4341 processor

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

The following terminals are supported by VM/370 as virtual system consoles (simulated as 3215 consoles):

- IBM 2741 Communication Terminal
- IBM 1050 Data Communication System
- IBM 3101 Display Terminal, Models 10, 12, 13, 20, 22, and 23 (supported as teletype Model ASR 33/35 teletypewriter)
- Terminals on switched lines compatible with the line control used by the IBM Telegraph Control Type II Adapter (8-level ASCII code at 110 bps) such as the CPT-TWX (Model 33/35) terminals
- IBM 3275 Display Station, Model 2 with integral control unit (remote attachment only)
- IBM 3276 Control Unit Display Station Models 2, 3, and 4[1] with integral control unit (supported as a 3277 attached to a 3271 for remote attachment only)
- IBM 3277 Display Station, Model 2, via 3272 Control Unit, Model 2 (local attachment only)
- IBM 3277 Display Station, Model 2, via 3271 Control Unit, Model 2 (remote attachment only)

---

[1]Models 3 and 4 operate in Model 2 default mode.

- IBM 3278 Display Station Models 2, 3, and 4[1] via 3274 Control Unit Model 1B (supported as a 3277 attached to a 3272 for local attachment only)

- IBM 3278 Display Station Models 2, 3, and 4[1] via 3274 Control Unit Model 1C (supported as a 3277 attached to a 3271 for remote attachment only)

Note: 3215 console simulation for graphics devices excludes processing multiple output channel programs which contain CCW's without carriage returns (X'01' CCW op code) on one line of the screen. These channel programs are treated separately and VM/370 uses a new line for each one.

----
[1]Models 3 and 4 operate in Model 2 default mode.

- IBM 3278 Display Station Models 2, 3, and 4[1] via 3276 Control Unit Display Station Models 2, 3, and 4 (supported as a 3277 attached to a 3271 for remote attachment only)

- IBM 3767 Communications Terminal, Models 1 and 2 (operating as a 2741)

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.

Prior availability of an RPQ does not guarantee or imply current or future availability. Contact your IBM branch office for ordering information concerning the RPQs mentioned with the following features.

<u>2741 Features: Required and Desirable Features</u>

The IBM 2741 Communication Terminal is supported on either duplexed switched or point-to-point nonswitched lines connected to a Western Electric 103A2 (or equivalent data set). The following features are required features:

- PTTC/EBCD (#9571, Part #1167963) or standard Correspondence (#9812, Part #1167043) print elements

- Transmit Interrupt (#7900) or Transmit Interrupt Control RPQ #E40681

- Receive Interrupt (#4708)

- For switched lines, the Data Set Attachment (#9114) and Dialup feature (#3255) are required.

- For point-to-point nonswitched lines, one of the following features is required:

  --Data Set Attachment (#9115 duplexed for facility D1) or
  --Data Set Attachment (#9116 duplexed for facility B2) or
  --Data Set Attachment (#9120 duplexed for facility B1 or D1) or
  --IBM Line Adapter (#4635 for 4-wire limited distance line)
  --IBM Line Adapter (#4691-4694 for 4-wire shared nonswitched line)

  --IBM Line Adapter (#4647 for 4-wire nonswitched line)

The following features, although not required, enhance the convenience and usability of the terminal:

- Print Inhibit (#5501)

- Red Ribbon Control RPQ #868019 (supported for PTTC/EBCD keyboard only)

- Typamatic Keys (#8341)

- Pin Feed Platen (#9509)

----

[1]Models 3 and 4 operate in Model 2 default mode.

1050 Control Units, Models, and Features: Supported, Required, and Desirable Features

The IBM 1050 Data Communication System is supported on either switched or point-to-point nonswitched lines with these features:

- IBM 1051 Control Unit (Model 1 or 2) with these features:

  --Transmit Interrupt (#7900) or Transmit Interrupt Control RPQ #E26903

  --Receive Interrupt (#6100) or Receive Interrupt Control RPQ #E27428

  --Text Time-Out Suppression (#9698)

  --First Printer Attachment (#4408). This feature is required to attach a 1052 Printer-Keyboard to the 1051.

- IBM 1052 Printer-Keyboard (Model 1 or 2) with the PTTC/EBCD print element (#9571, Part #1167963)

- For switched lines, the Data Set Attachment (#9114) is required.

- For point-to-point nonswitched lines, one of the following is required:

  --Data Set Attachment (#9115 for facility D1)

  -- or --

  --Data Set Attachment (#9116 for facility B2)

  -- or --

  --Data Set Attachment (#9120 for facility B1 or D1)

  -- or --

  --IBM Line Adapter (#4691-4694 for 4-wire shared nonswitched line)

  -- or --

  --IBM Line Adapter (#4647 for 4-wire nonswitched line)

The following features, although not required, enhance the convenience and usability of the terminal:

- Automatic Ribbon Shift and Line Feed Select (#1295)
- Automatic EOB on Carrier Return RPQ #E28235

3270 Components, Control Units, Models, and Features: Supported, Required, and Desirable Features

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

- IBM 3272 Control Unit Model 2 for attachment of up to thirty-two 3277 Display stations Model 2, 3284 Printers Model 2, 3286 Printers Model 2, 3287 Printer Models 1 and 2, and 3288 Line Printers Model 2. To support this configuration, the following are required:

  --Device Adapter feature (#3250) is required if more than 4 devices are attached to the 3272. Up to 4 additional devices can be attached with each device adapter.
  --A 3271/3272 Attachment (#8330) is required to attach each 3287 Printer.

- IBM 3274 Control Unit Model 1B (supported as a 3272) for attachment of up to 32 display stations and printers. All of the 32 devices can be 3278 Display Stations Models 2, 3, and 4[1] (supported as 3277s), 3287 Printers Models 1 and 2, (supported as 3284s or 3286s), and 3289 Line Printers Models 1 and 2. (supported as 3288s). A maximum of 16 of the 32 devices can be 3277 Display Stations Model 2, 3284 Printers Model 2, 3286 Printers Model 2, 3287 Printers Models 1 and 2, and 3288 Line Printers Model 2. To support this configuration, the following are required:

  --The basic 3274 Control Unit permits attachment of up to 8 devices (3278, 3287, and 3289). At least one 3278 is required.

  --Each of the Terminal Adapter Types A1, A2, and A3 (#6901, #6902, and #6903) permits the attachment of up to 8 additional devices per adapter (types 3278, 3287, and 3289). Only one of each type terminal adapter is permitted. Terminal Adapter Type A1 is a prerequisite to Type A2, and A2 is a prerequisite to Type A3.

  --Terminal Adapter Type B1 (#7802) permits the attachment of up to 4 additional devices (types 3277, 3284, 3286, 3287, and 3288). Only one adapter is permitted.

  --Each of Terminal Adapter Types B2, B3, and B4 (#7803, #7804 and #7805) permits the attachment of 4 additional devices per adapter (types 3277, 3284, 3286, 3287 and 3288). Only one of each type terminal adapter is permitted. Terminal Adapter Type B1 is a prerequisite to Type B2, Type B2 is a prerequisite to Type B3, and Type B3 is a prerequisite to Type B4. Terminal Adapter Types A1 and B3 are mutually exclusive.

  --A 3274/3276 Attachment (#8331) is required for each 3287 Printer Models 1 or 2 that attaches to the basic 3274 Control Unit, or that attaches to Terminal Adapter Types A1, A2, or A3. A 3271/3272 Attachment (#8330) is required for each 3287 Printer Model 1 or 2 that attaches to Terminal Adapter Types B1, B2, B3, or B4.

  And only needed if a Type B adapter is used:

  --Control Storage Expansion feature (#1801) provides the ability to access control unit storage addresses above 64K.

  --Extended Function Storage feature (#3622) provides additional storage required to support particular attachments or configurations. Control Storage Expansion feature (#1801) is a prerequisite.

---

[1]Models 3 and 4 are supported as Model 2 via hardware default.

The following control units can be remotely attached to leased lines via a 2701 Data Adapter Unit, a 2703 Transmission Control Unit, a 3704/3705 Communications Controller in emulation mode, or an Integrated Communications Adapter (ICA) to support 3270 devices:

- IBM 3271 Control Unit Model 2 for attachment of up to thirty-two 3277 Display Stations Model 2, 3284 Printers Model 2, 3286 Printers Model 2, 3287 Printers Models 1 and 2, and 3288 Line Printers Model 2. To support this configuration, the following may be required:

  --Device Adapter feature (#3250) is required if more than 4 devices are attached to the 3271. Up to 4 additional devices can be attached with each device adapter.
  --A 3271/3272 Attachment (#8330) is required to attach each 3287 Printer Model 1 or 2.
  --Copy feature (#1550) is required to use the VM/370 full-screen copy function.
  --Transmission Speed feature (#7820 or #7821) is required.

- IBM 3274 Control Unit Model 1C (supported as a 3271) for attachment of up to 32 display stations and printers. All of the 32 devices can be 3278 Display Stations Models 2, 3, and 4[1] (supported as 3277s), 3287 Printers Models 1 and 2, (supported as 3284s or 3286s), and 3289 Line Printers Models 1 and 2. (supported as 3288s). A maximum of 16 of the 32 devices can be 3277 Display Stations Model 2, 3284 Printers Model 2, 3286 Printers Model 2, 3287 Printers, Models 1 and 2, and 3288 Line Printers Model 2. To support this configuration, the following are required:

  --The basic 3274 Control Unit permits attachment of up to 8 devices (3278, 3287, and 3289). At least one 3278 is required.

  --Each of Terminal Adapter Types A1, A2, and A3 (#6901, #6902, and #6903) permits the attachment of up to 8 additional devices per adapter (types 3278, 3287, and 3289). Only one of each type terminal adapter is permitted. Terminal Adapter Type A1 is a prerequisite to Type A2, and Type A2 is a prerequisite to Type A3.

  --Terminal Adapter Type B1 (#7802) permits the attachment of up to 4 additional devices (types 3277, 3284, 3286, 3287, and 3288). Only one adapter is permitted.

  --Each of Terminal Adapter Types B2, B3, and B4 (#7803, #7804, and #7805) permits the attachment of 4 additional devices per adapter (types 3277, 3284, 3286, 3287, and 3288). Only 1 of each type of terminal adapter is permitted. Terminal Adapter Type B1 is a prerequisite to Type B2, Type B2 is a prerequisite to Type B3, and Type B3 is a prerequisite to Type B4. Terminal Adapter Types A and B are mutually exclusive.

  --A 3274/3276 Attachment (#8331) is required on the printer for each 3287 Printer Model 1 or 2 that attaches to the basic 3274 Control Unit, or that attaches to Terminal Adapter Types A1, A2, or A3. A 3271/3272 Attachment (#8330) is required on the printer for each 3287 Printer Model 1 or 2 that attaches to Terminal Adapter Types B1, B2, B3, or B4.

  --Copy feature (#1550) is required if you are planning to use the VM/370 full-screen copy function.

----

[1]Models 3 and 4 are supported as Model 2 via hardware default.

--Device Adapter feature (#3250) is required if more than four devices are attached to the 3271. Up to four additional devices can be attached with each Device Adapter.

--Transmission Speed feature (#7820 or #7821) is required.

--Control Storage Expansion feature (#1801) provides the ability to access control unit storage addresses above 64K.

--Extended Function Storage feature (#3622) provides additional storage required to support particular attachments or configurations. Control Storage Expansion feature (#1801) is a prerequisite.

--Each 3274 Model 1C requires a Common Communications Adapter feature (#6302) and an External Modem Interface (#3701).

- IBM 3276 Control Unit Display Station Models 2, 3, and 4[1] (supported as a 3271) for attachment of up to 7 additional 3278 Display Stations Models 2, 3, and 4[1] (supported as 3277s), 3287 Printers Models 1 and 2, (supported as 3284s or 3286s), and 3289 Printers Models 1 and 2 (supported on a 3276). To support this configuration, the following are required:

  --The basic 3276 Control Unit Display Station contains 1 integral display station, (supported as a 3277), and permits attachment of either 1 3278 Display Station Model 2, 3, and 4[1] (supported as a 3277) or 1 3287 Printer Models 1 and 2. (supported as a 3284 or 3286).

  --Terminal Adapter No. 1 (#3255) permits attachment of up to 2 additional devices (3278s, 3287s and 3289s). Only 1 adapter is permitted.

  --Terminal Adapter No. 2 (#3256) permits attachment of up to 2 additional devices (3278s, 3287s and 3289s). Only 1 adapter is permitted. Terminal Adapter No. 1 (#3255) is a prerequisite.

  --Terminal Adapter No. 3 (#3257) permits attachment of up to 2 additional devices (3278s, 3287s and 3289s). Only 1 adapter is permitted. Terminal Adapter No. 2 (#3256) is a prerequisite.

  --A 3274/3276 Attachment (#8331) is required for each 3287 Printer Model 1 or 2.

  --Each 3276 requires one of the Communications features (#6301 or #6302) and either the External Modem Interface (#3701) or the 1200 BPS Integrated Modem feature (#5500).

The following control unit is remotely attached to either leased or switched lines via a 2701 Data Adapter Unit, a 2703 Transmission Control Unit, a 3704/3705 Communications Controller in emulation mode, or an Integrated Communication Adapter (ICA) to support 3270 devices:

- IBM 3275 Display Station MOdel 2 standalone control unit and display station. A 3284 Printer Model 3 or 3286 Printer Model 3 can be attached. Also, the following may be required:

  --For the 3275 to be used on a switched line, Dial feature (#3440) is required.

  --Transmission Speed feature (#7820 or #7821) is required.

----

[1]Models 3 and 4 are supported as Model 2 via hardware default.

--To attach a 3284 Printer Model 3, a Printer Adapter feature (#5550) is required.

--To attach a 3286 Printer Model 3, RPQ MB4317 is required.

The 3275 Display Station (remote attachment) and the IBM 3277 Display Station Model 2 require one of the following features:

--66 Key EBCDIC Typewriter Keyboard (#4630)
--66 Key EBCDIC Data Entry Keyboard (#4631)
--78 Key Operator Console-Keyboard (#4632)
--78 Key EBCDIC Typewriter Keyboard (#4633)

The 3276 Control Unit Display Station and the IBM 3278 Display Station require one of the following features:

--75 Key EBCDIC Typewriter Keyboard (#4621)
--75 Key EBCDIC Data Entry Keyboard (#4622)

Note: A prerequisite is the EBCDIC Character Set feature (#9082) on the 3276 or 3278.

The following features, while not required, enhance the convenience and usability of the terminals:

--Keyboard Numeric Lock (#4690) - All terminals
--Audible Alarm (#1090) - All terminals
--Operator Identification Card Reader (#4600) - 3275 and 3277 only
--Security Keylock (#6340) - All terminals
--Magnetic Slot Reader (#5005) with Magnetic Reader Control (#4999) - 3278 only
--Lowercase Character Display (RPQ #8K0366) - 3275 and 3277 only

3270 APL Support

The following 3270 devices are supported by VM/370's 3270 APL support:

- IBM 3271 Model 2 or IBM 3272 Model 2 Control Unit

- IBM 3277 Display Station, Model 2, with either of the following APL keyboards:

APL Keyboard	Feature
66-character keyboard	#4637
78-character keyboard	#4638

  Note: The 78-character keyboard includes the 12 Program Function keys, which are required if you wish to use the local copy function.

- IBM 3284 Printer, Model 2, the IBM 3286 Printer, Model 2, or the IBM 3287 Printer, Models 1 and 2.

The 3270 Data Analysis feature (#1066) allows the use of APL with any or all of the above devices.

Note: The 3270 Data Analysis feature is field-installable. Since lowercase display support is included in the APL hardware feature, the Lowercase Display RPQ #8K0366 must be removed prior to installing the APL feature.

Devices Not Supported

The following 3270 devices (although supported by VM/370) are not supported by the 3270 Data Analysis-APL feature and therefore cannot be used to display, print, or enter APL characters:

- The 3284 Printer, Model 3
- The 3275 Display Station, Model 2
- The 3288 Printer, Model 2
- The 3286 Printer, Model 3
- The 3274 Control Unit, Models 1B and 1C
- The 3276 Control Unit Display Station, Models 2, 3, and 4
- The 3278 Display Station, Models 2, 2A, 3, and 4
- The 3289 Line Printer, Models 1 and 2

Note: With the 3270 Data Analysis-APL feature installed, all standard 3270 functions remain operational. The only exception is the backtab key, which is not supported by VM/370.

3270 Support for Text Processing

VM/370's support of the 3270 text feature allows the user to key in, as well as display and print, all the special characters.

The 3270 Data Analysis-APL feature (#1066) is required on the following 3270 devices if they are to be used with the special text characters:

- 3271 Control Unit, Model 2
- 3272 Control Unit, Model 2
- 3284 Printer, Model 2
- 3286 Printer, Model 2
- 3287 Printer, Models 1 and 2

The 3277 Display Station Model 2 must be equipped with the text keyboard (#4639). For a more detailed description of VM/370's text support, see the VM/370 Terminal User's Guide.

3270 Devices Not Supported by the Text Feature

The following 3270 devices, although supported by VM/370 for other uses, do not support the text feature:

- 3274 Control Unit, Models 1B and 1C
- 3275 Display Station, Model 2
- 3276 Control Unit Display Station, Models 2, 3, and 4
- 3278 Display Station, Models 2, 2A, 3, and 4
- 3284 Printer, Model 3
- 3286 Printer, Model 3
- 3288 Printer, Model 2
- 3289 Line Printer, Models 1 and 2

3767 Features: Required and Desirable Features

The IBM 3767 Communication Terminal, Models 1 and 2, is supported when it operates as an IBM 2741 Communication Terminal and is attached to a 3704 or 3705 Communications Controller. It requires the following features on either switched or nonswitched point-to-point lines:

- 2741 START/STOP (#7113)
- EBCDIC (#9391) or Correspondence (#9381) keyboard
- Duplexed, switched or nonswitched line (#9404) for connecting to a Western Electric 103A2 (or equivalent data set)
- One of the following:

  --EIA Interface with Clock (#3719) at 300 bps
  --1200 bps Integrated Modem/Interrupt (#5505 or #5500 or #5506)

The following features, although not required, enhance the convenience and usability of the terminal:

- Alternate Character Set (#1291), plus a defined character subset for the keyboard:

  --If the primary character set is Correspondence (#9381), the alternate character set can be APL (#9383) or EBCDIC (#9382).

  --If the primary character set is EBCDIC (#9391), the alternate character set can be APL (#9393) or Correspondence (#9392).

  Note: Line control is PTTC/EBCD with this feature.

- Acoustic Coupler (#1110) at 300 bps

Transmission Control Units

VM/370 supports the following transmission control units:

- IBM 2701 Data Adapter Unit
- IBM 2702 Transmission Control
- IBM 2703 Transmission Control
- IBM Integrated Communications Attachment (ICA), (#4640)
- IBM 3704, 3705-I, and 3705-II Communications Controllers

2701 REQUIRED FEATURES

- For line control of CPT-TWX (Model 33/35) terminals and the 3101 display terminals, the Telegraph Adapter Type II (#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 (#7698)
  --EBCDIC code (#9060)
  --EBCDIC transparency (#8029)

- For 1050 and 2741 terminals, the following are required:

  --IBM Terminal Adapter Type I, Model II (#4640)
  --Selective Speed, 134.5 bps (#9581)
  --2741 Break Feature RPQ #M53193, and Break Command RPQ #858492

- The Expanded Capability feature (#3815) is required if there are:

  --More than two low speed adapters (either IBM 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.

- The Expansion Feature (#3855) is required for each line adapter after the first.

2702 REQUIRED AND OPTIONAL FEATURES

- For 1050 and 2741 terminals, the following are required:

  --Terminal Control Base for IBM Terminal Control (#9696)

  --IBM Terminal Control Type I (#4615)

  --Selective Speed, 134.5 bps (#9684)

  --Type I Terminal Interrupt (#8200)

  --Data Set Line Adapter (#3233) or IBM Line Adapter (#4635), 4-wire IBM Terminal Control Type I (#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 (#9697)

  --Telegraph Terminal Control Type II (#7912)

  --Pluggable End Characters (return key generates an interrupt) RPQ #E62920, optional

  --Data Set Line Adapter (#3233)

  --Terminal Control Expansion (#7935), required only if both of the terminal bases (#9697 and #7912) are attached to the same 2702.

- The 31 Line Expansion (#7955) is supported as needed.

2703 REQUIRED AND OPTIONAL FEATURES

- For 1050 and 2741 terminals, the following are required:

  --Start-Stop Base Type I (#7505) or Type II (#7506)
  --IBM Terminal Control Base (#4619)
  --IBM Terminal Control Type I (#4696)
  --Line Speed Option, 134.5 bps (#4878)
  --Type I Terminal Interrupt (#8200)
  --Data Line Set (#3205) and/or IBM Line Set 1B (#4687)

- For line control of CPT-TWX (Model 33/35) terminals and 3101 display terminals, the following are required:

  --Telegraph Terminal Control Base (#7905)

  --Telegraph Terminal Control Type II (#7912)

  --Line Speed Option, 110 bps (#4877)

  --Data Line Set (#3205), and Data Line Set Expander (#3206)

  --Pluggable End Characters (return key generates an interrupt) RPQ #E66707, optional

- For 2770, 2780, and 3780 Terminals, the following are required:

  --Synchronous Base (#7703, 7704, or 7706)
  --Synchronous Terminal Control for EBCDIC (#7715)
  --Transparency (#9100)
  --Synchronous Line Set (#7710)

- The Base Expansion feature (#1440) is required if more than one base type is to be attached to the same 2703.

IBM INTEGRATED COMMUNICATIONS ATTACHMENT (ICA) REQUIRED AND OPTIONAL FEATURES

The ICA (#4640) is available on the System/370 Models 135, 135-3, and 138. Additional lines (#4722-4728) are supported.

- For 1050, 2741, and 3767 (as a 2741) terminals, the following are required:

  --Terminal Adapter Type I Model II (#9721-9728)

  --Switched Network Facility (#9625-9632), optional

  --Write Interrupt (#9745-9752)

  --Read Interrupt (#9737-9744)

  --Unit Exception Suppression (#9729-9730), optional

  --For the 3767 only, as a 2741, 200 bps (#2711-2718) or 300 bps (#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 (#9649-9656)
  --Half-Duplex Facility (#9617-9624)
  --EBCDIC Transparency (#9673-9680)

- For line control of CPT-TWX (Model 33/35) terminals and 3101 display terminals, the following are required:

  --Telegraph Adapter Type II (#9785-9792)
  --Switched Network Facility (#9625-9632)

3704/3705 REQUIRED FEATURES

The IBM 3704 and 3705 Communications Controllers are supported in Network Control Program mode, Partitioned Emulation Program mode, and 2701, 2702, 2703 Emulation Program mode.

Note: VM/370 supports the CPT-TWX (Model 33/35) terminals at 110 bps and the 3101 display terminals at 110, 150, 300, and 600 bps, when attached to a 3704 or 3705.

VM/370 supports all models of the 3704 and 3705 Communications Controllers. However, because the network control program and partitioned emulation program require 48K of storage, the following models are supported for the emulation program only.

- IBM 3704 Communications Controller Models A1 and A2
- IBM 3705-I Communications Controller Models A1, B1, C1, and D1
- IBM 3705-II Communications Controllers, Models E1, F1, G1, and H1.

The features required on a communications controller do not depend on VM/370. VM/370, when supporting network control mode, requires a communications controller with at least 48K of storage. Other 3704/3705 features depend upon the intended use of the communications controller and the type of 3704/3705 control program (emulation, network control, or partitioned emulation program) to be executed.

VM/370 does not support the following 3704/3705 features:

- Line Set Type 2A (#4721)
- Line Set Type 3A (#4731)
- Line Set Type 4B (#4742)

Remote Spooling Devices Supported by VM/370

Remote spooling is supported in VM/370 by the Remote Spooling Communications Subsystem (RSCS).

REMOTE SPOOLING COMMUNICATIONS SUBSYSTEM (RSCS)

The VM/370 Remote Spooling Communications Subsystem (RSCS) supports the following:

- IBM 2770 Data Communication System
- IBM 2780 Data Transmission Terminal, Models 1 and 2
- IBM 3770 Data Communication System (as a 2770)
- IBM 3780 Data Communication Terminal
- HASP supported programmable workstations

2770 Features: Required and Optional Features

The IBM 2770 Data Communication System with the 2772 Multipurpose Control Unit can be connected to the central System/370 via a switched or nonswitched point-to-point communication line.

Configurations (Remote Spooling)

The following devices and features are required for operating a 2770 as an RSCS nonprogrammable terminal:

- One IBM 2213 Printer, Model 2, or one IBM 2203 Printer, or one IBM 1053 Printer

- One IBM 2502 Card Reader, Model A1 or A2

- EBCDIC Transmission Code (#9761)

Other supported equipment and features are:

- One IBM 545 Card Punch, Model 3 or 4, with or without 3590 attachment
- EBCDIC Transparency (#3650)
- Additional Buffer Expansion (#1491)
- Space Compression/Expansion (#6555)
- Synchronous Clock (#7705)

2780 Features: Required and Optional Features

The IBM 2780 Data Transmission Terminal, Models 1 and 2, can be connected to the central System/370 via a switched or nonswitched point-to-point line. EBCDIC Transmission code (#9762) is required.

The following features are optional:

- EBCDIC Transparency (#8030)
- 120/144 Character Print Line (#5820 or #5821)
- Multiple Record Transmission (#5010)
- Synchronous Clock (#7705)

3770 Features

The IBM 3770 Data Communication System can be connected to the central System/370 via a switched or nonswitched point-to-point communications line. Required features are:

- EBCDIC Transmission Code (#9761)
- SDLC/BSC Switch Control (#1460), or BSC point-to-point (#1461)

3780 Features: Required and Optional Features

The IBM 3780 Data Communications Terminal can be connected to the central System/370 via a switched or nonswitched point-to-point communications line. EBCDIC transmission code (#9761) is required.

The following devices and features are optional:

- One IBM 3781 Card Punch
- Component Selection (#1601, required for the 3781)
- EBCDIC Transparency (#3601)
- Additional Print Positions (#5701)
- Synchronous Clock (#7705)

Programmable Terminals

RSCS Spool MULTI-LEAVING[1] (SML) supports, as a VM/370 remote workstation, any processor that is supported as a HASP workstation and is programmed to operate as a HASP workstation.

Processors Supported

- The IBM 1130 Computing System (except Models 4A and 4B) is supported if it has at least 8K words of storage and the Synchronous Communications Adapter (#7690).
- Any System/360 or System/370 is supported if it has at least 8K bytes of main storage and a 2701, 2703 or 3704/3705 in EP mode, equipped for EBCDIC transmission and binary synchronous communications.
- Any submodel of the System/360 Model 20 or the IBM 2922 is supported if it has at least 8K bytes of main storage and the following features:

  --Binary Synchronous Communications Adapter (#2074)
  --EBCDIC Transmission code (#9060)
  --Full Transparency (#4100)

- IBM System/3 Models 6, 8, 10, 12, and 15 are supported with the following features:

  --Binary Synchronous Communications Adapter (#2074)
  --EBCDIC Transmission code (#9060)
  --Text transparency (#7850)

- IBM System/32 is supported with the following features:

  --5320 System Unit (any model A12 through B33)
  --Binary Synchronous Communications Adapter (#2074)
  --System Control Program (5725-SC1)

----
[1]Trademark of IBM

Other Considerations for Planning Your Configuration

TWO-CHANNEL SWITCH

If any I/O devices controlled by VM/370 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.

See the "VM/370 Using Channel Switching" section earlier in Part 1 for more information about using the two-channel switch.

DEVICES USED ONLY BY AN OPERATING SYSTEM IN A VIRTUAL MACHINE AND NOT BY VM/370

Any input/output device that can be attached to the IBM System/370 can be used by a virtual machine under VM/370 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.
- None of the other restrictions outlined in "Appendix F: VM/370 Restrictions" are violated.

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

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

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

Service Record File

On 3031, 3032, and 3033 processors, each console station of the 3036 system console has a 7443 diskette attached to it, accessible when the console station is in SRF mode. In the normal console configuration, one of the processor's console stations is used as an operator's console, and the other console station is used as a service console. It is through the service console that SRF capability is provided. With the console in degraded configuration (i.e. one console station serves as both operator and service console), there is no SRF capability. Thus, it is recommended that the SRF address specified on the RIOGEN macro instruction at system generation be the address of the service record file attached to the service console.

MULTIPLE SERVICE RECORD FILES

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

3033 attached processor environments support multiple service record file devices. For VM/370 systems operating on a 3033 AP, you should specify multiple SRF devices at system generation. Code DEVTYPE=7443 in the RDEVICE macro statement and CUTYPE=7443 in the RCTLUNIT macro statement 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 statement as SRF=(cuu,cuu,...). 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.

In 3033 multiprocessing environments with I/O configured asymmetrically to one processor, in order to access the SRF devices in both 3036 consoles, a channel path must be available from the I/O processor to both SRF devices.

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

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

Part 2. Defining Your VM/370 System

Part 2 describes the macros and control statements you need to define your VM/370 system. It contains the following sections:

- Introduction
- Preparing the Real I/O Configuration File (DMKRIO)
- Preparing the CP System Control File (DMKSYS)
- Creating Your VM/370 Directory
- Preparing the System Name Table File (DMKSNT)
- Altering the Forms Control Buffer Load (DMKFCB)

Introduction

Before starting the system generation procedures on a real machine, you must create three files that describe the VM/370 system you are generating. There are two additional files, which are optional. You can use card input, or create these files using the CMS Editor. If you are modifying an existing VM/370 system, you can use the CMS Editor to alter the existing files.

The three files that you must prepare are:

- The real I/O configuration file (module name DMKRIO), which defines the I/O configuration on the real System/370 machine.
- The CP system control file (module name DMKSYS), which defines CP-controlled DASD volumes, allocation, and so on.
- The VM/370 directory file (normally a CMS file named USER DIRECT), which contains the VM/370 directory entries that define the virtual machine configuration for each user.

In addition, you should prepare the system name table (DMKSNT) file if you plan to save systems. If you generate the 3704/3705 control program, you must save it; otherwise, the 3704/3705 control program cannot be loaded by VM/370. "Part 1. Planning for System Generation" has a section that describes the requirements for saving systems.

You can also change the forms control buffer load (module name DMKFCB).

The notational conventions that describe the macro syntax for VM/370 system generation are listed in "Appendix D: Notational Conventions." These notational conventions are the same as the conventions used to describe VM/370 commands.

Preparing the Real I/O Configuration File (DMKRIO)

The real I/O configuration file consists of macros that describe the I/O devices, control units, and channels attached to the real System/370. VM/370 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 your installation has so that devices can be added in the future without performing another system generation, but bear in mind that the control blocks generated (RDEVBLOK, RCUBLOK, and RCHBLOK) occupy space in real storage.

When preparing the RDEVICE and RCTLUNIT entries, refer to "Appendix B: Configuration Aid" 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 macros, in their proper sequence, are:

```
Macro Name    Units Referred To
(CLUSTER )    Remote Display Stations
(TERMINAL)
(   .    )
(   .    )
(   .    )
RDEVICE       I/O Devices
RCTLUNIT      Control Units
RCHANNEL      Channels
RIOGEN        System Console
```

The file is placed in the reader as follows:

```
DMKRIO CSECT
       CLUSTER macro¹
       TERMINAL macro¹
           .
           .
       RDEVICE macros
           .
           .
       RCTLUNIT macros
           .
           .
       RCHANNEL macros
           .
           .
       RIOGEN macro
       END
```

----

[1]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 precede all the other real I/O configuration macros. See the special requirements for the TERMINAL macros for devices attached to the 3274 Model 1C in the section on "Coding the Real I/O Configuration Macros for Remote 3270s."

All the 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.

Coding the Real I/O Configuration Macros for Remote 3270s

Two types of remote 3270 configurations are supported: a cluster control unit 3271 with multiple terminals and printers attached and standalone display stations. The clustered configurations attach to either a 3271, 3274 Model 1C, or 3276 control unit, all of which are coded as a 3271. The standalone station is a 3275 display station which contains its own built-in control unit. All remote configurations are attached via 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 unique label. This label is coded on the RDEVICE macro that defines the corresponding binary synchronous line and thus 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 (attached via 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 via Terminal Adapter Types B1, B2, B3, or B4), or 3288s attached, directly following the CLUSTER macro, first code TERMINAL macros for all 3278s, 3287s (attached via 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, 3287s (attached via 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 the 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

Use the CLUSTER macro to define a control unit associated with a remote 3270. Each CLUSTER macro represents a display control unit (a 3271, 3274 Model 1C, or 3276, all specified as a 3271) on a leased BSC line, or a standalone 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 precede all other macros in the DMKRIO file.

The format of the CLUSTER macro is:

Name	Operation	Operands
label	CLUSTER	CUTYPE={3271 | 3275} ,GPOLL=cudv ,LINE=cuu ,DIAL={YES | NO}

where:

label
: is a name of the CLUSTER macro; it must be specified. The label may be any valid assembler language symbol. The label establishes a unique symbolic name for this cluster control unit or standalone station.

CUTYPE={3271 | 3275}
: is the control unit of the station; it is either 3271 or 3275.

  Note: Code a 3274 or 3276 as a 3271.

GPOLL=cudv
: 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 that the control unit determines is 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 control unit of the station. 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 the remote 3270 device (dv) are always X'7F' and the general polling characters for the control unit are defined when the control unit is physically installed. Use Figure 17 to determine what you should code as the general polling characters for the control unit. GPOLL is ignored if CUTYPE=3275 and DIAL=YES are specified.

Note: The 3274 and 3276 terminals have control unit address switches which are set by the user to match the polling and selection address characters shown in Figure 17.

LINE=cuu
is the line interface address. It is the address specified on the RDEVICE macro that is associated with this CLUSTER macro.

DIAL={YES | NO}
specifies whether the 3275 has the Dial feature. DIAL=NO must be specified if CUTYPE=3271.

Examples

The following CLUSTER macro describes a 3271, 3274, or 3276 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 precede the TERMINAL macros that define the stations attached to that cluster or standalone station.

TERMINAL Macro

Use the TERMINAL macro to define (1) a display station or printer that is attached to the remote 3270 display system or (2) 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 standalone 3275 display station.

Code one TERMINAL macro for each display device and each printer attached to a cluster control unit (3271 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.

For a 3274 Model 1C Control Unit that has 3277s, 3284s, 3286s, 3287s (attached via Terminal Adapter Types B1, B2, B3, or B4), or 3288s attached, first you must code a TERMINAL macro for all 3278s, 3287s, and 3289s in groups of 8 until all 3278s, 3287s, and 3289s have been included. You must code a TERMINAL macro for every terminal address in each group of 8. Then, following these macros, you must 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 standalone station (3275). 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.

Note: All the TERMINAL macros defining the devices attached to a remote 3270 station must follow the CLUSTER macro that defines the control unit for that station. The groups of CLUSTER and TERMINAL macros must precede all other macros in the DMKRIO file.

The format of the TERMINAL macro is:

Name	Operation	Operands
	TERMINAL	TERM= { 3275 | 3277 | 3284 | 3286 | 3288 } ,SELECT=cudv [,<u>MODEL=2</u> | ,MODEL=3] [,FEATURE=OPRDR]

where:

TERM= {3275 | 3277 | 3284 | 3286 | 3288}

is the device type of the remote 3270 station attached to the clustered or standalone 3270 control unit. The following devices are allowed:

Device	TERM=
IBM 3275 Display Station	3275
IBM 3276 Display Station	3277
IBM 3277 Display Station	3277
IBM 3278 Display Station	3277
IBM 3284 Printer	3284
IBM 3286 Printer	3286
IBM 3287 Printer	3284 or 3286
IBM 3288 Line Printer	3288
IBM 3289 Printer	3288

SELECT=cudv

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 Figure 17 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=2 | MODEL=3]

is the model number of the printer; the default is model 2.

The following printers can be attached to a 3271, 3274, and 3276 cluster control unit:

- IBM 3284 Printer, Model 2
- IBM 3286 Printer, Model 2
- IBM 3288 Printer, Model 2

The following printers can be attached to a remote 3274 Model 1C cluster control unit:

- IBM 3284 Printer Model 2
- IBM 3286 Printer Model 2
- IBM 3287 Printer Models 1 and 2
- IBM 3288 Printer Model 2
- IBM 3289 Printer Models 1 and 2

The following printers can be attached to a 3276 cluster control unit:

- IBM 3287 Printer Models 1 and 2

The following printers can be attached to a standalone 3275 station:

- IBM 3284 Printer, Model 3
- IBM 3286 Printer, Model 3 (via RPQ MB4317)

FEATURE=OPRDR
specifies the optional feature, operator identification card reader, that is available on the 3277 Display Station, Model 2, or the magnetic slot reader on a 3278 Display Station, Models 2, 3, and 4.

Examples

Example 1: This TERMINAL macro describes a 3277 with a selection address of 2, and a control unit 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 control unit address of 3.

```
TERMINAL TERM=3286,SELECT=E3C3
```

Example 3: This TERMINAL macro describes a 3284 with a selection address of 4 and a control unit 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 control unit address of 0.

```
TERMINAL TERM=3275,SELECT=6040,MODEL=3
```

If no printer is attached to the 3275, code:

```
TERMINAL TERM=3275,SELECT=6040
```

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

Figure 17. Remote 3270 Control Unit and Device Addressing

Figure 18 shows some examples of valid polling characters. For more information on polling sequences, see IBM 3270 Information Display System Component Description.

3271, 3274, and 3276 Addressing			3275 Addressing		
		EBCDIC			EBCDIC
General Poll for Control Unit 5	Control Unit Address	C5 C5	General Poll for Control Unit 5	Control Unit Address	C5 C5
	Device Address	7F 7F		Device Address	7F 7F
Specific Poll Device 4 on Control Unit 5	Control Unit Address	C5 C5	Specific Poll for Control Unit 5	Control Unit Address	C5 C5
	Device Address	C4 C4		Device Address	40 40
Select Device 4 on Control Unit 5	Control Unit Address	E5 E5	Select Control Unit 5	Control Unit Address	E5 E5
	Device Address	C4 C4		Device Address	40 40

Figure 18. Examples of Remote 3270 Addressing

RDEVICE Macro

Use the RDEVICE macro instruction to generate a real device block (RDEVBLOK). You must code an RDEVICE macro for each real I/O device in your I/O configuration. The maximum number of real devices that can be included on the real VM/370 system is 3276.

The RDEVICE macro instructions describe each device, or group of devices, attached to the System/370. These can be in any order, but they must be contiguous, and must precede all RCTLUNIT and RCHANNEL macros in the real I/O configuration file (DMKRIO). Also, the RDEVICE macro instructions must follow all the groups of CLUSTER and TERMINAL macros, if there are any. The first RDEVICE macro generates the label DMKRIODV, which indicates the start of the 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 or 3705 device, see the "Generating a VM/370 System that Supports the 3704/3705" section of Part 1 for additional information and special considerations.

Also, see the "Planning Considerations for Remote 3270s" section of Part 1 before you code an RDEVICE macro for a binary synchronous line that is used by remote 3270s.

Before you code an RDEVICE macro for a 3800 printer device, see the "Generating a VM/370 System that Supports the 3800 Printer" section of Part 1 for additional information and special considerations.

The format of the RDEVICE macro is:

```
Name | Operation | Operands
-----+-----------+--------------------------------------------------
     | RDEVICE   | ADDRESS={cuu     },DEVTYPE=type[,MODEL=model]
     |           |         {(cuu,nn)}
     |           | [,FEATURE=(feature[,feature]...)]
     |           | [,CLASS={(cl[,cl]...)}]
     |           |         {DASD        }
     |           |         {TAPE        }
     |           |         {TERM        }
     |           |         {GRAF        }
     |           |         {URI         }
     |           |         {URO         }
     |           | [           {BSCA }]
     |           | [           {IBM1 }]     [,CPTYPE={EP }]
     |           | [           {SDLC }]
     |           | [,ADAPTER= {TELE2}]
     |           | [           {TYPE1}]
     |           | [           {TYPE2}]
     |           | [           {TYPE3}]
     |           | [           {TYPE4}]     [,ALTCU=cuu    ]
     |           | [,SETADDR=sadnum]
     |           | [,CPNAME=cpname]
     |           | [,BASEADD=cuu]
     |           | [,CLUSTER=label]
     |           | [,IMAGE=imagelib]
     |           | [,CHARS=ffff]
     |           | [,FCB=lpi]
     |           | [,DPMSIZE=n]
```

<u>where</u>:

ADDRESS={cuu | (cuu,nn)}

is the real I/O device address (or addresses).

The address, cuu, is 3 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.

The value, 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 address 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 characters 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 because only one system display console can be specified for each RDEVICE macro.

DEVTYPE=type

is the type of device.

The device type can be ICA, CTCA, 1017, 1018, 1052, 1053, 1403, 1442P, 1442R, 1443, 2150, 2250, 2260, 2265, 2301, 2303, 2305, 2311, 2314, 2319, 2321, 2401, 2402, 2403, 2404, 2415, 2420, 2495, 2501, 2520P, 2520R, 2540P, 2540R, 2671, 2701, 2702, 2703, 2955, 3036, 3066, 3138, 3148, 3158, 3203, 3210, 3211, 3215, 3277, 3284, 3286, 3287, 3288, 3330, 3333, 3340, 3350, 3410, 3411, 3420, 3505, 3525, 3704, 3705, 3800, 3851, or 7443.

Coding Considerations

For TWX terminals and 3101 display 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, 2703, 3704, or 3705. 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. An MSS 3330V device address must be defined as DEVTYPE=3330 with one of the two FEATURE= operands allowed. Refer to the explanation of the FEATURE operand.

For local 3270 terminals attached via a 3272 Control Unit Model 2, specify the following for DEVTYPE=:

Device Type	DEVTYPE=
3277	3277
3284	3284
3286	3286
3287	3284 or 3286
3288	3288

For local 3270 terminals attached via a 3274 Control Unit Model 1B specify the following for DEVTYPE=:

Device Type	DEVTYPE=
3277	3277
3278	3277
3284	3284
3286	3286
3287	3284 or 3286
3288	3288
3289	3288

For a local 3270 attached through a 4341 support processor, code a 3287 as a 3284 or 3286.

For the following devices, which have no device type, specify:

- ICA for Integrated Communications Adapter
- CTCA for Channel-to-Channel Adapter

The system console must be specified in both the RDEVICE and RCTLUNIT macros. Specify the system console in both macros as follows:

System/370 Model	System Console
135, 135-3, 145, 145-3, 155 II	3210 or 3215
138	3138 (if in display mode) 3215 (if in printer-keyboard mode)

System/370 Model	System Console
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
3031, 3032, 3033	3036
4331, 4341	3278 Model 2A (if in display mode) 3215 (if in printer-keyboard mode)

The device types, 2540R and 2540P, refer to the same IBM 2540 Card Read Punch (as do 1442P and 1442R, and 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 System/370 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 the error recordings, if there are any. CP does not use unsupported devices for its own operations.

If a device specified in the RDEVICE macro is not supported by VM/370, the following MNOTE message (warning level) is generated:

    UNSUPPORTED DEVICE TYPE

The device is generated as an unsupported device. An unsupported device can only be used 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/370 directory for the virtual machine, or if it is attached to it by the CP ATTACH command.

Note: If you code a 2702 device type the SETADDR value must be specified.

MODEL=model

is the model letter and number for 3704 or 3705 or the model number, if any, of a 2305, 3330, or 3333 device, or a tape device, 3278 display or 3203 printer. Model number, if not specified, defaults to zero except that for the 3203 it defaults to 4. It must be coded for 3330 devices, 3278 display, and 3203 printer. If only a number is coded for 3704 or 3705 devices, an MNOTE is generated.

model is a value that can be:

Value	Device
1 or 2	2305
4 or 5	3203
1, 2, or 11	3330
1 or 11	3333
A1 - H8	3704, 3705-I, or 3705-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

Notes:

1. The 3277 Model 1 is a 480-character display screen and is only supported by VM/370 as a dedicated device.
2. If a model number is included for devices that do not require model numbers, the system generation is terminated with an error message.

FEATURE=(feature[,feature]...)

are the device's optional features. These features can be written in any order. These features 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, Models 2, 3, or 4
SYSVIRT	A 3330V (DEVTYPE=3330) device that may be used by VM/370 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
4WCGMS	A 3800 (DEVTYPE=3800) device with four Writeable Character Generation Modules

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.

If the local 3277 or 3278 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 if he inserts a magnetically encoded card. Use the FEATURE=OPRDR operand of the RDEVICE macro to specify a 3277 or 3278 device with a card reader. If you do not want access authorization by a card reader, do not code FEATURE=OPRDR in the RDEVICE macro. FEATURE=OPRDR is invalid if DEVTYPE=3158.

Although still allowable, it is not necessary to designate FEATURE=(2CHANSW/4CHANSW) on the RDEVICE macro. DMKCPI dynamically determines whether or not the hardware has a two- or four-channel switch feature.

```
CLASS= { (cl[,cl]...) }
       { DASD         }
       { TAPE         }
       { TERM         }
       { GRAF         }
       { URI          }
       { URO          }
```

is the device class; either the output spooling class or a special subclass for unsupported devices.

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 only be specified for a 1403, 1443, or 3211 printer, or 2520P, 2540P or 3525 card punch. The spooling class, cl, is one alphameric 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.

Coding Considerations
The following information should be helpful when you code this operand:

- For more information about spooling classes, see the VM/370 Operator's Guide.

- The class is used by the CP START command and may be changed by this command. For a complete description of the START command, see the _VM/370 Operator's Guide_.

- A class C punch should be included if accounting cards are desired.

Subclass for Unsupported Devices
Specify a device subclass only for unsupported device types. 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.

The 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/370. It does not appear in the list of supported devices in Part 1 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 inhibiting CCW translation (by issuing SET NOTRANS ON). Note that a maximum of 32 sense bytes can be contained in the RDEVBLOK created for an unsupported device.

2. The CLASS operand is invalid if you are specifying service record file devices.

ADAPTER= {BSCA | IBM1 | SDLC | TELE2 | TYPE1 | TYPE2 | TYPE3 | TYPE4}

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, or ICA is specified, and 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, or 3705. 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 or 3705.

SDLC specifies that a 4331 Communications Adapter operate 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, or to a Telegraph Terminal Control Type II in a 2702 or 2703, or to a Line Interface Base Type I in a 3704 or 3705.

TYPE1, specifies the channel adapter accessed by a 3704. For DEVTYPE=3705, TYPE4 should be coded. In identifying the channel adapter, TYPE1 or TYPE4 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 which has ADAPTER=TYPE1, or TYPE4.

SETADDR=sadnum

is the set address (SAD) command to be 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={EP}

is the 3704/3705 control program to be run in a 3704 or 3705 Communications Controller. EP specifies the 2701, 2702, or 2703 Emulation Program.

ALTCU=cuu
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 address position of 0 or 8. Otherwise, the following MNOTE is issued:

    INVALID ALTCU ADDRESS

The ALTCU operand is only valid for tape and DASD devices. An MNOTE is issued if an invalid device type is specified.

    "ALTCU" IS INVALID FOR DEVICE TYPE "devtype"

The ALTCU operand should only be specified when the installation has the string switch feature to support two control unit paths to a device.

The ALTCU cuu address should specify the low address associated with the alternate real control unit. There is one occasion when the ALTCU cuu should specify the logical control unit address. When the FEATURE=xxx-Device operand indicates that the control unit supports more than sixteen devices and the devices on the second or subsequent group of sixteen devices are defined by separate RDEVICE macros. The ALTCU cuu should identify the logical RCUBLOK in VM/370. VM/370 constructs one RCUBLOK for each set of sixteen devices supported by the real control unit.

Given an alternate control unit configuration where each of two control units support thirty-two devices, the following two macro definitions are acceptable:

```
RDEVICE ADDRESS=(300,32),ALTCU=400
RDEVICE 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=cpname
is the 1- to 8-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.

BASEADD=cuu
is the native address (load address) of the 3704/3705 that controls the physical line(s). This operand is required for correct operation of VM/370 recovery management for emulation lines based on a 3704/3705. This operand is valid only if ADAPTER=IBM1 (or =TELE2 or =BSCA).

CLUSTER=label
is the label of the CLUSTER macro that defines the clustered or standalone remote 3275 or 3277 station attached to this line. This operand is valid only if ADAPTER=BSCA is specified.

IMAGE=imagelib
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=ffff
is 1 to 4 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.

DPMSIZE=n
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.)

FCB=lpi
is the FCB to be used for the page separator (6, 8, or 12) for the 3800 printer device after a cold start if none is specified on the START command. If this operand is omitted, the default is 6.

Examples:

The following examples illustrate the use of the RDEVICE macro instructions to 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,
        CLASS=(A,C)
RDEVICE ADDRESS=(0C0,4),DEVTYPE=2401
RDEVICE ADDRESS=(030,8),DEVTYPE=2702,ADAPTER=TELE2,
        SETADDR=2
```

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, 3203, 3211, 1443, 2540P, 3525, 2540R, 3505, or 2501 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.

3704/3705 Error Messages

The RDEVICE macro instruction generates an entry in a table of programmable communications controllers when DEVTYPE=3704 or 3705 is specified. This table can have a maximum of 10 entries; the following message results if more than ten 3704 or 3705 devices are specified:

```
MORE THAN 10 TP CONCENTRATORS
```

If this message prints, it indicates that the RDEVBLOK is generated, but no entry is made in the Programmable Communications Controller table.

Control Unit Error Messages

The RCTLUNIT macro generates an RCUBLOK containing an index to each of sixteen 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 specifying the ALTCU operand is defined and an RDEVICE macro is also defined for a device with the alternate control unit address:

```
CONTROL UNIT TABLE FOR raddr1 IN USE by raddr2
```

For 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 150 and an alternate control unit address of 150. The ALTCU=150 specification indicates that the 150 RCUBIOK will contain an index to the 150 RDEVBLOK. In this example, a RDEVICE macro also appears for device 150. A conflict arises since the RCUBIOK index for control unit 150 cannot index to both RDEVBLOK 140 and RDEVBLOK 150. In the above example, the user must remove the 150 RDEVICE macro to resolve the conflict.

FEATURE=(2CHANSW/4CHANSW)

Specifying the FEATURE=(2CHANSW/4CHANSW) option on the RDEVICE macro to indicate whether the hardware will support reserve/release CCWs is unnecessary. DMKCPI makes an accurate and dynamic determination by issuing a release CCW to the tape or count-key-data DASD devices. If the hardware supports the Two- or Four-Channel Switch Feature, the FTRRSRL bit is turned on in the RDEVFTR field. The FEATURE=(2CHANSW/4CHANSW) option on the RDEVICE macro is ignored since determination is made by the initialization routine. For compatibility reasons, the FEATURE=(2CHANSW/4CHANSW) operand is still allowed, but when specified, causes the following MNOTE to be issued:

```
{2CHANSW}  FEATURE IGNORED
{4CHANSW}
```

RCTLUNIT Macro

Use the RCTLUNIT macro to generate 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 categories: 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 divisible by eight. 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, several 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 system console and the 2540 card reader punch 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 sixteen. 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 has the hardware feature that allows it to support more than eight devices, the RCTLUNIT macro must specify FEATURE=xxx-DEVICE, where xxx is the number of addressable devices that can be attached to this control unit. The number of devices specified must be divisible by sixteen and rounded to the next higher increment of sixteen 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/370 requires that all devices on one physical control unit be specified on a single RCTLUNIT macro. The microcode in the 3830-2 which supports 3350 DASD allows address skipping (in blocks of eight addresses) on the same physical control unit.

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 given physical control unit must be specified with a single RCTLUNIT macro and use 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 a RCTLUNIT macro coded for it. For example, if a 3210 is defined with an RDEVICE macro, it must have a corresponding RCTLUNIT macro.

The RCTLUNIT macro instructions describing the control units for your installation's computing system may be in any order, but they must be contiguous and follow all of the RDEVICE macro instructions in the 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 may 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, the name RCU230 is generated.

The format of the RCTLUNIT macro is:

Name	Operation	Cperands
	RCTLUNIT	ADDRESS= cuu ,CUTYPE=type [ALTCH=(n,n,n)][,FEATURE=xxx-DEVICE]

where:

ADDRESS=cuu
: is the real address of the control unit. 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: If your installation has a 2701, 2702, or 2703, and a 3704 or 3705 executing in emulation mode, you must be sure that their addresses are not the same.

CUTYPE=type
: is the device type of the control unit. One of the following device type numbers can be specified: 1052, 1442, 2150, 2250, 2314, 2319, 2403, 2404, 2415, 2495, 2501, 2520, 2701, 2702, 2703, 2803, 2804, 2820, 2821, 2822, 2826, 2835, 2840, 2841, 2844, 2845, 2848, 2955, 3036, 3066, 3138, 3148, 3158, 3210, 3215, 3272, 3274, 3345, 3411, 3505, 3525, 3704, 3705, 3803, 3811, 3830, 3851, 7443, ICA, IFA, ISC, CTCA.

  In addition, any other control unit that can be attached to a real System/370 may be specified in a RCTLUNIT macro instruction by its device type.

  Note: Specify an Integrated Printer Adapter (IPA) as a 2821. Specify a 3274 Model 1B as a 3272. Also, specify a 3880 Model 1 as a 3830. If using IFA with 3344 or 3350 devices with more than 16 logical units, specify CUTYPE=3830 with either FEATURE=32-DEVICE or 64-DEVICE. If CUTYPE=CTCA, the low order digit of ADDRESS= must be 0. The Integrated File Adapter's 9821 feature provides a range of addresses that is too large for the RCTLUNIT macro to process when used with 3344s. The range of addresses (160-1F7) is treated as invalid.

Even though some devices attach directly to the channel without a separate control unit, a 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 a RCTLUNIT macro. If several different devices have addresses that are within the same control unit address, only one RCTLUNIT macro can be specified. Which control unit you specify is not relevant.

ALTCH=(n,n,n)
: 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. Up to three alternate channels can be specified.

  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).

FEATURE=xxx-DEVICE
: is the optional control unit feature. The feature, xxx-DEVICE, indicates that the control unit is 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. "Appendix B: Configuration Aid" lists the maximum number of devices that may be specified for each control unit. This feature may be specified for a 2403, 2702, 2703, 2803, 2835, 3272, 3345, 3704, 3705, 3803, 3830, ICA, IFA, or ISC.

  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 the "Configurations" section of Part 1.

  Warning: The starter system does not provide support for device configurations over 16 when you are defining the devices needed to do the system generation. Therefore, if your installation includes control units that share more than 16 devices and are also switchable to another processor, the channel interface enable switch from the other processor should be in the disable position while you perform the system generation.

Examples:

The following examples illustrate the use of the RCTLUNIT macro instruction to describe the control units for: a 3215 console printer-keyboard with address 01F, a 2314, and a 3705 with lines 040 through 04B.

```
RCTLUNIT ADDRESS=018,CUTYPE=3215
RCTLUNIT ADDRESS=230,CUTYPE=2314
RCTLUNIT ADDRESS=040,CUTYPE=3705,FEATURE=16-DEVICE
```

Channel Error Message

The RCHBLOK contains an index to each of thirty-two 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 message is issued when an RCTLUNIT macro is defined specifying the ATLCH operand and an RCTLUNIT macro is also defined which creates an RCUBLOK for the alternate channel address.

```
CHANNEL TABLE FOR RCUxxx IN USE BY RCUyyy
```

For 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 since the RCHBLOK cannot index to both the 250 and 350 RCUBLOKS. In the above configuration, the user must remove the RCTLUNIT macro and RDEVICE macro for 350.

RCHANNEL Macro

Use the RCHANNEL macro to generate 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 the channels for your installation may be in any order, but they must be contiguous and follow all of the RCTLUNIT macro instructions in the module DMKRIO. The first RCHANNEL macro instruction also generates the label DMKRIOCH, which indicates the start of the real channel blocks to CP.

No name is 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.

The format of the RCHANNEL macro is:

Name	Operation	Operands
	RCHANNEL	ADDRESS=address ,CHTYPE= { SELECTOR | MULTIPLEXOR | BLKMPXR }

where:

ADDRESS=address
: is the real address of the channel. It is a hexadecimal number from 0 to F.

CHTYPE= { SELECTOR | MULTIPLEXOR | BLKMPXR }
: is the type of channel.

SELECTOR indicates a selector channel.

MULTIPLEXOR indicates a byte-multiplexer channel.

BLKMPXR indicates a block multiplexer channel.

Examples:

The following examples illustrate the use of the RCHANNEL macro instruction to 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.

RIOGEN Macro

Use the RIOGEN macro instruction to generate the channel index table and unit record and console tables. It 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. The format of the RIOGEN macro is:

Name	Operation	Operands
	RIOGEN	CONS=cuu [,ALTCONS=(cuu[,cuu,cuu...])] [,SRF=(cuu[,cuu,cuu...])]

where:

CONS=cuu
is the address of the VM/370 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 (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=(cuu=[,cuu,cuu...])]
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 physically located as close as possible to the primary system console, may be any device supported as a VM/370 logon device (except for those remote terminals connected via 3704/3705 Communications Controllers). If the primary system console is not operational at VM/370 system initialization, an attempt will be made to access the first alternate console. If the first alternate console is not operational, an attempt will be made to start the next alternate console. If an operational console is found, the console will be used as the VM/370 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 IBM 3704/3705 Communications Controller unless the 3704/3705 was previously loaded by some other operating system with a 270X Emulator Program.

If the alternate console is an IBM 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=(cuu[,cuu,cuu...])]
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 3033 AP system, there are two 3036 consoles. This configuration has four 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. The RIOGEN macro statement produces an MNOTE warning message if you specify more than 32 SRF devices.

Notes:

1. In 3033 MP environments with I/O configured asymmetrically to one processor, in order 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 found to be inaccessible during initialization of the error recording cylinders, an error message is sent to the system operator. Processing continues, however, the frames from that SRF device are not placed on the error recording cylinders.

Examples:

The following examples define a primary system console (01F) with an alternate console (050), a system console (009) with no alternate console, and a primary system console (01F) with alternates at (050) and (060).

```
RIOGEN CONS=01F,ALTCONS=050
RIOGEN CONS=009
RIOGEN CONS=01F, ALTCONS=(050, 060)
```

Example of Coding the Real I/O Configuration File (DMKRIO)

In this example, macros are coded to support the following real devices:

- 1 2540 Card Reader/Punch
- 1 3505 Card Reader
- 1 3525 Card Punch
- 2 1403 Printers with the Universal Character Set feature
- 1 3211 Printer with the Universal Character Set feature
- 1 3215 Console Printer-Keyboard
- 1 2955
- 2 3705 Communications Controllers (one with an IBM1 adapter and the other with a BSCA adapter)
- 2 2702 Transmission Control Units (with one that supports Teletype terminals)
- 1 2314 Direct Access Storage Facility with 8 modules, attached via an Integrated File Adapter
- 1 2305 Fixed Head Storage with 8 addresses
- 2 3330 Disk Storage (One unit has eight modules and the other has ten. The unit with ten modules has eight of them switchable between two channels.)
- 1 3350 Direct Access Storage with 8 addresses
- 1 2401 Magnetic Tape Unit with 4 tape drives
- 1 3410 Magnetic Tape Unit, Model 3, with 1 tape drive
- 1 3420 Magnetic Tape Unit, Model 7, with 2 tape drives
- 1 Multiplexer channel
- 1 Selector channel
- 3 Block multiplexer channels
- 1 Channel-to-Channel Adapter
- 2 channel interfaces on the 3851 MSC
- 2 3830-3 control units for access to 3330V devices, each with a single channel interface
- 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/370 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

Figure 19 shows the real configuration.

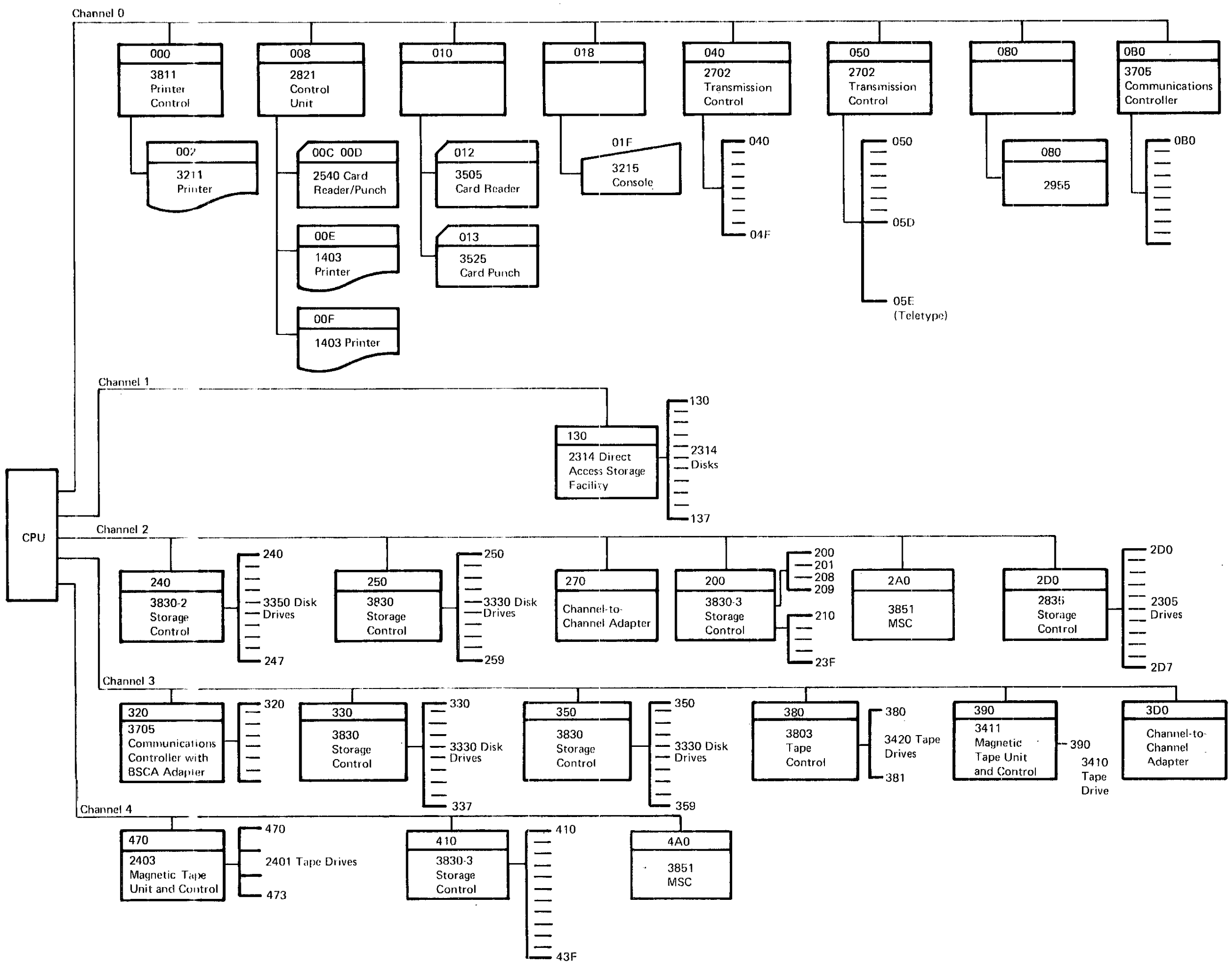


Figure 19. Example of a Real Configuration

The real I/O configuration file that supports this example is shown in Figure 20.

```
DMKRIO CSECT
         RDEVICE ADDRESS=002,DEVTYPE=3211,CLASS=(X,A),FEATURE=UNVCHSET
         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=012,DEVTYPE=3505
         RDEVICE ADDRESS=013,DEVTYPE=3525,CLASS=(X,A)
         RDEVICE ADDRESS=01F,DEVTYPE=3215
         RDEVICE ADDRESS=(040,16),DEVTYPE=2702,ADAPTER=IBM1,SETADDR=2
         RDEVICE ADDRESS=(050,14),DEVTYPE=2702,ADAPTER=IBM1,SETADDR=2
         RDEVICE ADDRESS=05E,DEVTYPE=2702,ADAPTER=TELE2,SETADDR=1
         RDEVICE ADDRESS=080,DEVTYPE=2955
         RDEVICE ADDRESS=0B0,DEVTYPE=3705,ADAPTER=IBM1
         RDEVICE ADDRESS=(130,8),DEVTYPE=2314
         * 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
         RDEVICE ADDRESS=(250,64),DEVTYPE=3330,MODEL=1
         RDEVICE ADDRESS=270,DEVTYPE=CTCA
         RDEVICE ADDRESS=2A0,DEVTYPE=3851
         RDEVICE ADDRESS=2D0,DEVTYPE=2305,MODEL=2
         RDEVICE ADDRESS=320,DEVTYPE=3705,ADAPTER=BSCA
         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=(35A,54),DEVTYPE=3330,MODEL=1
         RDEVICE ADDRESS=(380,2),DEVTYPE=3420,FEATURE=DUALDENS,MODEL=7
         RDEVICE ADDRESS=390,DEVTYPE=3410,FEATURE=DUALDENS,MODEL=3
         RDEVICE ADDRESS=3D0,DEVTYPE=CTCA
         RDEVICE ADDRESS=(410,48),DEVTYPE=3330,MODEL=1,FEATURE=VIRTUAL
         RDEVICE ADDRESS=(470,4),DEVTYPE=2401,FEATURE=DUALDENS,MODEL=5
         RDEVICE ADDRESS=4A0,DEVTYPE=3851
         RCTLUNIT ADDRESS=000,CUTYPE=3811
         RCTLUNIT ADDRESS=008,CUTYPE=2821
         RCTLUNIT ADDRESS=010,CUTYPE=3505
         RCTLUNIT ADDRESS=018,CUTYPE=3215
         RCTLUNIT ADDRESS=040,CUTYPE=2702,FEATURE=16-DEVICE
         RCTLUNIT ADDRESS=050,CUTYPE=2702,FEATURE=16-DEVICE
         RCTLUNIT ADDRESS=080,CUTYPE=2955
         RCTLUNIT ADDRESS=0B0,CUTYPE=3705,FEATURE=16-DEVICE
         RCTLUNIT ADDRESS=130,CUTYPE=IFA
         RCTLUNIT ADDRESS=200,CUTYPE=3830,FEATURE=64-DEVICE
         RCTLUNIT ADDRESS=240,CUTYPE=3830
         RCTLUNIT ADDRESS=250,CUTYPE=3830,FEATURE=16-DEVICE
         RCTLUNIT ADDRESS=270,CUTYPE=CTCA
         RCTLUNIT ADDRESS=2A0,CUTYPE=3851
         RCTLUNIT ADDRESS=2D0,CUTYPE=2835
         RCTLUNIT ADDRESS=320,CUTYPE=3705
         RCTLUNIT ADDRESS=330,CUTYPE=3830
         RCTLUNIT ADDRESS=350,CUTYPE=3830,FEATURE=16-DEVICE
         RCTLUNIT ADDRESS=380,CUTYPE=3830
         RCTLUNIT ADDRESS=390,CUTYPE=3411
```

Figure 20. Real I/O Configuration File (Part 1 of 2)

```
RCTLUNIT ADDRESS=3D0,CUTYPE=CTCA
RCTLUNIT ADDRESS=400,CUTYPE=3830,FEATURE=64-DEVICE
RCTLUNIT ADDRESS=470,CUTYPE=2403
RCTLUNIT ADDRESS=4A0,CUTYPE=3851
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=050
END
```

Figure 20. Real I/O Configuration File (Part 2 of 2)

Preparing the CP System Control File (DMKSYS)

The CP system control file consists of macros that describe the CP system residence device, the system storage size, the CP-owned direct access devices, the system operator's user identification, the system timer value, the system pointer variables, automatic performance monitoring parameters, and security journaling parameters. The installation is responsible for ensuring the presence and accuracy of the macros described below.

The DMKSYS ASSEMBLE file provided with the starter system does not assemble properly unless you have reserved adequate space for the CP nucleus.

The file should be placed in the card reader in the order shown:

```
DMKSYS  CSECT
        SYSOWN  macro
        SYSRES  macro
        SYSOPR  macro
        SYSCOR  macro
        SYSTIME macro
        SYSMON  macro
        SYSJRL  macro
        SYSLOCS macro
        END
```

Notes:

1. Samples of the DMKSYS and DMKSNT files are provided with the starter system. If you use these, you can save the CMS system at the end of the system generation procedure. VM/370 prompts you to tell it if you want the sample DMKSYS file punched. 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.

The DMKSYS module supplied with the 2314 starter system is:

```
DMKSYS CSECT
       SYSOWN (VMRELn¹,TEMP)
       SYSRES SYSVOL=VMRELn,SYSRES=131,SYSTYPE=2314,SYSNUC=12,          X
              SYSWRM=16,SYSERR=17,SYSCKP=(101,2)
       SYSOPR SYSOPER=OPERATOR,SYSDUMP=OPERATNS
       SYSCOR RMSIZE=512K
       SYSTIME ZONE=4,LOC=WEST,ID=EDT
       SYSMON AUTO=NO
       SYSJRL
       SYSLOCS
       END
```

----

[1]VMRELn may be VMREL1, VMREL2, VMREL3, and so forth depending on the release level.

The DMKSYS module supplied with the 3330 starter system is:

```
DMKSYS CSECT
       SYSOWN  (VMRELn¹,TEMP)
       SYSRES SYSVOL=VMRELn,SYSRES=131,SYSTYPE=3330,SYSNUC=7,          X
              SYSWRM=10,SYSERR=11,SYSCKP=202
       SYSOPR SYSOPER=OPERATOR,SYSDUMP=OPERATNS
       SYSCOR RMSIZE=512K
       SYSTIME ZONE=4,LOC=WEST,ID=EDT
       SYSMON AUTO=NO
       SYSJRL
       SYSLOCS
       END
```

The DMKSYS module supplied with the 3340 starter system is:

```
DMKSYS CSECT
       SYSOWN  (VMRELn¹,TEMP)
       SYSRES SYSVOL=VMRELn,SYSRES=131,SYSTYPE=3340,SYSNUC=15,         X
              SYSWRM=21,SYSERR=22,SYSCKP=(174,3)
       SYSOPR SYSOPER=OPERATOR,SYSDUMP=OPERATNS
       SYSCOR RMSIZE=512K
       SYSTIME ZONE=4,LOC=WEST,ID=EDT
       SYSMON AUTO=NO
       SYSJRL
       SYSLOCS
       END
```

The DMKSYS module that is used with the 3350 starter system is:

```
DMKSYS CSECT
       SYSOWN  (VMRELn¹,TEMP)
       SYSRES SYSVOL=VMRELn,SYSRES=131,SYSTYPE=3350,SYSNUC=4,         X
             SYSWRM=6,SYSERR=7,SYSCKP=277
       SYSOPR SYSOPER=OPERATOR,SYSDUMP=OPERATNS
       SYSCOR RMSIZE=512K
       SYSTIME ZONE=4,LOC=WEST,ID=EDT
       SYSMON AUTO=NO
       SYSJRL
       SYSLOCS
       END
```

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/370 system.

- Provide separate CP volumes for paging and spooling and have the volumes mounted on separate channels.

- If you have a heavy I/O production virtual machine (for example, one that is executing OS/VS1 or DOS/VS), 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

----

[1] VMRELn may be VMREL5, VMREL6, and so forth, depending on the release level.

separate volumes from users' read/write minidisks. If possible, also keep them on separate channels.

- If your installation is likely to have a large number of CMS users active at one time, you should distribute the 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 program products on one spindle and the CMS non-resident commands on another spindle.

- If your entire paging area can be contained in the fixed head area of a 3340 or 3350, you should place it there. To do this, use the Format/Allocate program to allocate the fixed head area as temporary space; this should be the only temporary space allocated on that volume. Then specify PAGE for that volume when you code the SYSOWN macro during system generation.

- The relative amounts of free storage used for dynamic paging and free storage can be optimized by using the FREE operand of the SYSCOR macro statement. You should allocate one 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 environment 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: Performance/Monitor Analysis Program are designed to make data collection and reduction easy, thereby allowing the analyst to concentrate on analysis. Data collection can be performed on a regular basis by specifying AUTO=YES on the SYSMON macro instruction. The system will assume the default values for the other operands if none are specified.

SYSOWN Macro

Use the SYSOWN macro to generate 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/370 paging, spooling, or temporary disk space. It must contain a CP allocation table at cylinder 0, record 4 allocating these areas. Even if a volume has a VM/370 allocation table at cylinder 0, record 4, 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.

Special Considerations for Allocating Space on CP-Owned Volumes: The following considerations should help you to allocate space efficiently on CP-owned volumes:

- If a volume is specified in a SYSOWN statement but is not mounted when the generated system is loaded (via the IPL command), that volume is considered unavailable to VM/370. Processing continues, if possible. The operator can mount and attach the volume later, if it is needed.

- Only those volumes that contain paging and spooling space or TDSK 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, rebuild the CP nucleus, and reload it on the system residence volume. Use the GENERATE EXEC procedure to reassemble DMKSYS and reload the CP nucleus.

- If your installation has saved systems (systems that can be loaded by name, thus bypassing 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 one page for system use.

- If your VM/370 system has a 3704 or 3705, you must reserve space on a CP-owned volume to contain the 3704/3705 control program image. See the "Generating a VM/370 System that Supports the 3704/3705" section of Part 1 for information about how much DASD space you should reserve.

The name field must not be specified for the SYSOWN macro. The format of the SYSOWN macro is:

Name	Operation	Operands
	SYSOWN	(volid ┌,TEMP┐) [,(volid ┌,TEMP┐)...] └,PAGE┘ └,PAGE┘

where:

volid
    is the CP-owned volume identification of from 1 to 6 alphameric characters.

```
┌      ┐
|,TEMP|
|,PAGE|
└      ┘
```

indicates to VM/370 how allocatable space on the specified volume should be used.

TEMP indicates that this volume is to be used primarily for spooling space and it will also be used for paging space if all volumes normally used for paging allocation are full or unavailable. TEMP is the default option. (TEMP space must be formatted by the CP Format/Allocate service program. At this time areas for user TDSKs and directory are allocated.)

Note: If no volume is specified as being preferred for TEMP space allocation, no spooling operations may be performed. Thus any data transfer channel program started to a virtual unit record output device ends with UC status in the virtual CSW and the intervention required bit set in the virtual sense byte.

PAGE indicates that this volume is to be used primarily for paging, (this does not include spool space). Note that TDSK requests from this volume occur only when a request for space cannot be satisfied on a volume with a TEMP allocation.

Example:

The following SYSOWN macro designates the CPDRM1 volume as paging space and the CPDSK1 and CPDSK2 volumes as spooling space and paging overflow:

```
SYSOWN (CPDRM1,PAGE),(CPDSK1,TEMP),CPDSK2
```

SYSRES Macro

Use the SYSRES macro instruction to describe the characteristics of the CP system residence volume.

The name field must not be specified for the SYSRES macro instruction.

<u>Special Considerations for Coding the SYSRES Macro</u>: The following information should help you when you code the SYSRES macro:

- All operands must be specified with appropriate values.
- The cylinders 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.
- VM/370 allows the 2314 or 2319 "alternate tracks" cylinders 200-202 to be used for normal data if they are not needed to replace defective tracks.
- On a 3340, the "alternate tracks" cylinders can no longer be used for normal data. (Previously, these cylinders were sometimes used for normal 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/370 SYSRES volume.

The format of the SYSRES macro is:

```
| Name | Operation | Operands                                       |
|------|-----------|------------------------------------------------|
|      | SYSRES    | SYSVOL=serial,                                 |
|      |           | SYSRES=address,                                |
|      |           | SYSTYPE= /2305\ ,                              |
|      |           |          |2314|                                |
|      |           |          }2319{                                |
|      |           |          }3330{                                |
|      |           |          |3340|                                |
|      |           |          \3350/                                |
|      |           | SYSNUC=strtcyl,                                |
|      |           |                                                |
|      |           |                       ┌          ┐             |
|      |           | SYSERR=[ (]strtcyl    |,cylcount| [) ]         |
|      |           |                       |,2        |             |
|      |           |                       └          ┘             |
|      |           |                                                |
|      |           |                       ┌          ┐             |
|      |           | SYSCKP=[ (]strtcyl    |,cylcount| [) ]         |
|      |           |                       |,1        |             |
|      |           |                       └          ┘             |
|      |           |                                                |
|      |           |                       ┌          ┐             |
|      |           | SYSWRM=[ (]strtcyl    |,cylcount| [) ]         |
|      |           |                       |,1        |             |
|      |           |                       └          ┘             |
```

where:

SYSVOL=serial
    is the volume identification of the system residence disk. The volume serial number, serial, is a character string with a maximum length of 6 characters.

SYSRES=address
    designates the device address of the DASD to contain the newly-generated system. This address is used only when the VM/370 nucleus is generated and written on the disk. Thereafter, you can IPL the system from any addressable disk device. The address is a 3-digit hexadecimal device address.

```
SYSTYPE= 2305
         2314
         2319
         3330
         3340
         3350
```

    is the device type of the system residence device.

    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. 2305 applies to both 2305-1 and 2305-2.

SYSNUC=strtcyl
    is the number of the real starting cylinder where the CP nucleus resides.

    The cylinder strtcyl is a 1- to 3-digit decimal number.

    Normally, a 2314 or 2319 device requires five contiguous cylinders, a 2305 or 3340 device also requires five contiguous cylinders, a 3330/3333 device requires three contiguous cylinders, and a 3350 device requires two cylinders, to contain the CP nucleus.

    Systems being generated with the virtual=real option require additional space. For information about how much space to allocate for virtual=real configurations, see "Specifying the Amount of Virtual=Real Space" in Part 1.

```
SYSERR=[ (]strtcyl |,cylcount| [)]
                   |,2       |
```

    is the number of the real starting cylinder where the error records are written, and optionally the number of cylinders required for error recording.

    The strtcyl is a 1- to 3-digit decimal number designating the starting cylinder of the error recording area.

    The cylcount is a 1-digit decimal number between 2 and 9 designating the number of cylinders.

```
SYSCKP=[ (]strtcyl|,cylcount|[)]
                  |,1       |
```

    is the number of the real starting cylinder, and optionally the maximum number of cylinders, to contain the dynamic checkpoint start data.

The strtcyl is a 1- to 3-digit decimal number designating the first real cylinder where CP checkpoint start information is to be saved.

The cylcount value is a 1-digit decimal number (1 through 9) that defines the maximum number of cylinders to contain checkpoint start data. If cylcount is not specified, 1 is the default value.

The cylcount operand is optional; if included, the strtcyl and cylcount operands must be separated by a comma and enclosed in parentheses. Parentheses are optional when only the strtcyl operand is specified.

The number of cylinders required for the checkpoint start data is dependent upon the device type. They are as follows:

Device Type	No. of Cylinders
2305	3
2314	2
2319	2
3330	1
3340	3
3350	1

```
               ┌            ┐
SYSWRM=[ (]strtcyl|,cylcount|[) ]
               |,1          |
               └            ┘
```

is the number of the real starting cylinder, and optionally the maximum number of cylinders, to contain the warm start data.

The strtcyl is a 1- to 3-digit decimal number designating the first real cylinder where CP warm start information is to be saved.

The cylcount value is a 1-digit decimal number (1 through 9) that defines the maximum number of cylinders to contain warm start data. If cylcount is not specified, 1 is the default value.

The cylcount operand is optional; if included, the strtcyl and cylcount operands must be separated by a comma and enclosed in parentheses. Parentheses are optional when only the strtcyl operand is specified. The following are valid entries for one cylinder warm start areas:

```
SYSWRM=(202,1)
SYSWRM=(202)
SYSWRM=202
```

Use the following formulas to calculate the number of warm start cylinders required. When you use the formulas, disregard all remainders. For example, for a 3330 system residence volume plus:

- A maximum of 40 spool files in the system at one time
- A maximum of 170 cylinders available for spool files
- A maximum of 50 active users at one time

the calculation is

$$N = \frac{[59 + 40/40 + 170/170 + 200/50]}{57} = \frac{65}{57} = 1$$

Example:

The following SYSRES macro defines the system residence volume as the 2314 volume with a serial number of CPDSK1. During the system generation procedure this volume is found at address 230. The VM/370 system starts at cylinder 198, the error recording area starts at cylinder 4, and the warm start storage area is cylinder 202 and the checkpoint start storage area is cylinders 101 and 102. The format of the SYSRES macro is:

```
SYSRES SYSVOL=CPDSK1,SYSRES=230,SYSTYPE=2314,SYSNUC=198,         X
       SYSERR=4,SYSWRM=(202,1),SYSCKP=(101,2)
```

The formula for each device type is shown in Figure 21.

Device Type	Formula
2314/2319	$N = \frac{[34 + (NSF/40) + (NCS/170) + ((NAU \times 4)/50)]}{32}$
3340/2305	$N = \frac{[26 + (NSF/40) + (NCS/170) + ((NAU \times 4)/50)]}{24}$
3330	$N = \frac{[59 + (NSF/40) + (NCS/170) + ((NAU \times 4)/50)]}{57}$
3350	$N = \frac{[122 + (NSF/40) + (NCS/170) + ((NAU \times 4)/50)]}{120}$

where:

N is the number of cylinders required for warm start data.

NSF is the maximum number of spool files in the system at any one time. There are 40 spool file blocks per 4096-byte record.

NCS is the number of cylinders available for spool files. There are 170 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.

Figure 21. Warm Start Cylinder Calculations

SYSOPR Macro

Use the SYSOPR macro instruction to specify the system operator's userid, and the userid of the operator who is to receive VM/370 system dumps. The same userid may be specified in both operands.

The name field must not be specified for the SYSOPR macro instruction.

The format of the SYSOPR macro is:

Name	Operation	Operands
	SYSOPR	[SYSOPER=OPERATOR | SYSOPER=userid]
		[,SYSDUMP=OPERATNS | ,SYSDUMP=userid]

where:

```
┌                ┐
│SYSOPER=OPERATOR│
│SYSOPER=userid  │
└                ┘
```

is the userid of the virtual machine to be assigned to the system operator. If SYSOPER is not specified, the userid OPERATOR is used.

The userid is a character string up to 8 characters long.

```
┌                ┐
│SYSDUMP=OPERATNS│
│SYSDUMP=userid  │
└                ┘
```

is the userid of the virtual machine whose spool input receives the system dump file after a system restart. This userid also receives quest virtual machine dumps produced by the CP command VMDUMP, if you specify the destination as SYSTEM in the VMDUMP command. If SYSDUMP is not specified, the userid OPERATNS is used. If you intend to use IPCS, allow this operand to default to OPERATNS or specify the IPCS userid.

The userid is a character string up to 8 characters long.

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
```

SYSCOR Macro

Use the SYSCOR macro instruction to generate the internal control block called the CORTABLE. The AP operand specifies whether VM/370 will try to make use of an attached processor.

The name field must not be specified for the SYSCOR macro instruction.

The format of the SYSCOR macro is:

```
| Name | Operation | Operands                                |
|------|-----------|-----------------------------------------|
|      | SYSCOR    | RMSIZE={xxxxxK} [,FREE=ffff]            |
|      |           |        {   yyM} [,AP={YES}]             |
|      |           |                       {NO }             |
|      |           | [TRACE=nnn]                             |
```

where:

RMSIZE={xxxxxK | yyM}

is the amount of real storage available for VM/370. This value limits the amount of real storage used by VM/370 if it is less than the total amount of real storage available in the real machine. If the available real storage is less than this value when VM/370 is initialized, a message indicating the amount of storage available is displayed at the operator's console.

The value, xxxxx, is a 3- to 5-digit number that denotes 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 1- or 2-digit number that denotes 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 substantially larger than the size of real storage, because the generated core table uses a large amount of real storage.

FREE=ffff

is a 1- to 4-digit number that specifies the number of fixed free storage pages to be allocated at VM/370 initialization. This number must be greater than 3; the amount of storage represented must not be greater than 25% of the value specified for RMSIZE.

The recommended value for ffff is one page for each virtual machine that is logged on, based on the average number of virtual machine users.

If the FREE operand is not specified, VM/370 allocates three pages for the first 256K of real storage and one page for each additional 64K thereafter, not including the V=R size, if any. In AP mode, the default is increased by 25%.

AP= $\begin{Bmatrix} \text{YES} \\ \text{NO} \end{Bmatrix}$

YES specifies that processing is in attached processor mode if the attached processor is available at system IPL.

Note: An additional 25% of free storage (see FREE=) in AP mode.

NO specifies that processing is in uniprocessor mode regardless of the presence of an attached processor.

TRACE=nnn

is the decimal number of 4K pages to be used for the trace table. If the number of pages specified on the TRACE operand is not larger than the default trace table size provided by the system (one page for each 256K of real storage), the default size will be used for the trace table.

Examples:

The first example defines real storage as 256K (262,144 bytes) and the second example defines real storage as 1M (1,048,576 bytes).

```
SYSCOR   RMSIZE=256K
SYSCOR   RMSIZE=1M
```

SYSTIME Macro

Use the SYSTIME macro instruction to generate 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.

The format of the SYSTIME macro is:

```
┌──────────────────────────────────────────────────────────────────┐
│ Name  │ Operation  │ Operands                                     │
│──────────────────────────────────────────────────────────────────│
│       │            │ ┌              ┐                             │
│       │ SYSTIME    │ │ZONE=0        │                             │
│       │            │ │ZONE=h        │                             │
│       │            │ │ZONE=(h,m)    │                             │
│       │            │ │ZONE=(h,m,s)  │                             │
│       │            │ │ZONE=(h,,s)   │                             │
│       │            │ └              ┘                             │
│       │            │                                              │
│       │            │ ┌          ┐                                 │
│       │            │ │,LOC=EAST │                                 │
│       │            │ │,LOC=WEST │                                 │
│       │            │ └          ┘                                 │
│       │            │                                              │
│       │            │ ┌        ┐                                   │
│       │            │ │,ID=GMT │                                   │
│       │            │ │,ID=xxx │                                   │
│       │            │ └        ┘                                   │
└──────────────────────────────────────────────────────────────────┘
```

where:

```
┌              ┐
│ZONE=0        │
│ZONE=h        │
│ZONE=(h,m)    │
│ZONE=(h,m,s)  │
│ZONE=(h,,s)   │
└              ┘
```

is the time zone differential 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 differential must not exceed 13 hours.

The variable m is a number that represents minutes.

The variable s is a number that represents seconds.

```
┌         ┐
│LOC=EAST │
│LOC=WEST │
└         ┘
```

specifies whether the time zone differential 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=GMT│
│ID=xxx│
└      ┘
```

is the name of the time zone. The default for ID is GMT. The variable xxx is a 3-character string.

Examples:

The following examples show how to code the SYSTIME macro for several 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 is used to invoke daily automatic performance data collection with the VM Monitor. The IBM Field Developed Program 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.

The format of the SYSMON macro is:

Name	Operation	Operands
	SYSMON	[USERID=OPERATOR | USERID=userid] [,CLASS=M | ,CLASS=class] [,AUTO=NO | ,AUTO=YES] [,ENABLE=(PERFORM,USER,DASTAP) | ,ENABLE=(classa,classb,classc,...)] [,TIME=(09:00,17:00) | ,TIME=(h1:m1,h2:m2) | ,TIME=ALL | ,TIME=NONE] [,LIMIT=(50000,NOSTOP) | ,LIMIT=(limit,STOP) | ,LIMIT=(limit,NOSTOP) | ,LIMIT=(limit,SAMPLE)] [,BUFFS=cpu default | ,BUFFS=n]

where:

[USERID=OPERATOR | USERID=userid]

is the userid of the virtual machine that will receive the monitor spool file in its virtual reader. The default is OPERATOR but any valid system directory entry may be specified.

```
┌            ┐
|CLASS=M     |
|CLASS=class |
└            ┘
```

specifies the spool file to be generated to contain monitor data. Any valid class (A through Z and 0 through 9) may be used but the default M is preferred since the VMAP data reduction Field Developed Program is designed to reduce only spool files of that class.

```
┌        ┐
|AUTO=NO |
|AUTO=YES|
└        ┘
```

specifies whether or not automatic monitoring should take place according to the remaining SYSMON parameter specifications. The default, NO, requires the installation to make a specific change to cause automatic monitoring. All other parameters may be system default values, giving positive and useful monitoring results.

```
┌                                  ┐
|ENABLE=(PERFORM,USER,DASTAP)      |
|ENABLE=(classa,classb,classc,...) |
└                                  ┘
```

specifies any combination of valid monitor classes of data collection. It is assumed that the system analyst understands the use of the various classes, the overhead incurred in data collection, and the relative magnitude of the corresponding data reduction. The default specifies sampled data classes only and are considered minimal for useful data reduction. The default classes are sufficient for analysis of a system's load environment and performance profile with a view to diagnosis of possible bottlenecks and for establishing long term growth trends.

```
┌                     ┐
|TIME=(09:00,17:00)   |
|TIME=(h1:m1,h2:m2)   |
|TIME=ALL             |
|TIME=NONE            |
└                     ┘
```

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) 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, several hours of monitoring is suggested.

Note: This same closeout occurs at midnight if ALL is specified.

```
┌                        ┐
|LIMIT=(50000,NOSTOP)    |
|LIMIT=(limit,STOP)      |
|LIMIT=(limit,NOSTOP)    |
|LIMIT=(limit,SAMPLE)    |
└                        ┘
```

specifies the maximum number of monitor record buffers that can be added to the monitor spool file before it is closed, whether or not monitoring should be terminated 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 the installation more control over the amount of spool space that can be used by the automatic monitoring facility. It can also be used to create several small monitor spool files, rather than one large file and, for instance, give the data reduction facility an opportunity to start processing the morning's data while collecting the afternoon's data. 'limit' 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, the sampling interval and remember that each monitor buffer contains approximately 4000 bytes of data space.

Specifying SAMPLE allows the installation analyst to define the rate at which spool files will be produced. Since 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 and containing 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=cpu default|
|BUFFS=n          |
└                 ┘
```

specifies the number of data collection buffers needed by the monitor to avoid suspension occurrences. Data collection suspension occurs when output to tape or spool files cannot keep ahead of the collection of data and an overrun condition occurs. By increasing the number of monitor buffers the suspension occurrences can be reduced or eliminated. The default depends on the processor on which the system is running. (See the _VM/370 System Programmer's Guide_ description of the MONITOR command.) If the user is not satisfied with the defaults, he 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 or at system restart or shutdown. The spool file class is the default M.

Note: All of the above automatic monitoring specifications may be overridden by the operator or system analyst using the MONITOR command.

SYSJRL Macro

The SYSJRL macro is used to specify the inclusion of the journaling and/or password suppression facility.

Name	Operation	Operands
	SYSJRL	,JOURNAL=NO ,JOURNAL=YES
		,STQUERY=NO ,STQUERY=YES
		,LOGUID=OPERATOR ,LOGUID=userid
		,LOGLMT=(2,3,4) ,LOGLMT=(x,y,z)
		,LNKUID=OPERATOR ,LNKUID=userid
		,LNKLMT=(2,5,10) ,LNKLMT=(x,y,z)
		,PSUPRS=NO ,PSUPRS=YES

where:

JOURNAL=NO
JOURNAL=YES

indicates whether or not the journaling facility is to be operative in the system being generated.

STQUERY=NO
STQUERY=YES

indicates whether or not the ability to SET and QUERY the journaling function should be a part of the system being generated. YES may only be specified if JOURNAL=YES is also specified.

LOGUID=OPERATOR
LOGUID=userid

is the userid that should receive the indication that an invalid logon password count has been reached or exceeded.

```
┌            ┐
|LOGLMT=(2,3,4)|
|LOGLMT=(x,y,z)|
└            ┘
```

is the invalid LOGON/AUTOLOG password threshold specification. The value specified applies to a single userid for a single LOGON session. x is the value which, when reached or exceeded, causes a type 04 accounting record to be generated for that and each subsequent LOGON/AUTOLOG containing an invalid password. y is the value which, when reached or exceeded, causes a message to be sent to the userid specified by LOGUID for that and each subsequent LOGON/AUTOLOG containing an invalid password. z is the value which, when reached, causes the LOGON/AUTOLOG command to be disabled.

Note: z replaces the present fixed limit of 4 and may be any decimal from 1 to 255. x and y may be any decimal 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 would be generated.

```
┌               ┐
|LNKUID=OPERATOR|
|LNKUID=userid  |
└               ┘
```

is the userid that should receive the indication that an invalid link password count has been reached or exceeded.

```
┌              ┐
|LNKLMT=(2,5,10)|
|LNKLMT=(x,y,z) |
└              ┘
```

is the invalid LINK password threshold specification. The value specified applies to a single userid for a single LOGON session. x is the value that, when reached or exceeded, causes a type 06 accounting record to be generated for that and each subsequent LINK containing an invalid password. y is the value that, when reached or exceeded, causes a message to be sent to the userid specified by LNKUID for that and each subsequent LINK containing an invalid password. z is the value that, when reached, causes the LINK command to be disabled for the current LOGON session. This replaces the current fixed limit of 10 and may be any decimal digit from 1 to 255. x and y may be any decimal digit from 0 to 255. 0 is a special case which indicates the applicable function to be bypassed. For example, if LNKLMT=(2,0,10) is specified, no message records would be sent.

```
┌          ┐
|PSUPRS=NO |
|PSUPRS=YES|
└          ┘
```

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, the print suppress feature of the 2741 will not be used. Passwords will always be typed upon a mask.

SYSLOCS Macro

The SYSLOCS macro instruction is a required macro used to generate internal pointer variables. This must be the last macro in the DMKSYS deck.

This macro is required and 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.

The format of the SYSLOCS macro is:

Name	Operation	Operands
	SYSLOCS	

Example:

An example of the SYSLOCS macro is:

```
SYSLOCS
```

Creating Your VM/370 Directory

The VM/370 directory contains the entries of all potential virtual machines that are permitted to logon the VM/370 system. Without the proper directory entry, a user cannot log on to VM/370. The entries in the directory contain the user identification and password, the virtual machine I/O configuration, associated virtual and real addresses, disk usage values, virtual processor storage size, and other options. These options are discussed in the directory program control statement descriptions.

Each user in the directory, except those whose password is NOLOG, must have at least one device. Any of the various devices described meet this requirement; for example, the device may be a console or a spool device. The number of virtual devices for a virtual machine cannot exceed the value determined by (7FFF/VDEVSIZE), where VDEVSIZE is the size of the VDEVBLOK. If a greater number of virtual devices is specified, results may be undesirable.

The VM/370 directory usually resides on the VM/370 system residence disk, and is pointed to by the VOL1 label (cylinder 0, track 0, record 3). The VM/370 Directory program (module DMKDIR, invoked by the DIRECT command, or run standalone) processes the control statements you prepare and writes the VM/370 directory on disk. You already described your installation's real configuration when you created the real I/O configuration file. Now, you describe the many virtual configurations for your installation with the Directory program control statements.

To create a VM/370 directory, you must:

- Prepare the Directory program control statements
- Format and allocate the DASD space to contain the VM/370 directory
- Execute the Directory program

At this time, you should prepare the Directory program control statements. Later, during the system generation procedure, you must (1) format and allocate DASD space for the VM/370 directory and (2) generate it. The step-by-step description of the system generation procedure that is in Part 3 of this manual reminds you to create your VM/370 directory.

Considerations for Preparing the Directory Control Statements

First, prepare a directory control statement that defines the device on which the VM/370 directory is to be written. This statement (DIRECTORY) must be the first control statement in the input to the Directory program, and is followed by the sets of statements describing your installation's virtual machines.

Next, prepare Directory program control statements describing each virtual machine in your installation. The descriptions contain accounting data, options, and virtual machine configurations for each virtual machine that appears in the VM/370 directory. Information about coding these control statements is found in the section, "The Directory Program."

VM/370 does not check for overlapping extents; therefore, you must ensure that minidisk extents defined in the VM/370 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 and should define virtual machines for the system analyst or system programmer.

The operator's virtual machines should be able to control:

- The VM/370 sessions
- Allocation of machine resources
- Spooling activity
- Online disk areas

You should also define virtual machines for system analysts that are equipped to:

- Perform system analysis
- Modify certain VM/370 functions

and additional virtual machines to update or operate:

- The CP system
- The CMS system
- The RSCS system, if you generate one
- The hardware
- Other operating systems that run in the virtual machine environment
- The Installation Verification Procedure

SYSTEM SUPPORT VIRTUAL MACHINES

At system generation time, two additional virtual machines should be created beyond those needed by normal users (one each for hardware and software support). The IBM FE programming support representative should be consulted when the configurations for these virtual machines are being determined.

Hardware Support

The hardware support is for:

- The processor, which must be supported in a dedicated environment because there is no method currently available that allows concurrent support of the processor, real storage, or channels when executing problem programs.

- The input/output equipment, which can be supported using online test (OLT) under OLTSEP. The OLTSEP program can be executed in its own virtual machine.

Any of the offline testing capabilities of the 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/370 directory for the IBM service representative. The virtual machine should have enough virtual storage defined to execute OLTSEP. Normally, the service representative requires that the device being tested be dedicated to his virtual machine. (The system operator can dedicate devices to a virtual machine by issuing the ATTACH command.)

Also 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 utilized.

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 thus allowing editing and printing of all VM/370-recorded machine check and channel check errors.

This directory entry is included in the VM/370 directory provided with the starter system.

Software Support

The virtual machine for software support should have the minimum configuration necessary to recreate (virtually) problems that occur on the real machine. The ECMODE option must be specified in the directory OPTION control statement for this machine. You should assign privilege class G to the user of this machine. Also, you should assign privilege class E if he is to examine real storage addresses, and privilege class B if he is to allocate devices.

Sample Directory Entries

The following sample VM/370 directory entries provide an installation with some of the entries necessary for operation and updating. The indentations are for readability and are not required by the directory program. LINK control statements are used whenever possible to minimize the number of changes to the VM/370 directory whenever a minidisk extent is moved. A brief explanation of some of the virtual machine userids follows.

THE SYSTEM OPERATOR'S VIRTUAL MACHINE (OPERATOR)

The userid for this directory entry must be the same as the userid on the SYSOPER operand of the SYSOPR system generation macro. 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/370, the operator's virtual machine only needs class A command privileges. The MDISK control statement defines the 191 minidisk which contains CMS files, EXEC procedures, and service programs to update VM/370.

```
USER OPERATOR PASSWORD 256K 1M ABCDEG
     ACCOUNT ACCTNO BIN1
          CONSOLE 009 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          LINK VMSYS 190 190 RR
          MDISK 191 3330 1 10 UDISKA WR RPASS WPASS
```

A VIRTUAL MACHINE TO RECEIVE SYSTEM DUMPS (OPERATNS)

The userid for the following directory entry is the userid that was specified on the SYSDUMP operand of the SYSOPR macro when the VM/370 system was generated. All abnormal termination dumps are sent to this virtual machine. This user normally is given command privilege classes A, B, C, and E. If the directory entry contains all of the disks normally attached to the system, described as full-volume minidisks, the user can rewrite the VM/370 directory by using the DIRECT command. The operations group can also examine any disk while it is attached to the system, when these disks are defined as full-volume minidisks.

```
USER OPERATNS PASSWORD 320K 1M ABCEG
     ACCOUNT ACCTNO BIN2
          CONSOLE 009 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403 A
          LINK VMSYS 190 190 RR
          MDISK 191 3330 101 10 UDISKA WR RPASS WPASS
          MDISK 350 3330 0 404 SYSRES WR RPASS WPASS
          MDISK 351 3330 0 404 SYSWRK RR RPASS WPASS
          MDISK 250 3330 0 404 UDISK1 RR RPASS WPASS
          MDISK 251 3330 0 404 UDISK2 RR RPASS WPASS
          MDISK 232 2314 0 203 BATCH1 RR RPASS WPASS
          MDISK 233 2314 0 203 BATCH2 RR RPASS WPASS
          MDISK 234 2314 0 203 TSOSYS RR RPASS WPASS
          MDISK 235 2314 0 203 TSOWRK RR RPASS WPASS
```

Some installations may want to combine the functions of OPERATOR and OPERATNS into one virtual machine. This can be accomplished in the above examples by adding the last 8 control statements of the directory entry for OPERATNS to the directory entry for OPERATOR. The OPERATOR virtual machine can then perform all the system functions required to operate the VM/370 system.

Other System Virtual Machines

In addition to the virtual machines discussed up to this point, there are those virtual machines whose function is to:

- Support and update the VM/370 system
- Test new releases of the system before placing them into production status
- Provide the hardware service representative with the ability to run diagnostics and extract recorded error data
- Provide other users with a remote file spooling capability

A VIRTUAL MACHINE FOR UPDATING AND SUPPORTING VM/370 (VMSYS)

The following directory entry defines a virtual machine (VMSYS) that can support and update the VM/370 system. The 194 minidisk contains alterations or fixes to CP; these alterations and fixes are not applied until they have been thoroughly tested and considered stable. The 394 minidisk contains the distributed source files. The VMSYS virtual machine's privilege classes include class E and class G command privileges, so that it can issue the SAVESYS command to save CMS and other systems. The 190 minidisk contains the CMS nucleus and all of the CMS modules and EXEC procedures available to all CMS users. Any virtual machine that wants to use the CMS system, links to this disk (190). The 393 minidisk contains the distributed CMS source code in an unaltered form. The 193 minidisk holds CMS PTFs and updates.

```
USER VMSYS PASSWORD 512K 16M BCEG
     ACCOUNT ACCTNO BIN9
     OPTION ECMODE REALTIMER
          CONSOLE 01F 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          SPOOL 002 3211
          MDISK 190 3330 030 085 CPRnLO MR RPASS
          MDISK 191 3330 017 007 CPRnLO WR RPASS
          MDISK 194 3330 115 027 CPRnLO MR RPASS
          MDISK 199 3330 029 001 CPRnLO WR RPASS
          MDISK 193 3330 001 030 USERD1 MR RPASS
          MDISK 294 3330 031 030 USERD1 MR RPASS
          MDISK 393 3330 061 080 USERD1 MR RPASS
          MDISK 394 3330 141 090 USERD1 MR RPASS
          MDISK 390 3330 231 002 USERD1 MW RPASS
```

A HARDWARE SERVICE VIRTUAL MACHINE (SERV)

The following directory entry defines the virtual machine (SERV) that can be used by the hardware service representative. This virtual machine usually has class F command privileges. For more information on the hardware service virtual machine, see the publication VM/370 OLSTEP and Error Recording Guide.

```
USER SERV PASSWORD 256K 1M FG
     ACCOUNT ACCTNO BIN10
          CONSOLE 009 3215
          SPOOL 00C 2540 R
          SPOOL 00D 2540 P
          SPOOL 00E 1403
          LINK VMSYS 190 190 RR
          MDISK 191 3330 151 10 UDISKA WR RPASS WPASS
```

The VM/370 Directory Supplied with the Starter System

A control statement file for the VM/370 directory program is supplied with the:

- 2314 Starter System
- 3330 Starter System
- 3340 Starter System
- 3350 Starter System

If the supplied control statement file meets your needs, you can execute the Directory program using the supplied control statements. Otherwise, you can code your own control statements or edit the supplied control statements to produce the file you need for your installation.

2314 STARTER SYSTEM SUPPLIED DIRECTORY

The VM/370 Directory program control file supplied with the 2314 starter system is:

```
* CHANGE THE NEXT ENTRY FOR YOUR SYSTEM RESIDENCE DEVICE

 DIRECTORY XXX 2314 LABEL
*
USER OPERATOR OPERATOR 320K 1M ABCDEG
 ACCOUNT ACT1 OPERATOR
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 2314 008 007 CPRnL0[1] WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER CE CE 512K 1M EFG
 ACCOUNT ACT2 CE
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 2314 015 004 CPRnL0 WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER MAINT CPCMS 720K 16M BCEG
 ACCOUNT ACT3 MAINT
 OPTION ECMODE REALTIMER
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 190 2314 035 135 CPRnL0 MR READ
 MDISK 191 2314 019 010 CPRnL0 WR READ
 MDISK 194 2314 170 033 CPBnL0 MR READ
 MDISK 199 2314 034 001 CPBnL0 WR READ
*
*****
*
* MDISK XXX 2314 000 203 YYYYYY MW
*   THE ABOVE ENTRY SHOULD BE MODIFIED TO MATCH THE ADDRESS AND LABEL
*   OF YOUR SYSTEM RESIDENCE VOLUME. IT MAY THEN BE USED BY THIS ID
*   TO LOAD A DIRECTORY AND WRITE A CP NUCLEUS.
*   DELETE THE ' * ' (IN FRONT OF MDISK) ALSO.
*
*****
USER IVPM1 IVPASS 320K 16M G
 ACCOUNT ACT4 IVPM1
 CONSOLE 009 3210
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 2314 001 001 CPRnL0 WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
```

---
[1]CPRnL0 may be CPR4L0, CPR5L0, CPR6L0 and so forth, depending on the release level.

```
*
USER IVPM2 IVPASS 320K 1M G
 ACCOUNT ACT5 IVPM2
 CONSOLE 009 3210
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 2314 002 001 CPRnL0 WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER RSCS RSCS 512K
 ACCOUNT ACT6 RSCS
 OPTION ECMODE
 CONSOLE 009 3215
 SPOOL 001 2540 READER A
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 2314 003 005 CPRnL0 WR READ WRITE
 LINK MAINT 190 190 RR
 DEDICATE 0B1 078
 DEDICATE 0B2 079
 DEDICATE 0B3 07A
*
USER ECMODE ECMODE 512K 1M G
 ACCOUNT ACT7 ECMODE
 OPTION ECMODE REALTIMER
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 2314 029 005 CPRnL0 WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER OPERATNS OPERATNS 512K 1M BCEG
 ACCOUNT ACT8 OPERATNS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
**********
*
* THE FOLLOWING MINIDISK ENTRY IS PROVIDED AS AN EXAMPLE OF
* THE SPACE RECOMMENDED FOR AN IPCS VIRTUAL MACHINE.
* IF YOU INTEND TO USE THE OPERATNS USERID AS YOUR IPCS
* VIRTUAL MACHINE, YOU SHOULD CHANGE THE FOLLOWING STATEMENT
* TO ALLOCATE MINIDISK SPACE ON ONE OF YOUR SYSTEM DASD VOLUMES.
*
* MDISK 191 2314 XXX 010 YYYYYY WR READ WRITE
*
*
```

3330 STARTER SYSTEM SUPPLIED DIRECTORY

The VM/370 Directory program control file supplied with the 3330 starter system is:

```
* CHANGE THE NEXT ENTRY FOR YOUR SYSTEM RESIDENCE DEVICE
DIRECTORY XXX 3330 LABEL
*
USER OPERATOR OPERATOR 320K 1M ABCDEG
 ACCOUNT ACT1 OPERATOR
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3330 008 005 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER CE CE 512K 1M EFG
 ACCOUNT ACT2 CE
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3330 013 004 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER MAINT CPCMS 720K 16M BCEG
 ACCOUNT ACT3 MAINT
 OPTION  ECMODE REALTIMER
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 190 3330 030 085 CPRnLO MR READ
 MDISK 191 3330 017 007 CPRnLO WR READ
 MDISK 194 3330 115 027 CPRnLO MR READ
 MDISK 199 3330 029 001 CPRnLO WR READ
*
*****
* MDISK XXX 3330 000 404 YYYYYY MW
*   THE ABOVE ENTRY SHOULD BE MODIFIED TO MATCH THE ADDRESS AND LABEL
*   OF YOUR SYSTEM RESIDENCE VOLUME. IT MAY THEN BE USED BY THIS ID
*   TO LOAD A DIRECTORY AND WRITE A CP NUCLEUS.
*   DELETE THE ' * ' (IN FRONT OF MDISK) ALSO.
*   CHANGE THE '404' TO A '808' IF YOURS IS A 3330-11
*
*****
USER IVPM1 IVPASS 320K 16M G
 ACCOUNT ACT4 IVPM1
 CONSOLE 009 3210
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3330 001 001 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
```

```
*
USER IVPM2 IVPASS 320K 1M G
 ACCOUNT ACT5 IVPM2
 CONSOLE 009 3210
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3330 002 001 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER RSCS RSCS 512K
 ACCOUNT ACT6 RSCS
 OPTION ECMODE
 CONSOLE 009 3215
 SPOOL 001 2540 READER A
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3330 003 005 CPRnLO WR READ WRITE
 LINK MAINT 190 190 RR
 DEDICATE 0B1 078
 DEDICATE 0B2 079
 DEDICATE 0B3 07A
*
USER ECMODE ECMODE 512K 1M G
 ACCOUNT ACT7 ECMODE
 OPTION ECMODE REALTIMER
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3330 024 005 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER OPERATNS OPERATNS 512K 1M BCEG
 ACCOUNT ACT8 OPERATNS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
****************
*
* THE FOLLOWING MINIDISK ENTRY IS PROVIDED AS AN EXAMPLE OF
* THE SPACE RECOMMENDED FOR AN IPCS VIRTUAL MACHINE.
* IF YOU INTEND TO USE THE OPERATNS USERID AS YOUR IPCS
* VIRTUAL MACHINE, YOU SHOULD CHANGE THE FOLLOWING STATEMENT
* TO ALLOCATE MINIDISK SPACE ON ONE OF YOUR SYSTEM DASD VOLUMES.
*
* MDISK 191 3330 XXX 005 YYYYYY WR READ WRITE
*
****************
*
*
*
*     CYLINDERS 142 TO 403 ARE UNUSED AND MAY BE USED FOR
*     ANY OTHER VIRTUAL MINIDISK SPACE. IT CAN ALSO BE
*     USED FOR PAGING, SPOOLING OR T-DSK SPACE.
*
*
*
```

3340 STARTER SYSTEM SUPPLIED DIRECTORY

The VM/370 Directory program control statements supplied with the 3340 starter system is:

```
* CHANGE THE NEXT ENTRY FOR YOUR SYSTEM RESIDENCE DEVICE

DIRECTORY XXX 3340 LABEL
*
USER OPERATOR OPERATOR 320K 1M ABCDEG
 ACCOUNT ACT1 OPERATOR
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3340 011 010 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER CE CE 512K 1M EFG
 ACCOUNT ACT2 CE
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3340 021 005 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER MAINT CPCMS 720K 16M BCEG
 ACCOUNT ACT3 MAINT
 OPTION ECMODE REALTIMER
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 190 3340 048 240 CPRnLO MR READ
 MDISK 191 3340 026 015 CPRnLO WR READ
 MDISK 194 3340 288 060 CPRnLO MR READ
 MDISK 199 3340 046 002 CPRnLO WR READ
*
*****
*
* MDISK XXX 3340 000 348 XXXXXX MW
*   THE ABOVE ENTRY SHOULD BE MODIFIED TO MATCH THE ADDRESS AND LABEL
*   OF YOUR SYSTEM RESIDENCE VOLUME. IT MAY THEN BE USED BY THIS ID
*   TO LOAD A DIRECTORY AND WRITE A CP NUCLEUS.
*   DELETE THE ' * ' (IN FRONT OF MDISK) ALSO.
*   CHANGE THE '348' TO A '696' IF YOURS IS A 3340-70
*
*****
USER IVPM1 IVPASS 320K 16M G
 ACCOUNT ACT4 IVPM1
 CONSOLE 009 3210
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3340 001 002 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
```

```
*
USER IVPM2 IVPASS 320K 1M G
 ACCOUNT ACT5 IVPM2
 CONSOLE 009 3210
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3340 003 002 CPRnL0 WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
 USER RSCS RSCS 512K
 ACCOUNT ACT6 RSCS
 OPTION ECMODE
 CONSOLE 009 3215
 SPOOL 001 2540 READER A
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3340 005 006 CPRnL0 WR READ WRITE
 LINK MAINT 190 190 RR
 DEDICATE 0B1 078
 DEDICATE 0B2 079
 DEDICATE 0B3 07A
*
USER ECMODE ECMODE 512K 1M G
 ACCOUNT ACT7 ECMODE
 OPTION ECMODE REALTIMER
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3340 041 005 CPRnL0 WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER OPERATNS OPERATNS 512K 1M BCEG
 ACCOUNT ACT8 OPERATNS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
**********
*
* THE FOLLOWING MINIDISK ENTRY IS PROVIDED AS AN EXAMPLE OF
* THE SPACE RECOMMENDED FOR AN IPCS VIRTUAL MACHINE.
* IF YOU INTEND TO USE THE OPERATNS USERID AS YOUR IPCS
* VIRTUAL MACHINE, YOU SHOULD CHANGE THE FOLLOWING STATEMENT
* TO ALLOCATE MINIDISK SPACE ON ONE OF YOUR SYSTEM DASD VOLUMES.
*
* MDISK 191 3340 XXX 015 YYYYYY WR READ WRITE
```

3350 STARTER SYSTEM SUPPLIED DIRECTORY

The VM/370 Directory program control file supplied with the 3350 starter system is:

```
* CHANGE THE NEXT ENTRY FOR YOUR SYSTEM RESIDENCE DEVICE
DIRECTORY XXX 3350 LABEL
*
USER OPERATOR OPERATOR 320K 1M ABCDEG
 ACCOUNT ACT1 OPERATOR
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3350 006 003 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER CE CE 320K 1M EFG
 ACCOUNT ACT2 CE
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3350 009 002 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER MAINT CPCMS 720K 16M BCEG
 ACCOUNT ACT3 MAINT
 OPTION ECMODE REALTIMER
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 190 3350 021 035 CPRnLO MR READ
 MDISK 191 3350 011 005 CPRnLO WR READ
 MDISK 194 3350 056 009 CPRnLO MR READ
 MDISK 199 3350 020 001 CPRnLO WR READ
*
*****
*
* MDISK XXX 3350 000 555 YYYYYY MW
*
*   THE ABOVE ENTRY SHOULD BE MODIFIED TO MATCH THE ADDRESS AND LABEL
*   OF YOUR SYSTEM RESIDENCE VOLUME. IT MAY THEN BE USED BY THIS ID
*   TO LOAD A DIRECTORY AND WRITE A CP NUCLEUS.
*   DELETE THE ' * ' (IN FRONT OF MDISK) ALSO.
*
*****
USER IVPM1 IVPASS 320K 16M G
 ACCOUNT ACT4 IVPM1
 CONSOLE 009 3210
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3350 001 001 CPRnLO WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
```

```
*
USER IVPM2 IVPASS 320K 1M G
 ACCOUNT ACT5 IVPM2
 CONSOLE 009 3210
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3350 002 001 CPRnL0 WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER RSCS RSCS 512K
 ACCOUNT ACT6 RSCS
 OPTION ECMODE
 CONSOLE 009 3215
 SPOOL 001 2540 READER A
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3350 003 003 CPRnL0 WR READ WRITE
 LINK MAINT 190 190 RR
 DEDICATE 0B1 078
 DEDICATE 0B2 079
 DEDICATE 0B3 07A
*
USER ECMODE ECMODE 512K 1M G
 ACCOUNT ACT7 ECMODE
 OPTION ECMODE REALTIMER
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 MDISK 191 3350 016 004 CPRnL0 WR READ WRITE
 LINK MAINT 194 194 RR
 LINK MAINT 190 190 RR
*
USER OPERATNS OPERATNS 512K 1M BCEG
 ACCOUNT ACT8 OPERATNS
 CONSOLE 009 3215
 SPOOL 00C 2540 READER A
 SPOOL 00D 2540 PUNCH A
 SPOOL 00E 1403 A
 LINK MAINT 190 190 RR
****************
*
* THE FOLLOWING MINIDISK ENTRY IS PROVIDED AS AN EXAMPLE OF
* THE SPACE RECOMMENDED FOR AN IPCS VIRTUAL MACHINE.
* IF YOU INTEND TO USE THE OPERATNS USERID AS YOUR IPCS
* VIRTUAL MACHINE, YOU SHOULD CHANGE THE FOLLOWING STATEMENT
* TO ALLOCATE MINIDISK SPACE ON ONE OF YOUR SYSTEM DASD VOLUMES.
*
* MDISK 191 3350 XXX 003 YYYYYY WR READ WRITE
*
****************
*
*
*   CYLINDERS 065 TO 555 ARE UNUSED AND MAY BE USED FOR
*   ANY OTHER VIRTUAL MINI DISK SPACE.  IT CAN ALSO BE
*   USED FOR PAGING, SPOOLING OR T-DSK SPACE.
*
*
*
```

Allocating DASD Space for the VM/370 Directory

Before you create your VM/370 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 the cylinders to be used for the VM/370 directory. The cylinders must be allocated as DRCT. To calculate the total number of cylinders required, first calculate the total number of records used:

$$NR = \frac{NU}{169} + \frac{((NU + NM) \times 2) + \text{all other control statements}}{170}$$

where:

NR = total number of records used
NU = number of USER control statements
NM = number of MDISK control statements (except for temporary disks)

Then, calculate the number of cylinders (NC):

- For 3330: NC = NR/57
- For 2314,2319: NC = NR/32
- For 2305, 3340: NC = NR/24
- For 3350: NC = NR/120

Note: You should initially format and allocate space for two VM/370 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/370 directory is written, space is allocated from available cylinders, a full cylinder at a time, and a minimum of two cylinders are used for the VM/370 directory.

Once a new VM/370 directory is successfully written, cylinders used for the old directory (marked as temporarily allocated during directory creation) are marked as free. In this way, DASD space allocated for DRCT cylinders is freed and can be reused for the next directory creation. 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.

The Directory Program

The VM/370 directory program can be run under CMS (using the DIRECT command) or standalone. The standalone 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 fileid of "USER DIRECT". The DIRECT command loads the directory creation module. If no filename is specified, the program looks for a file named USER DIRECT. Otherwise, it looks for a file named filename 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/370 directory in the VOL1 label, and if it is updating the active system directory, it places the new directory in use by VM/370. You can print the new directory by issuing the CMS command PRINT USER DIRECT (or PRINT filename DIRECT).

The virtual machine executing 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/370 system residence volume, your virtual machine's current directory entry must have write access to the volume containing the current VM/370 directory.

Example: Assume that you have the following virtual machine for online directory modification.

```
USER UPDRCT PASSWORD 256K 1M ABC
  ACCOUNT NUMBER BIN2
    IPL CMS
    CONSOLE 009 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    LINK CMSSYS 190 190 R
    MDISK 330 3330  0 404 SYSRES WR RPASS WPASS
    MDISK 331 3330  0 404 SYSWRK WR RPASS WPASS
    MDISK 230 2314  0 203 UDISK1 RR RPASS WPASS
    MDISK 231 2314  0 203 UDISK2 RR RPASS WPASS
    MDISK 232 2314  0 203 BATCH1 RR RPASS WPASS
    MDISK 233 2314  0 203 BATCH2 RR RPASS WPASS
    MDISK 191 3330 26 010 VMDSK2 WR RPASS WPASS
```

Using the CMS EDIT command and its subcommands, you can create or modify a card-image file of the VM/370 directory input. When you are ready to write a new directory, issue the command:

```
DIRECT  filename
```

where filename is a CMS file (normally named USER) with filetype DIRECT containing the necessary Directory program control statements. The DIRECT command puts this file into the form of a directory, and replaces the old directory with this new one.

Loading the DMKDIR object deck via the card reader is the same as issuing 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.

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 (via IPL).

INVOKING THE DIRECTORY PROGRAM (DMKDIR) UNDER CMS

The VM/370 Directory program records the configuration of each user's virtual machine in the VM/370 directory. Each virtual machine configuration includes counterparts of the components found in a real System/370: 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 the CMS Editor to include additional directory entries.

The CMS DIRECT Command

Use the CMS DIRECT command to process any file to see if it follows the required directory format. To actually change or swap the currently active VM/370 directory, you must have both of the following:

1. User class A, B, or C.
2. 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 qualifications 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/370 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/370 directory each time it is created.

If you execute the Directory program under VM/370, the newly created directory is dynamically swapped, and placed in use by VM/370 (provided that you have class A, B, or C and that the directory you updated is the one that is currently in use by the system). If you do not have the proper privilege class, the directory is updated on the directory volume but not dynamically swapped, so the change will not go into effect until the next time the system is loaded (via IPL). The format of the DIRECT command is:

```
┌─────────────────────────────────────────────────────────────────────────────┐
│            │ ┌            ┌           ┌         ┐┐┐                          │
│  DIRECT    │ │filename    │filetype   │filemode│││  [(EDIT)]                 │
│            │ │USER        │DIRECT     │   *    │││                           │
│            │ └            └           └         ┘┘┘                          │
└─────────────────────────────────────────────────────────────────────────────┘
```

<u>where:</u>

filename [filetype [filemode]]
is the identification of the file containing the control statements for the Directory program. If no filename and filetype are given, the program defaults to a file named USER DIRECT; otherwise, it looks for the file named. The filetype must be DIRECT. If only filename is given, filetype defaults to DIRECT. The filemode defaults to * if not specified.

(EDIT) specifies that the directory is to be examined, but not changed.

Under CMS, the DIRECT command loads the directory creation module. The first statement encountered must be a DIRECTORY statement. If not found, or another DIRECTORY statement is found, the program terminates. 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 terminates, the old directory is not altered. Normal completion places the directory in use by VM/370. After the new directory is created, it can be printed by issuing the CMS command PRINT USER DIRECT or PRINT filename DIRECT.

The DIRECT command filename and filetype default to a CMS file identification of USER DIRECT. The filemode defaults to * if not specified. Any or all of the defaults can be overridden by the command line. The EDIT option allows you to run the program without updating the directory on disk. This enables you to check the syntax of the directory statements without accessing the directory disk.

INVOKING DIRECTORY AS A STANDALONE PROGRAM

Standalone 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.

DIRECTORY CONTROL STATEMENTS

The control statements should be in the following formats, with one or more blanks as operand delimiters. All operands are positional from left to right. If any operands are omitted, all 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 found to be in error, the program continues to process the control statements, validating all control statements before terminating. If the directory runs out of space, the program terminates immediately. After an abnormal termination (or, for CMS, the EDIT run), 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 directory is allocated. It must be the first statement. The format of the DIRECTORY control statement is:

```
DIRectory  cuu  devtype  volser
```

where:

cuu is the address of the device that is to contain the directory and is specified in three hexadecimal digits.

devtype is four decimal digits that represent a supported device type suitable for the VM/370 directory (2314, 2319, 2305, 3330, 3340 or 3350). 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/370 directory.

volser is the volume serial number of the directory volume (1 to 6 alphameric characters).

USER Control Statement

The USER control statement defines a virtual machine and creates a VM/370 directory entry. It delimits the directory entry for one user. A separate USER statement must be prepared for each directory entry required. The format of the USER control statement is:

```
                                         ┌    ┌    ┌    ┌    ┐┐┐┐
User userid pass [stor [mstor [cl [pri  |le  |ld  |cd  |es  ||||]]]]
                                        |ON  |ON  |ON  |ON  ||||
                                        |OFF |OFF |OFF |OFF ||||
                                         └    └    └    └    ┘┘┘┘
```

where:

userid is a 1- to 8-character user identification. Any alphameric characters may be used except SYSTEM. SYSTEM is the userid of the VM/370 system VMBLOK, and should never be used for a virtual machine. Each user in the directory, except for those whose password is NOLOG, must have at least one device. Any of the various devices described meet this requirement; for example, the device may be a console or spool device.

Notes:

1. The userid should not contain the characters "LOGONxxx", where xxx is a terminal address of the installation. This character string is assigned to the terminal at address xxx from the time the initial interrupt is received until the user is identified, during logon.

2. Do not specify SYSTEM as a userid. VM/370 reserves SYSTEM as an identifier for its own use. Similarly, do not use ALL as a userid as it is reserved by VM/370.

3. If the userid of AUTOLOG1 (a reserved system user identification) is used, then during the VM/370 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, will initiate the logon of other virtual machines to the system.

pass is a 1- to 8-character user-security password that must be entered by the user to gain access to the VM/370 system and the virtual machine you are defining in these control statements.

Note: Use the reserved password NOLOG for users who do not have a virtual machine configuration in the VM/370 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 log on using this password are inhibited.

stor is 1 to 8 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 128K. 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 1 to 8 decimal digits that define the maximum virtual machine storage size that this user can define as his storage 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 1 to 8 alphabetic characters from A to H (with no intervening blanks) defining the privilege class(es) given to this user. The default is G.

Note: If privilege class F is assigned to a virtual machine, I/O error recording is not automatically done. This allows the class F user to set the kind of error recording he wants to perform.

pri is a number from 1 to 99 used by the control program priority dispatcher. One is the highest priority and 50 is the default.

Note: The same priority value can be used for several users. Also, if the specification for this statement is not entered, then line end (le), line delete (ld), character delete (cd), and escape (es) characters default to system-defined values.

The following special VM/370 logical editing symbols may be set ON, OFF, or substituted with two hexadecimal characters or one graphic character of the user's choice.

Note: In addition to the directory specification, the user 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 initiated 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.

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. However, if this statement is omitted, both the account number and the distribution code default to the userid. This statement (if coded) must follow the USER statement and precede the first device statement. The format of the ACCOUNT control statement is:

```
Account  number  [distribution]
```

where:

number — is a one- to eight-character account number that is punched in the accounting data for this virtual machine. The USERID 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 userid in the separator for spooled output for this user. This value is optional and defaults to the userid from the USER statement if omitted.

OPTION Control Statement

The OPTION control statement selects specific options available to the user. This statement is optional and, if used, must follow the USER statement or another OPTION statement, and precede the first device statement (CONSOLE, MDISK, DEDICATE, LINK, or SPOOL). Multiple OPTION statements can be inserted if the options selected exceed the statement record length. The format of the OPTION control statement is:

```
Option  Realtimer  Ecmode  Isam  Virt=real  Acct  Svcoff   BMX
        CPUID bbbbbb   AFFinity nn
```

where:

REALTIMER — provides a timer for the virtual machine that is updated during virtual processor run time and also 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 issuing the CP command line SET TIMER REAL.

ECMODE — allows the virtual machine to run in extended control mode. The ECMODE option must be specified for virtual machines using operating systems that:

1. Operate in System/370 extended control mode (such as VM/370 itself).
2. Use the dynamic address translation facility (such as OS/VS1, OS/VS2, DOS/VS, and VM/370).
3. Use control registers other than zero (such as OS GTF (General Trace Facility), which uses Monitor Call and requires control register eight).
4. 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, and for an RSCS virtual machine. ECMODE is also required when using the clock comparator.

Note: A virtual machine defined without the ECMODE option in the directory is limited to 6 I/O channels, while a virtual machine with the ECMODE option may address up to 16 I/O channels. If a virtual machine with the ECMODE option executes 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 executes in extended control mode, the I/O masking for all 16 channels is handled via extended control register 2. This facility can also be obtained by issuing the CP command SET ECMODE ON.

ISAM provides special channel command word translation routines that permit OS/PCP, MFT, and MVT ISAM programs (which 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 executing in a V=R partition under OS/VS. This option is not needed for DOS, DOS/VS, or OS/VS ISAM when run only in a V=V partition of OS/VS. This facility can also be obtained by issuing the CP command SET ISAM ON.

VIRT=REAL is a performance option that allows the user to place his virtual machine in lower storage, such that its virtual storage addresses correspond to the real storage address (except for its page zero, which is relocated). The real page zero 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/370 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 given time. A named or shared system cannot be loaded (via IPL) in a virtual=real area. The device address must be specified in the IPL command. To generate a VM/370 system with a virtual=real machine, see "Specifying a Virtual=Real Machine" in the Part 1.

ACCT a user with the ACCT option in his directory can charge another user for virtual machine resources. For example, a user who sends a job to the CMS batch virtual machine can be charged for the time that he uses in the batch machine. Note that the ACCT option should be specified in the directory of the CMSBATCH virtual machine so that user/job identifying information will be printed on the forms separator that separates spooled output files.

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 issuing SET ASSIST SVC.

Note: All SVC 76 interrupts are handled by CP 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. Channels that have a channel-to-channel adapter are restricted to selector channel operation.

The channel mode setting for all channels except virtual channel zero can be changed by the use of the CP DEFINE CHANNEL command.

CPUID bbbbbb

provides a unique processor identification (CPUID) to be stored in response to the STIDP instruction. It is necessary to associate a unique CPUID with each virtual machine that is attached to an MSC port since solicited/unsolicited messages are directed to the host system in the virtual environment by means of the CPUID. There is no checking by VM/370 to ensure that all virtual machines using the SET CPUID command have specified unique 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/370.

bbbbbb is up to 6 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 (0-9). 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' since CP does not reflect machine checks to the virtual machine.

If the CPUID was not specified by means of the SET CPUID command or the OPTION control statement, the CPUID stored as a result of the STIDP instruction is the real CPUID with the first two digits set to X'FF' and the last four digits 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 less than six hexadecimal digits results in zeros being padded to the left of the number. A three-byte field in the VMBLOK (VMCPUID) contains the value set as a result of invoking this DIRECTORY option.

AFFINITY nn

is 2 decimal digits between 00 and 63 that specify that virtual machine execution is to be performed on a designated processor (nn). This attribute is only applicable in the VM/370 attached processor environments. Any hexadecimal value from 00 to 3F is a valid main or attached processor address; however, the value selected must match the preset values established for your installation's main and attached processor when the system was installed. If the AFFINITY option is not selected, then the virtual machine is serviced by the first available processor from the VM/370 dispatch queue. An affinity setting in the VM/370 directory can be overridden by the CP SET AFFINITY command. If the system is running in attached processor mode and an error forces recovery to uniprocessor mode, the affinity setting of virtual machines assigned to the attached processor is nullified and virtual machine processing may be continued on the main processor.

IPL Control Statement

The IPL control statement contains a one- to eight-character name of the system (or one- to three-digit I/O device address) to be loaded for the user when he logs on. This statement is optional; if specified, it must follow the USER statement, and must precede the first device statement (CONSOLE, MDISK, or SPOOL). The IPL statement can be overridden by the user at logon time by specifying "LOGON userid NOIPL".

Note: If the user is the primary system operator, an automatic IPL is not performed when he logs on.

The format of the IPL statement is:

```
Ipl   iplsys
```

where:

iplsys is a one- to eight-character system name or the virtual address of the device containing the system to be loaded.

CONSOLE Control Statement

The CONSOLE control statement specifies the virtual console. The format of the CONSOLE control statement is:

```
Console  cuu  devtype [class]
```

where:

cuu is the virtual device address of one to three hexadecimal digits.

devtype is the device type:
1052
3210
3215

Note: The system accepts any of the devtypes indicated regardless of the real console or terminal being used. Device types 3275, 3276, 3277, 3278, 3036, 3066, 3138, 3148, 3158, 2741, and 3767 cannot be specified. Only one console can be specified. If a different console is sometimes required, use the CP DEFINE command to change the console address or add an alternate console.

class — is a one-character spooling class. A through Z and 0 through 9 are valid. The class governs the printing of the real spooled output. If the class operand is omitted, the default is class T and is for console spooling.

For more information about defining consoles, including a tutorial discussion, see <u>VM/370 Operating Systems in a Virtual Machine</u>.

<u>MDISK Control Statement</u>

The MDISK control statement describes the cylinder extent on a direct access device to be owned by the user. The DASD area assigned with this statement becomes the user's minidisk.

<u>Caution:</u> Neither CP nor the directory checks that minidisks defined with the MDISK statement do not overlap each other and (for 3330, 3340, and 3350 disks) that they do not overlap the "alternate track" cylinders at the end of the real disk. If overlap occurs, file damage is inevitable.

The format of the MDISK control statement is:

```
Mdisk cuu devtype {cylr    cyls volser [mode [pr [pw [pm]]]]}
                  {T-DISK  cyls                              }
```

<u>where:</u>

cuu — is the virtual device address of 1 to 3 hexadecimal digits.

devtype — is the device type:

```
2305
2311  Top     (Top half of a 2314 or 2319)
2311  Bottom  (Bottom half of a 2314 or 2319)
2314
2319
3330
3340
3350
```

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.

{cylr / T-DISK} — is a three-digit decimal cylinder relocation factor that specifies the cylinder on a real disk which 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 he logs on and is a part of his virtual configuration until he logs off or detaches the disk, at which time the data area is returned for reallocation for another T-DISK area. To maintain security this area should be physically erased before it is returned.

Note: It is not advisable to define that a minidisk start at real cylinder zero (unless the minidisk is to be used by OS ISAM, in which case it must begin at real cylinder zero). If you do assign a minidisk beginning at real cylinder zero, the user who owns it must realize that the minidisk label is the real label that both he and the VM/370 system use to identify the disk. Also note that CP-owned volumes must not have minidisks beginning at real cylinder zero.

cyls is a 1- to 3-digit decimal number specifying the number of cylinders.

Maximum Minidisk Sizes (cylinders)

Disk Type	DOS and VSAM under CMS	CMS	OS/VS
2314	200	203	200
2319	200	203	200
3330-1 or 2	404	246	404
3330-11	808	246	808
3340-35	348	348	348
3340-70	696	682	696
*3344	696	682	696
3350	555	115	555

*Note: The number of cylinders indicated for the 3344 is for each of the four logical 3340-70 devices.

If the device is a 2314 or 2319 and it is to be formatted by IBCDASDI, then the minimum minidisk size is two cylinders because, for these devices, IBCDASDI reserves a cylinder at the end of every minidisk for alternate tracks. For other devices, the minimum size is one cylinder.

volser is the volume serial number of the DASD volume (1 to 6 alphameric characters).

mode is the primary access mode requested for the device (read-only, write, or multiple-write), and the alternate access (read-only or write) desired (if any). An optional 'V' character, when appended to the mode request, specifies virtual RESERVE/RELEASE processing.

R specifies that read-only (R/O) access is requested. The access is not given if any other user has the disk in write status.

RR specifies that read-only access is requested, even if another user has the disk in write status.

W specifies that write access is requested. The disk is not defined if any other user has the disk in read or write status.

WR specifies that write access is requested if no other user has the disk in read or write status, but that an alternate access of read-only is acceptable if others do have a link to the disk.

M specifies that multiple access is requested. This means that a write link is to be given to the disk unless another user already has write access to it, in which case no link is to be given.

MR specifies that a write link is to be given to the disk unless another user already has write access to it. In this case, a read link is given to the user and the "DEV xxxx FORCED R/O" message is issued.

MW specifies that a write link is to be given to the disk in any case.

V specifies that CP's virtual reserve/release support is to be used in the I/O operations for the specified device. The V is appended to the immediate right of the primary access mode specification (or the alternate access mode specification, if any). Thus, if the mode specified for a minidisk is MWV, then the minidisk will function with write linkage using CP's virtual reserve/release function.

If a mode specification is omitted from the statement, it defaults to W.

pr is the password that allows sharing in read mode (a 1- to 8-character field).

pw is the password that allows sharing in write mode (a 1- to 8-character field).

pm is the password that allows sharing in multiple-write mode (a 1- to 8-character field).

Notes:

1. A write password (pw) cannot be specified without a read password (pr); a multiple-write password (pm) cannot be specified without both a read password (pr) and a write password (pw).

2. If a password of ALL is used for pr, pw, or pm, a user other than the owner of the minidisk is allowed to link 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 long.

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, then VM/370 will attempt 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 the minidisk if the minidisk is reserved by another virtual machine.

Examples:

MDISK 230 3330 5 10 WORK01 W ALL WRITE

is an MDISK statement for a minidisk with read/write access to 10 cylinders located on a real 3330 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 2314 50 15 CPDSK4 W RDPASS WRX2*

is an MDISK statement for a minidisk with read/write access to 15 cylinders located on a real 2314 labeled CPDSK4 starting at cylinder 50. A read password of RDPASS and a write password of WRX2* are provided. This allows the other users to access the minidisk through the directory LINK statement (see the description of the LINK statement in this section) or the LINK command.

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. The format of the SPOOL control statement is:

```
Spool  cuu devtype  [class]
```

where:

cuu is the virtual device address (1- to 3-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:

PRINTER 00E
PUNCH 00D
READER 00C

devtype is the device type:

1403
2501
1443
3203
3211
2540 R[EADER]
2540 P[UNCH]
3525
3505

class is a 1-character spooling class. The characters A through Z, 0 through 9, and * are valid. For spool output devices, the class governs the punching or printing of the real spooled output. If this operand is omitted, the default class A is used. This operand is required for all output devices defined on the spool record. For spool input devices, the class controls access to spool files by virtual card readers. The default class for readers is an asterisk (*), which 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.

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 via the DEDICATE statement. If the device is a unit record device, input and output are not spooled by VM/370. 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. The format of the DEDICATE control statement is:

```
| Dedicate  cuu { rdev |[VOLID] volser }  [R/O] [3330V]  |
```

where:

cuu is the virtual device address (1- to 3-hexadecimal digits).

rdev is the real device address (1- to 3-hexadecimal digits).

VOLID is the required keyword used if the volser is less than four characters long. It is optional if volser is four, five, or six characters long.

If the VOLID operand is used, the volume must be attached to the system when the user logs on. When he 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 (1- to 6-alphameric characters) or of an MSS volume to be dedicated to the virtual machine.

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.

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.

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 or detach a dedicated 2305 from a user, all 8 exposures are processed.

2. Use caution in defining the hexadecimal addresses of virtual devices (cuu) in DEDICATE statements, in order to avoid a usage conflict caused by control unit I/O interface protocol. The following is an example of a virtual machine's DEDICATE statements that can cause operational conflict.

```
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. Thus, the subsequent processing of a channel program involving these devices can result in a hung or busy condition (caused by a conflict in real-to-virtual I/O processing through the common VCUBLOK). Therefore, when defining devices, make sured 11 the devices are defined (and separated) within their own control unit range and not shared with other 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.

Since 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.

This restriction also applies to SPOOL statements and combinations of DEDICATE and SPOOL statements.

3. When the real device is a 3330V, the action VM/370 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 cuu 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/370, transparently to the virtual machine.

DED cuu rdev 3330V

The real device must again be a VIRTUAL 3330V. All cylinder faults and unsolicited interrups received by VM/370 on the rdev will be passed to the virtual machine.

DED cuu 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 cuu. 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 is used to dedicate volumes to non-MSS operating systems, such as CMS, since the control program chooses the real device address and no cylinder fault interrupts are passed to the virtual machine.

DED cuu rdev volser

The difference between this example and the one immediately preceding is that in this case the real device address is preselected and must have the VIRTUAL feature. This format allows the installation to 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 cuu rdev volser 3330V

This format is the same as the preceding, except that the virtual device becomes a 3330V, such that VM/370 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 F: VM/370 Restrictions.")

5. When dedicating a real CTCA, the CTCA should be on a separate real channel from all other virtual devices because of a possible lock-out problem.

LINK Control Statement

The LINK control statement makes a device that belongs to another user (userid) available to this virtual machine at logon time. If you want to make one volume available to several 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.

Then, if you later must move or change that volume, you need only update the one MDISK statement, the LINK statements need not be updated. The format of the LINK control statement is:

```
Link  userid  ldev  [cuu [mode]]
```

where:

userid is the 1- to 8-character user identification of the user to be linked to.

ldev is the virtual device address of the device owned by userid to be linked to (3 hexadecimal digits). This is the virtual device address, assigned by userid, of the disk you wish to link to.

cuu is the virtual device address for the virtual machine being defined. "cuu" defaults to the same address as the linked-to device (3-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 primary access mode requested for the device (read-only, write, or multiple-write), and the alternate access (read-only or write) desired, if any, as follows:

R specifies that read-only (R/O) access is requested. The link is not given if any other user has the disk in write status.

RR specifies that read-only access is requested, even if another user has the disk in write status.

W specifies that write access is requested. The disk is not defined if any other user has the disk in read or write status.

WR specifies that write access is requested if no other user has the disk in read or write status, but that an alternate access of read-only is acceptable if others do have a link to the disk.

M specifies that multiple access is requested. This means that a write link is to be given to the disk unless another user already has write access to it, in which case no link is to be given.

MR specifies that a write link is to be given to the disk unless another user already has write access to it. In this case, a read link is to be given to the user, and the "DEV xxxx FORCED R/O" message is issued.

MW specifies that a write link is to be given to the disk in any case.

Note: If the mode is not specified, the default is R.

It is the responsibility of the operating system executing in each virtual machine to keep data from being destroyed or altered on shared disks.

CMS supports multiply-accessed read-only disks in full. CMS does not support write access to disks by multiple users. 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.

CMS disks should never be linked in write mode to more than one user. If two or more CMS users have write access to the same disk, all data on the disk may be destroyed.

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 off. The format of the SPECIAL control statement is:

```
SPEcial  cuu  devtype [IBM|Tele]
```

where:

cuu is a 1- to 3-character virtual device address.

devtype is the device type:

```
2701
2702
2703
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)
```

IBM
TELE valid only if devtype is 2701, 2702, or 2703

For example, a virtual machine executing 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 to this multiple-access system rather than logging on as a separate virtual machine.

Note: The Integrated Communications Attachment (ICA) on the System/370 Models 135, 135-3, or 138 should be specified as a 2701.

DIRECTORY ENTRIES FOR CMS/DOS

The DOS/VS system and private libraries are accessed in read-only mode under CMS/DOS. If more than one CMS virtual machine is using the CMS/DOS environment, you should update the VM/370 directory entries so that the DOS/VS system residence volume and the DOS/VS private libraries are shared by all the CMS/DOS users.

The VM/370 directory entry for one of the CMS virtual machines should contain the MDISK statements defining the DOS/VS volumes. The VM/370 directory entries for the other CMS/DOS users should contain LINK statements.

For example, assume the DOS/VS system libraries are on cylinders 0-149 of a 3330 volume labeled DOSRES. Also, assume the DOS/VS private libraries are on cylinders 0-99 of a 2314 volume labeled DOSPRI. Then one CMS machine (for example, DOSUSER1) would have the MDISK statements in its directory entry.

```
USER DOSUSER1 password 320K 2M G
    .
    .
    .
    MDISK 331 3330 0 150 DOSRES R rpass
    MDISK 231 2314 0 100 DOSPRI R rpass
```

All the other CMS/DOS users would have links to these disks. For example

```
LINK DOSUSER1 331 331 R rpass
LINK DOSUSER1 231 231 R rpass
```

For more information about directory entries for CMS/DOS virtual machines, see the *VM/370 Operating Systems in a Virtual Machine*.

Preparing the System Name Table File (DMKSNT)

The system name table consists of entries that identify the name and location of saved systems. Three 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 for a 3704/3705 control program.
- The NAME3800 macro creates an entry in the system name table for a 3800 named system.

A system name table is supplied for each starter system. The DMKSNT module supplied with the 2314 starter system is:

```
DMKSNTBL CSECT
CMS      NAMESYS SYSSIZE=256K,SYSNAME=CMS,                                X
               VSYSADR=190,SYSVOL=VMRELn¹,SYSCYL=035,SYSSTRT=(001,1),   X
               SYSPGCT=33,SYSPGNM=(0-32),SYSHRSG=(1),VSYSRES=CPRnL0¹

CMSSEG   NAMESYS SYSNAME=CMSSEG,SYSVOL=VMRELn,SYSCYL=,                    X
               SYSSTRT=(002,03),SYSPGCT=(16),SYSHRSG=(16),                X
               SYSPGNM=(256-271),SYSSIZE=64K,VSYSRES=,VSYSADR=IGNORE

CMSVSAM  NAMESYS SYSNAME=CMSVSAM,SYSVOL=VMRELn,SYSPGNM=(272-367),         X
               SYSSTRT=(002,20),SYSPGCT=96,SYSSIZE=384K,SYSCYL=,          X
               SYSHRSG=(17,18,19,20,21),VSYSRES,VSYSADR=IGNORE

CMSAMS   NAMESYS SYSNAME=CMSAMS,SYSVOL=VMRELn,SYSPGNM=(368-495),          X
               SYSSTRT=(005,21),SYSPGCT=128,SYSSIZE=448K,SYSCYL=,         X
               SYSHRSG=(23,24,25,26,27,28),VSYSRES=,VSYSADR=IGNORE

CMSDOS   NAMESYS SYSNAME=CMSDOS,SYSVOL=VMRELn,SYSHRSG=(31),               X
               SYSSTRT=(009,22),SYSPGCT=8,SYSSIZE=32K,SYSCYL=,            X
               SYSPGNM=(496-503),VSYSRES=,VSYSADR=IGNORE

INSTVSAM NAMESYS SYSNAME=INSTVSAM,SYSVOL=VMRELn,SYSCYL=,                  X
               SYSSTRT=(009,31),SYSPGCT=8,SYSSIZE=32K,SYSHRSG=(254),      X
               SYSPGNM=(4064-4071),VSYSRES=,VSYSADR=IGNORE
         END
```

The DMKSNT module supplied for the 3330 starter system is:

```
CMS      NAMESYS SYSSIZE=256K,SYSNAME=CMS,                                X
               VSYSADR=190,SYSVOL=VMRELn,SYSCYL=030,SYSSTRT=(001,1),      X
               SYSPGCT=33,SYSPGNM=(0-32),SYSHRSG=(1),VSYSRES=CPRnL0

CMSSEG   NAMESYS SYSNAME=CMSSEG,SYSVOL=VMRELn,SYSCYL=,                    X
               SYSSTRT=(001,35),SYSPGCT=16,SYSHRSG=(16),                  X
               SYSPGNM=(256-271),SYSSIZE=64K,VSYSRES=,VSYSADR=IGNORE
```

---

[1] n may be 4, 5, 6 and so forth, depending on the Release level.

```
CMSVSAM  NAMESYS SYSNAME=CMSVSAM,SYSVOL=VMRELn,SYSPGNM=(272-367),        X
               SYSSTRT=(001,52),SYSPGCT=96,SYSSIZE=384K,SYSCYL=,         X
               SYSHRSG=(17,18,19,20,21),VSYSRES=,VSYSADR=IGNORE

CMSAMS   NAMESYS SYSNAME=CMSAMS,SYSVOL=VMRELn,SYSPGNM=(368-495),         X
               SYSSTRT=(003,35),SYSPGCT=128,SYSSIZE=448K,SYSCYL=,        X
               SYSHRSG=(23,24,25,26,27,28),VSYSRES=,VSYSADR=IGNORE

CMSDOS   NAMESYS SYSNAME=CMSDOS,SYSVOL=VMRELn,SYSHRSG=(31),              X
               SYSSTRT=(005,50),SYSPGCT=8,SYSSIZE=32K,SYSCYL=,           X
               SYSPGNM=(496-503),VSYSRES=,VSYSADR=IGNORE

INSTVSAM NAMESYS SYSNAME=INSTVSAM,SYSVOL=VMRELn,SYSCYL=,                 X
               SYSSTRT=(006,2),SYSPGCT=8,SYSSIZE=32K,SYSHRSG=(254),      X
               SYSPGNM=(4064-4071),VSYSRES=,VSYSADR=IGNORE
         END
```

The DMKSNT module supplied for the 3340 starter system is:

```
CMS      NAMESYS SYSSIZE=256K,SYSNAME=CMS,                               X
               VSYSADR=190,SYSVOL=VMRELn,SYSCYL=040,SYSSTRT=(001,1),     X
               SYSPGCT=33,SYSPGNM=(0-32),SYSHRSG=(1),VSYSRES=CPRnL0

CMSSEG   NAMESYS SYSNAME=CMSSEG,SYSVOL=VMRELn,SYSCYL=,                   X
               SYSSTRT=(002,11),SYSPGCT=16,SYSHRSG=(16),                 X
               SYSPGNM=(256-271),SYSSIZE=64K,VSYSRES=,VSYSADR=IGNORE

CMSVSAM  NAMESYS SYSNAME=CMSVSAM,SYSVOL=VMRELn,SYSPGNM=(272-367),        X
               SYSSTRT=(003,04),SYSPGCT=96,SYSSIZE=384K,SYSCYL=,         X
               SYSHRSG=(17,18,19,20,21),VSYSRES=,VSYSADR=IGNORE

CMSAMS   NAMESYS SYSNAME=CMSAMS,SYSVOL=VMRELn,SYSPGNM=(368-495),         X
               SYSSTRT=(007,5),SYSPGCT=128,SYSSIZE=448K,SYSCYL=,         X
               SYSHRSG=(23,24,25,26,27,28),VSYSRES=,VSYSADR=IGNORE

CMSDOS   NAMESYS SYSNAME=CMSDOS,SYSVOL=VMRELn,SYSHRSG=(31),              X
               SYSSTRT=(012,14),SYSPGCT=8,SYSSIZE=32K,SYSCYL=,           X
               SYSPGNM=(496-503),VSYSRES=,VSYSADR=IGNORE

INSTVSAM NAMESYS SYSNAME=INSTVSAM,SYSVOL=VMRELn,SYSCYL=,                 X
               SYSSTRT=(012,23),SYSPGCT=8,SYSSIZE=32K,SYSHRSG=(254),     X
               SYSPGNM=(4064-4071),VSYSRES=,VSYSADR=IGNORE
         END
```

The DMKSNT module supplied for the 3350 Starter System is:

```
DMKSNTBL CSECT
CMS      NAMESYS SYSSIZE=256K,SYSNAME=CMS,                               X
               VSYSADR=190,SYSVOL=VMRELn,SYSCYL=021,                     X
               SYSSTRT=(001,1),SYSPGCT=33,SYSPGNM=(0-32),SYSHRSG=(1),    X
               VSYSRES=CPRnL0

CMSSEG   NAMESYS SYSNAME=CMSSEG,SYSVOL=VMRELn,SYSCYL=,                   X
               SYSSTRT=(001,35),SYSPGCT=16,SYSHRSG=(16),                 X
               SYSPGNM=(256-271),SYSSIZE=64K,VSYSRES=,VSYSADR=IGNORE

CMSVSAM  NAMESYS SYSNAME=CMSVSAM,SYSVOL=VMRELn,SYSPGNM=(272-367),        X
               SYSSTRT=(001,52),SYSPGCT=96,SYSSIZE=384K,SYSCYL=,         X
               SYSHRSG=(17,18,19,20,21),VSYSRES=,VSYSADR=IGNORE

CMSAMS   NAMESYS SYSNAME=CMSAMS,SYSVOL=VMRELn,SYSPGNM=(368-495),         X
               SYSSTRT=(002,29),SYSPGCT=128,SYSSIZE=448K,SYSCYL=,        X
               SYSHRSG=(23,24,25,26,27,28),VSYSRES=,VSYSADR=IGNORE
```

```
CMSDOS    NAMESYS SYSNAME=CMSDOS,SYSVOL=VMRELn,SYSHRSG=(31),             X
                SYSSTRT=(003,038),SYSPGCT=8,SYSSIZE=32K,SYSCYL=,         X
                SYSPGNM=(496-503),VSYSRES=,VSYSADR=IGNORE

INSTVSAM  NAMESYS SYSNAME=INSTVSAM,SYSVOL=VMRELn,SYSCYL=,                X
                SYSSTRT=(003,47),SYSPGCT=8,SYSSIZE=32K,SYSHRSG=(254),    X
                SYSPGNM=(4064-4071),VSYSRES=,VSYSADR=IGNORE
          END
```

The supplied DMKSNT modules each have six entries: entries for saving copies of CMS, CMSSEG, CMSVSAM, CMSAMS, CMSDOS, and INSTVSAM, if you use all the recommended labels and allocations and the starter system supplied DMKSYS when you follow the system generation procedure. (The INSTVSAM segment is a CMSDOS segment that is used only during the procedure for loading and saving CMSVSAM and CMSAMS segments. For an explanation of this procedure, see the section "Loading and Saving Discontiguous Saved Segments" in Part 3.)

For an illustration of the storage layout resulting from this sample configuration of discontiguous saved segments, see Figure 31.

The supplied DMKSNT assumes a DOS/VS Release 34 starter system is being used. If you are using a DOS/VS Release 33 starter system, or earlier, to generate CMSVSAM, and/or CMSAMS, you must change the DMKSNT file. The CMSVSAM segment generated with these starter systems requires five shared segments and one nonshared segment. The CMSAMS segment generated with these starter systems requires 6 shared segments and 2 nonshared segments.

If you wish to change or add to the system name table that is supplied, you must code your own macro and create a DMKSNT file of your own. Note that one entry can be created for each type of discontiguous segment. For example, in addition to a CMSSEG entry, you could code an alternate entry, CMSSEG1, for testing purposes.

The system generation procedure tells you when to assemble your own file. If the supplied DMKSNT module meets your installation's needs, you need not code or assemble the DMKSNT module.

If you do create your own version of the system name table, your file must have a CSECT and END statement:

```
DMKSNTBL 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 named system you create)
         END
```

Note that the loader automatically inserts a PUNCH SPB (Set Page Boundary) card, to force this module to a 4K boundary when the CP system is built. Also, DMKSNT is a pageable module which should not exceed the size 4K. DMKSNT should be made resident if its size is greater than 4K.

Information about coding the NAMESYS and NAMENCP macros follows.

Coding the NAMESYS Macro

The NAMESYS macro describes the name and location of the saved system or discontiguous saved segment. Shared segments may be specified, but they must consist of reenterable code, with no alteration of its storage space permitted.

The format of the NAMESYS macro is:

```
| label | NAMESYS | SYSSIZE=nnnK,SYSNAME=name,[VSYSRES=cccccc,]        |
|       |         | VSYSADR=[cuu   ],SYSVOL=cccccc,[SYSCYL=nnn,]       |
|       |         |         [IGNORE]                                   |
|       |         | SYSSTRT=(cc,p),[SYSPGCT=pppp,]                     |
|       |         | SYSPGNM=(nn,nn,nn-nn,...),                         |
|       |         | SYSHRSG=(s,s,...),                                 |
|       |         | PROTECT={OFF}                                      |
|       |         |         {ON }                                      |
```

<u>where:</u>

label is any desired user label.

SYSSIZE=nnnK
: is up to 3 decimal digits representing the minimum amount of storage you must have available in order to IPL the saved system. K must be specified. Although you must code this operand for discontiguous saved segments, it is not used for them.

SYSNAME=name
: is the name (up to 8 alphameric characters) given to the system or segment to be used for identification by the SAVESYS and/or IPL commands. The name selected must not be one that could be interpreted as a hexadecimal device address (for example, A or E).

VSYSRES=cccccc
: is the real volume serial number (up to 6 alphameric characters) of the DASD volume containing the minidisk that is the system residence volume of the system to be saved. This operand is ignored if VSYSADR=IGNORE, but you must specify it as null (VSYSRES=,).

VSYSADR=cuu
: is the virtual address of the minidisk that is the system residence volume of the system to be saved.

VSYSADR=IGNORE
: indicates that the NAMESYS macro is describing a system or segment that does not require a virtual system residence volume. Code VSYSADR=IGNORE when you are defining a discontiguous saved segment.

SYSVOL=cccccc
: is the volume serial number (up to 6 alphameric characters) of the DASD volume designated to receive the saved system or segment. This must be a CP-owned volume.

SYSCYL=nnn
is the real starting cylinder of the minidisk (specified by VSYSRES and VSYSADR) that is the system residence volume of the system to be saved. This operand is ignored if VSYSADR=IGNORE, but you must specify it as null (SYSCYL=,).

SYSSTRT=(cc,p)
designates the starting cylinder (cc) and page address (p) on SYSVOL at which this named system is to be saved. During the SAVESYS and IPL command processing, this is used to generate the "cylinder page and device" address for the DASD operations. These numbers are specified in decimal.

The number of pages written to this area is the total number specified via the SYSPGNM operand, plus one information page.

SYSPGCT=pppp
is the total number of pages (pppp) to be saved (that is, the total number of pages you indicate via the SYSPGNM operand). This is a decimal number, up to four digits. The SYSPGCT operand is optional; if you do not specify it, the NAMESYS macro will calculate the number of pages to be saved.

SYSPGNM=(nn,nn,nn-nn,...)
are the numbers of the pages to be saved. Pages may be specified singly or in groups. For example: if pages 0, 4, and 10 through 13 are to be saved, use the format: SYSPGNM=(0,4,10-13). The total must be equal to the SVSPGCT specification.

SYSHRSG=(s,s,...)
are the segment numbers designated as shared (numbered from zero up, with the first segment, for example, specified as 0). The pages in these segments are set up at IPL time to be used by any user loading by this name. All segments to be shared must be reenterable. The maximum number of shared segments that can be defined is 78.

PROTECT= {OFF | ON}
indicates that VM/370 is to run either with protected (ON) or unprotected (OFF) shared segments for the particular 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/370 control program; the change will be seen by all users of the shared page.

The number of 4K pages available per DASD cylinder is:

Pages/Cylinder	DASD Type
24	3340-35, 3340-70, 2305
32	2314, 2319
57	3330, 3333
120	3350 (in native mode)

Information on the following subjects is in the *VM/370 System Programmer's Guide:*

- Determining when to save a system
- Using the SAVESYS command
- Saving the CMS system
- Saved system restrictions for CMS
- Saving OS
- Using discontiguous saved segments (CMSDOS, CMSSEG, CMSVSAM, CMSAMS)

Coding the NAMENCP Macro

You must create an entry in the system name table (DMKSNT) for each unique 3704/3705 control program that you generate. If you can foresee generating several versions of the 3704/3705 control program, define extra entries in the system name table when you generate VM/370. In this way, you do not have to regenerate the VM/370 system just to update the system name table.

Use the NAMENCP macro to define 3704/3705 program entries in the system name table. The format of the NAMENCP macro is:

```
| label | NAMENCP | CPSIZE=nnnK,
|       |         | CPNAME=ncpname,
|       |         | CPTYPE={EP }
|       |         | SYSPGCT=pp,
|       |         | SYSVOL=volser,
|       |         | SYSSTRT=(ccc,p)
```

where:

label — is any desired user label.

CPSIZE=nnnK — is the storage size of the 3704/3705 specified during the 3704/3705 control program generation. A maximum of 256K can be specified.

CPNAME=ncpname — is the name of the 3704/3705 control program image. This name is used in the SAVENCP and NETWORK LOAD commands. The name must be from one to eight alphameric characters.

CPTYPE={EP } — is the 3704/3705 control program type.

SYSPGCT=pp — is the total number of pages (pp) to be saved. This decimal value may be equal to the number of pages implied by the CPSIZE operand plus four pages for control information, but it must not exceed that total.

SYSVOL=volser — is the volume serial number (volser) of the DASD volume designated to receive the control program image. That volume must be a CP-owned volume.

SYSSTRT=(ccc,p) — is the starting cylinder (ccc) and page address (p) on SYSVOL at which this image is to be saved. These numbers must be specified in decimal.

Coding the NAME3800 Macro

The NAME3800 macro describes the name and location of the named system that will contain the 3800 character arrangement tables, graphic modifications, FCBs, and copy modifications for the 3800 printers. Multiple named systems may be specified. The 3800's RDEVBLOK contains a pointer to the named system currently in use for that particular 3800.

The format of the NAME3800 macro is:

label	NAME3800	CPNAME=libname, SYSPGCT=pp, SYSVOL=volser, SYSSTRT=(ccc,p)

where:

label is any desired user label.

CPNAME=libname is the name of the 3800 image library. This name is used in the IMAGELIB command. The name must be from one to eight alphameric characters.

SYSPGCT=pp is the total number of pages (pp) to be saved for the image library. This value is a decimal number up to two digits. To determine the number of pages to be saved, use the following steps:

1. The image library contains several core image members. Find the size of each core image member that was created by GENIMAGE; bytes seven and eight of the core image contain the member's size in bytes. Add eight bytes to each member's size.
2. Sum the sizes and add 16 bytes to the total.
3. Divide the total by 4096 bytes to achieve the page count (pp). Be sure to round up to the next whole page.

SYSVOL=volser is the volume serial number (volser) of the DASD volume designated to receive the 3800 image library. The volume must be a CP-owned volume.

SYSSTRT=(ccc,p) is the starting cylinder (ccc) and page address (p) on SYSVOL at which this image library is to be saved. These numbers must be specified in decimal.

Altering the Forms Control Buffer Load (DMKFCB)

The DMKFCB module is supplied with the starter system. This module defines a 3211 forms control buffer image with 6 lines per inch, 66 lines per page and the following channel skip 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

If you wish to alter the supplied buffer load, see the *VM/370 System Programmer's Guide* for directions.

Part 3. Generating VM/370 (CP, CMS, RSCS, and IPCS)

Part 3 describes the step-by-step generation procedures for CP, CMS, RSCS, and IPCS; the Installation Verification Procedure (IVP) for CP and CMS; and the procedures for loading and saving discontiguous saved segments. It contains the following sections:

- Introduction
- Generating CP and CMS Using the Starter Systems
- Verifying CP and CMS Using the IVP
- Loading and Saving Discontiguous Saved Segments
- Generating and Installing RSCS
- Generating and Installing IPCS

Introduction

Before you start to install VM/370, be sure you have the following available:

- Two real disk drives
- At least one real tape drive (the system generation process is simpler if you use two tape drives)
- Two scratch disks (one is used for the starter system and the other is used for the new system residence volume)
- The starter system tape
- The system Program Update Tape (PUT)
- At least one scratch tape

The system generation procedures described in this manual refer to related publications. These publications are listed in the Preface.

The following procedures for installing CP and CMS are described:

- 2314 starter system
- 3330 starter system
- 3340 starter system
- 3350 starter system

From these starter systems you can generate a VM/370 system for residence on a 2305, 2314, 3330, 3340, or 3350. (It is recommended, however, that 2305 devices be used for paging rather than for system residence; therefore the 2305 is not included in the system generation procedures.)

An MSS 3330V volume may not be used for system residence.

You can then load and save the CMS discontiguous saved segments your installation may wish to use.

Following the procedures for installing CP and CMS are the procedures for installing the optional RSCS (Remote Spooling Communications Subsystem) component of the VM/370 SCP.

In addition, the information you need to install the 3704/3705 control program is found in "Part 4. Generating the 3704/3705 Control Program."

General Information

CP and CMS have separate system residence disks which may be located on the same or different physical disks. The following procedure tells you how to generate the CP system residence disk or move the CMS system residence disk. Before you attempt to generate a VM/370 system, make sure that the real I/O configuration file (DMKRIO), CP system control file (DMKSYS), the VM/370 Directory file (DMKDIR), and, optionally, the forms control buffer load (DMKFCB) and the system name table file (DMKSNT) are punched. Information about preparing these files is in "Part 2: Defining Your VM/370 System." If CMS is to be saved as a named system, be sure that the NAMESYS macro is coded correctly in the DMKSNT file.

The VM/370 starter system is distributed on a 9-track tape (1600 or 6250 bpi), that can be restored to direct access volumes. You must specify the device type (2314/2319, 3330, 3340, or 3350) when you order VM/370. The 3340 starter system fits on a 35 megabyte disk, so it can be restored to any model of the 3340 or 3344. After the starter system has been restored to the particular type of device (2314, 3330, 3340, or 3350) it was ordered for, you can use it to generate a VM/370 system for residence on any other type of device as well as for the type of device for which it was ordered. You should also specify the tape density required (1600 or 6250 bpi).

The layout of the starter system's minidisk areas are constrained by the number of cylinders that may be dumped onto one volume of 1600 bpi tape. Therefore, it is strongly recommended that MAINT's 190 (the CMS system disk) and 194 (the CP area) be reproduced on larger minidisks for ease of maintenance. Also, "service staging areas" for CP/RSCS and CMS/IPCS (294 and 193 respectively), must be created to receive the auxiliary files and "update" files from the system PUT. This process is detailed later in this publication under the heading "Updating VM/370" and in the Memo-to-Users that is contained on the system PUT.

The VM/370 system tapes are as follows:

- The VM/370 starter system contains the base level of both the CP and CMS systems, the text decks with which to build these systems, and the maclib and support procedures.
- The SOURCE tape contains all source files, and macros of VM/370.
- The system Program Update Tape (PUT) contains all source updates, text decks, modules, macros and macro libraries, and procedures required to build the latest level of CP, CMS, RSCS, and IPCS.
- The SOURCE tape and the system PUT are created (and restored) with the VMFPLC2 command.

Five optional sets of tapes can also be ordered:

- The assembler tape containing source, macros, text, modules, and procedures for the assembler
- CP (UP) assembly listings (three tapes)
- CP (AP) assembly listings (two tapes)
- CMS assembly listings (two tapes)
- IPCS and RSCS assembly listings (one tape)

Generating CP and CMS Using the Starter Systems

Except where otherwise noted, you can substitute other values in place of the device addresses, volume labels, and allocations shown. Note that if you use the sample DMKSNT and DMKSYS files provided with the starter system, and the sample allocations shown in Steps 2 and 3, you can save your CMS system at the end of the procedure.

It is strongly recommended that you use the sample allocations given in Step 2 and the label VMRELn for the new system residence volume, to ensure that you have sufficient TEMP space to complete the system generation. (The TEMP space provided on the starter system volume may not be sufficient for large systems.)

The examples of messages and responses assume that you are performing the system generation at a typewriter terminal, such as a 3210, 2741, or 3767 (operating as a 2741). If you are using a display device, such as the 3277, when you type the response to a prompting message, that response appears in the user input area. When you enter that response, it is redisplayed in the output area on the line below the prompting message. Also, if the standalone service programs (such as the DASD Dump Restore program or Format/Allocate program) send output to a terminal display screen, the output is wrapped around immediately, when the screen becomes full, to continue displaying.

While you are generating the system, you may see some extraneous messages as the starter system is processing. These are not shown in the examples below. Only those messages that you should take note of, or respond to, are shown.

Step 1. Load the Format Program from the Starter System Tape

Mount the CP starter system tape and IPL the tape. The CP Format/Allocate service program is the first file on the tape; it is now loaded. Do not rewind the tape because the next file is needed later in the system generation procedure (Step 4).

Step 2. Format, Label, and Allocate the System Residence Volume

Use the CP Format/Allocate program to format, label, and allocate space on the new system residence volume. This label must be VMRELn; where n is the release level of the VM/370 System control program. VMRELn is used in the starter system's system control file - SYSOWN marco - to allow the volume to be used for paging, spooling, and TDSK allocations. First, identify the system console by pressing the Request key (or equivalent); if the console address is either 009 or 01F, you do not have to press the Request key. Then, to execute the Format/Allocate service program, respond to the prompting messages.

In the following example, the responses (format, 131, device type, 000, end cylinder, and VMRELn) format the real disk at address 131 and label it VMRELn. The label you specify must match what you specified in the SYSVOL operand of the SYSRES macro statement when you defined the system in your DMKSYS module. In any case, do not use CPRnL0 because that is the label of the starter system disk. The console output looks like:

```
VM/370 FORMAT/ALLOCATE PROGRAM RELEASE n
ENTER FORMAT OR ALLOCATE:format
FORMAT FUNCTION SELECTED
ENTER DEVICE ADDRESS (CCU):131
ENTER DEVICE TYPE:device type¹
ENTER START CYLINDER (XXX) OR "LABEL":000
ENTER END CYLINDER (XXX):end cylinder¹
ENTER DEVICE LABEL:VMRELn²
FORMAT STARTED
FORMAT DONE
000 NO. PAGE RECORDS WITH READ-CHECK ERRORS
```

When the format operation completes, the prompting message

```
ENTER FORMAT OR ALLOCATE:
```

is displayed. Now that the system residence volume is formatted and labeled, you must allocate the disk space. Again, you must respond to the prompting messages. In the following example, the space on the various device types at address 131, with the label VMRELn, are indicated. You can use the formulas given in the "Creating Your VM/370 Directory" section of Part 2 to ensure enough space is allocated for your VM/370 directory. If you do not allocate your DASD space as shown in this example, you are responsible for ensuring that you have enough TDSK space to perform the assemblies associated with VM/370 system generation.

```
ENTER FORMAT OR ALLOCATE:allocate
ALLOCATE FUNCTION SELECTED
ENTER DEVICE ADDRESS(CCU):131
ENTER DEVICE TYPE:device type
ENTER DEVICE LABEL:VMRELn
ENTER ALLOCATION DATA FOR VOLUME VMRELn
TYPE CYL CYL
.... ... ...
```

	2314		3330		3340		3350	
perm	000	019	000	012	000	023	000	008
drct	020	023	013	016	024	027	009	012
temp	024	100	017	201	028	173	013	276
perm	101	102	202	202	174	176	277	277
temp	103	180	203	389	177	310	278	399
tdsk	181	202	390	402	311	346	400	554
perm[3]			403	807	347	697		
end								

----

[1]The specifiable device types and their respective "end cylinders" are: 2314 is 202, 2319 is 202, 3330 is 403, 3330-11 is 807, 3340-35 is 347, 3340-70 is 697, 3350 is 554, 2305-1 is 47 and 2305-2 is 95.

| [2]VMRELn must be VMREL4, VMREL5, or VMREL6, depending on the release level of the VM/370 SCP.

[3]This line gives the required specifications for the 3330 Model 11 and the 3340 Model 70.

```
ALLOCATION RESULTS

        2314           3330           3340           3350
PERM 000  019       000  012       000  023       000  008
DRCT 020  023       013  016       024  027       009  012
TEMP 024  100       017  201       028  173       013  276
PERM 101  102       202  202       174  176       277  277
TEMP 103  180       203  389       177  310       278  399
TDSK 181  202       390  402       311  346       400  554
PERM                403  807       347  697

DEVICE 131 VOLUME VMRELn ALLOCATION ENDED
ENTER FORMAT OR ALLOCATE:
```

Step 3. Label the Starter System Volume

Use the Format/Allocate program to label the scratch volume that is to contain the CP starter system. This label must be CPRnL0. You can format and label this volume, or just label it. Formatting is unnecessary (unless the pack has never been initialized before) because you are going to restore the starter system to this volume. If you get an I/O error trying to label the pack, format only cylinder zero and then try to label the pack again. In the following example, the responses (format, 130, device type, label, and CPRnL0) to the prompting messages put the label CPRnL0 on the real disk at address 130.

```
ENTER FORMAT OR ALLOCATE: format
FORMAT FUNCTION SELECTED
ENTER DEVICE ADDRESS (CCU):130
ENTER DEVICE TYPE:device type
ENTER START CYLINDER (XXX) OR "LABEL":label
ENTER DEVICE LABEL: CPRnL0
LABEL IS NOW CPRnL0
```

When the Format/Allocate program is complete, it responds:

```
ENTER FORMAT OR ALLOCATE:
```

You need not respond to this message. Press the PA1 key twice to return to CP mode.

Now the starter system volume is available and ready for the data that is to be placed on it by the DASD Dump Restore service program (module DMKDDR). The DASD Dump Restore program is the second file on the starter system tape.

Step 4. Load the DASD Dump Restore Program from the Starter System Tape

IPL the starter system tape a second time to load the DASD Dump Restore (DDR) program. It is the second file on the starter system tape. Do not rewind the tape, because the next file is needed in Step 5.

Step 5. Restore the Starter System to Disk

Respond to the DDR prompting messages to restore the starter system.

In the following example, the starter system is restored from the 2400 series tape drive at address 280 to the real disk at address 130 (and with label CPRnL0). The console output is:

```
VM/370 DASD DUMP/RESTORE PROGRAM RELEASE n
ENTER CARD READER ADDRESS OR CONTROL STATEMENTS
ENTER: sysprint cuu (cuu=real printer address)
ENTER: input 280 2400
ENTER: output 130 device type CPRnL0
ENTER: restore all
RESTORING CPRnL0
END OF RESTORE
ENTER: (null line -- END key on 3215 or Enter key on 3277)
END OF JOB
```

The DDR program restores the third file on the starter system tape to the disk labeled CPRnL0. The restored disk contains:

- A 191 minidisk for the userid CPGEN, which contains a sample VM/370 directory (RELEASEn DIRECT), as well as sample source for DMKSYS, DMKSNT, and DMKFCB
- A VM/370 starter system nucleus
- A complete CMS system residence volume
- A complete CP system containing macro libraries and text files

When the disk is restored, continue with the system generation procedure. The format of the restored disk is shown in Figures 22, 23, 24, and 25.

Real Cylinder	Number of Cylinders	Contents
0	1	VM/370 directory
1	1	191 minidisk for the IVPM1 user
2	1	191 minidisk for the IVPM2 user
3-6	4	CP nucleus
7	1	Warm start data
8-9	2	I/O Error Recording area
10	1	Spool file checkpoint
11-33	22	Spooling and paging space
34	1	191 minidisk for the CPGEN user
35-169	135	190 minidisk (the CMS system disk) for the CMSSYS user Note: The nucleus occupies the last two cylinders of the minidisk.
170-202	33	194 minidisk of the CPGEN user - it contains the CP object modules (text decks)

Figure 22. Format of a 2314 Restored Disk

Real Cylinder	Number of Cylinders	Contents
0	1	VM/370 directory
1	1	191 minidisk for the IVPM1 user
2	1	191 minidisk for the IVPM2 user
3-5	3	CP nucleus
6	1	Warm start data
7-8	2	I/O Error Recording area
9	1	Spool file checkpoint
10-28	19	Spooling and paging space
29	1	191 minidisk for the CPGEN user
30-114	85	190 minidisk (the CMS system disk) for the CMSSYS user Note: The nucleus occupies the last cylinder of the minidisk.
115-141	27	194 minidisk of the CPGEN user - it contains the CP object modules (text decks)
142-403	262	Not used.

Figure 23. Format of a 3330 Restored Disk

Real Cylinder	Number of Cylinders	Contents
0	1	VM/370 directory
1-2	2	191 minidisk for the IVPM1 user
3-4	2	191 minidisk for the IVPM2 user
5-10	6	CP nucleus
11	1	Warm start data
12-13	2	I/O Error Recording area
14	1	Spool file recovery
15-45	31	Spooling and paging space
46-47	2	191 minidisk for the CPGEN user
48-287	240	190 minidisk (the CMS system disk) for the CMSSYS user Note: The nucleus occupies the last two cylinders of the minidisk.
288-347	60	194 minidisk of the CPGEN user - it contains the CP object modules (text decks)
Note: Cylinders 348-695 are not used when the starter system is restored to a 3340 Model 70.		

Figure 24. Format of a 3340 Restored Disk

Real Cylinder	Number of Cylinders	Contents
0	1	VM/370 directory
1	1	191 minidisk for the IVPM1 user
2	1	191 minidisk for the IVPM2 user
3-4	2	CP nucleus
5	1	Warm start data
6-7	2	I/O Error Recording area
8	1	Spool file checkpoint
9-19	11	Spooling and paging space
20	1	191 minidisk for the CPGEN user
21-55	35	190 minidisk (the CMS system disk) for the CMSSYS user Note: The nucleus occupies the last cylinder of the minidisk.
56-64	9	194 minidisk of the CPGEN user - it contains the CP object modules (text decks)
65-554	490	Not used.

Figure 25. Format of a 3350 Restored Disk

CPGEN, CMSSYS, IVPM1, and IVPM2 are user identifications that own certain minidisks on the starter system volume.

CPGEN is the userid of the operator (that is, you use this userid to control the real system and to build a version of CP tailored to your installation).

CMSSYS is a directory entry on the starter system volume that owns CMSSYS 190 (the CMS system disk). CPGEN has a read-only link to CMSSYS 190 in order to use it to create the new system. Using this link, the CPGEN user can read from, but not write on, the 190 minidisk belonging to the CMSSYS user.

IVPM1 and IVPM2 are used with the Installation Verification Procedure to test the new system.

Step 6. IPL the Starter System

Load (IPL) the starter system from the disk you restored it to. In our example the address is 130. At this point, only the device containing the system residence volume, 130, is known to the starter system.

Remember, if you have control units that share more than 16 devices and are also switchable to another processor, the channel interface enable switch from the other processor should be in the disable position while you perform the system generation.

Step 7. Define the Devices Needed To Do the System Generation

If your system console is at an address other than 009 or 01F, after you load the starter system you must press the Request key (or equivalent key) to enable the starter system to recognize the system console. If the console is not recognized, the VM/370 starter system enters a disabled wait state with code X'27' in the PSW.

Note: Either an unrecoverable I/O error occurred or the system input was incorrect. Determine the cause of the problem and correct it; then reload the starter system.

At this point both the system residence volume and system console are recognized by the starter system and you can define the other devices you need. The starter system supports up to 16 channels, 8 control units, and 16 devices. The real control blocks for these devices are not built in the standard manner; the starter system builds them dynamically.

The starter system program (DMKSSP) then prompts you to answer the following questions until all the real control blocks necessary to operate a minimum machine configuration are created. The following example assumes: a 1403 printer at address 00E, a 2540 card reader/punch at addresses 00C (reader) and 00D (punch), tape drives at addresses 280 and 281, and a DASD device appropriate for the new system residence volume at address 131. The messages you receive at this time are:

```
VM/370 STARTER SYSTEM RELEASE n
ENTER PRINTER ADDRESS (CUU):00e
ENTER DEVICE TYPE (1403,1443,3203,3211,3800):1403
ENTER PUNCH ADDRESS (CUU):00d
ENTER DEVICE TYPE(2540P,3525):2540p
ENTER READER ADDRESS (CUU):00c
ENTER DEVICE TYPE (2501,2540R,3505):2540r
ENTER ADDRESS WHERE PID TAPE IS MOUNTED (CUU):280
ENTER DEVICE TYPE (2401,2415,2420,3420):2401
ENTER ADDRESS WHERE SCRATCH TAPE IS MOUNTED (CUU):281
ENTER DEVICE TYPE (2401,2415,2420,3420):2401
ENTER DEVICE ADDRESS WHERE SYSTEM RESIDENCE WILL BE BUILT (CUU):131
ENTER DEVICE TYPE (2319,2314,3330,3340,3350,2305):device type

***SYSTEM DEFINITION COMPLETED***

00E PRINTER
00D PUNCH
00C READER
280 PID TAPE
281 SCRATCH TAPE
131 NEW SYSTEM RESIDENCE
ARE THE ABOVE ENTRIES CORRECT (YES,NO):yes
```

```
VM/370 VERSION vv LEVEL 00 PLC 0000; mm/dd/yy hh:mm:ss
NOW 08:54:23 EDT FRIDAY mm/dd/yy
CHANGE TOD CLOCK (YES|NO):yes
SET DATE MM/DD/YY :mm/dd/yy
SET TIME HH:MM:SS :09:04:36
PRESS "TOD ENABLE SET" KEY AT DESIGNATED INSTANT
```

The TOD clock referred to is the System/370 Time of Day clock. Enter the actual date and time in response to the "SET DATE" and "SET TIME" messages, and press the TOD Enable Set switch on the system control panel when the exact time specified agrees with the installation wall clock.

```
NOW 09:04:36 EDT FRIDAY mm/dd/yy
CHANGE TOD CLOCK (YES|NO):no
09:04:38 START ((COLD|WARM|CKPT|FORCE) (DRAIN))(SHUTDOWN):cold
```

Some DMKLNK117E messages appear at this time. They can be ignored.

```
09:04:42 AUTO LOGON *** CPGEN USERS=001 BY SYSTEM
09:04:42
```

The following message appears only if your system storage size is different from that specified in the SYSCOR macro in the DMKSYS module supplied with the starter system:

```
DMKCPI952I nnnnK SYSTEM STORAGE
```

The following informational message provides a storage allocation map of CP:

```
DMKCPI957I STOR sssssK, NUC nnnK, DYN dddddK, TRA tttK, FREE ffffK,
     V=R vvvvvK
```

If you have not defined your system residence volume with a label of VMRELn messages are issued indicating that the volume labeled VMRELn is not mounted. If you have labeled it as VMRELn, some messages indicating VMRELn conflicts are issued. These are caused by the MDISK statements for the various supported system residence devices in the starter system's directory. You can ignore them and the following message.

```
09:04:43 FILES:  NO RDR,  NO PRT,  NO PUN
```

Step 8. Set the Terminal Mode and Spool the Console

If the system console you are using is a display device, you should at this point spool your console output so that you have a record of what you do. To spool the console input and output, issue the command:

```
spool console start
```

to save a copy of the system generation.

Because the default terminal environment for the primary system operator is CP, you should also issue the command:

```
terminal mode vm
```

The virtual machine terminal mode lets you remain in the CMS environment when you enter data on the display device.

Step 9. Define or Attach the System Residence Device

The CPGEN virtual machine assumes that the system residence volume is labeled VMRELn, and that it is a device type like that of the starter system volume and resides at virtual address 350. Device types unlike that of the starter system reside at virtual address 351, 352, and 353. If you labeled your system residence volume VMRELn in Step 2, your system residence device is already available; now you, as the operator of CPGEN, must define it. Use Procedure 1 to define it.

To see which virtual device was defined as your system residence volume, issue:

```
query virtual dasd
```

Note: If you are using a 3330 device for your new system residence volume, it will appear to CPGEN to have 808 cylinders. Users of 3330 Model 1 devices can ignore this. Users of 3330 Model 11 devices should be aware that this permits them to use any part of the volume for system residence.

If you did not label your system residence device VMRELn, you must now attach it to your virtual machine, CPGEN. Use Procedure 2 to attach your system residence device.

For example, if you used the values shown in Step 7, you designated the 131 drive, labeled VMRELn, as your system residence device. You must define your virtual device 350 so that it corresponds to the real disk 131.

Use one of the following procedures to define or attach your system residence volume:

- Procedure 1

  If you are creating a system residence with label VMRELn, see the following table to determine which virtual device must be defined as 131.

Enter the following command:

```
define nnn as 131
```

Where nnn is 350, 351, 352, or 353.

Starter System Device Type	System Residence Device Type			
	2314	3330	3340	3350
2314	350	351	352	353
3330	351	350	352	353
3340	351	352	350	353
3350	351	352	353	350

- Procedure 2

  If you did not label your system residence volume as VMRELn, you must attach the volume to your virtual machine now. Enter:

  ```
  attach 131 to cpgen as 131
  ```

  Note: The first device address you specify in the ATTACH command is for the real device; the second device address is for the virtual device. The real 131 is the system residence volume you formatted in Step 2. The virtual 131 is the system residence device you defined in DMKSYS (using the SYSRES macro instruction). The system residence device was also entered in response to the message:

  ```
  ENTER DEVICE ADDRESS WHERE SYSTEM RESIDENCE WILL BE BUILT (CUU):
  ```

Step 10. Make Other Devices Available

You must attach your tape drives and designate the printer to receive abnormal termination dumps if they occur. Use the explanation that follows to decide how to do this. Press the Request key before entering each command, for example:

```
attach 280 to cpgen as 181
attach 281 to cpgen as 182
set dump 00e
```

In the first command, the real tape drive (280) attached must be the same real address you entered in Step 7 in response to the message:

```
ENTER ADDRESS WHERE STARTER SYSTEM TAPE IS MOUNTED (CUU):
```

This real device must be attached to CPGEN as 181, because the VMSERV EXEC procedure (which is used later) expects it to be 181. Mount your system PUT on this tape drive for Step 13.

In the second command, the real tape drive (281) attached must be the same real address you entered in Step 7 in response to the message:

```
ENTER ADDRESS WHERE SCRATCH TAPE IS MOUNTED (CUU):
```

This real device must be attached to CPGEN as 182, because the GENERATE EXEC procedure (which is used later) expects the scratch tape to be mounted on 182.

If only one tape drive is available, enter

```
attach 280 to cpgen as 181
```

You can define your virtual 181 as 182 later in the system generation procedure.

The address of the real printer, defined in Step 7, is 00E. Any system dumps that occur are directed to that address.

If you wish, you can issue the CP command:

```
query virtual all
```

before and after performing Step 10, to see how your virtual machine's configuration changes.

Step 11. Load CMS

Load CMS from virtual address 190 by issuing the CP command:

```
ipl 190 parm seg=null
```

You are loading a CMS nucleus without a shared segment and can expect an error message stating that the segment specified is invalid. After CMS is loaded, a message is displayed on the system console indicating that CMS has successfully loaded:

```
CMS VERSION n.n mm/dd/yy hh.mm
```

Press the ENTER key (END key on a 3215) so that the CPGEN 191 minidisk is accessed as your A-disk.

Step 12. Define and Format A Temporary Minidisk

In Step 2 you allocated TDSK space on your system residence volume. In order to be able to assemble files (DMKRIO, DMKSYS, DMKSNT, DMKFCB) later, you have to define a temporary minidisk. Issue the command:

```
define txxxx 192 yy
```

where: xxxx is the device type of your system residence volume (2314, 3330, 3340, or 3350)

yy is: 20 if xxxx is 2314
11 if xxxx is 3330
25 if xxxx is 3340
5 if xxxx is 3350

A message displayed on the console, indicates the 192 minidisk is defined:

```
DEVICE 192 DEFINED
```

Before any new minidisk area can be used for CMS files, it must be initialized with the CMS FORMAT command, which formats the area into fixed-sized blocks. Issue the command:

```
format 192 d
```

The CMS FORMAT command prompts you with the following message:

```
FORMAT WILL ERASE ALL FILES ON DISK 'D(192)'.
DO YOU WISH TO CONTINUE? (YES/NO):
```

Respond "yes", CMS prompts you with:

```
ENTER DISK LABEL:
tmp192
```

Enter a one-to-six character alphameric label for the minidisk. CMS then issues:

```
FORMATTING DISK D

'nn' {CYLINDERS} FORMATTED ON 'D(192)'
```

Obtain write access to your CMS system disk:

```
link cmssys 190 190 w write
```

Step 13. Apply the CP (Control Program) Service

The system PUT that was mounted in Step 10 is a system update service that can include new functions as well as cumulative system changes for VM/370. The latest system PUT contains all new updates, as well as all previous updates since the last VM/370 base release. PID automatically ships the system PUT to the user location, and the user is responsible for applying the updates to VM/370 systems. The system PUT supplied with the starter system should be installed as part of the system generation procedure.

Use the 192 TDSK created in Step 12 in place of the CPGEN 191 disk for the remainder of this procedure (up to the point of loading your new CP system via IPL).

Issue the following CMS commands:

```
copy * * a = = d
vmfplc2 load * * d
```

The system issues the following messages:

```
LOADING.....
5749010 06pp38     D1¹
5749010   EXEC     D1
VMSERV    EXEC     D1
VMFPLC2 MODULE     D1
END-OF-FILE OR END-OF-TAPE
R;
```

Issue the following CMS commands:

```
access 192 c
release a
vmserv
```

The VMSERV EXEC maps the system PUT and requests permission to print the "Memo to Users". Reply 'YES'. VMSERV will then print the memos and remind you to read the memos prior to installing service. VMSERV exits after the memos are printed.

After reviewing the "Memo to Users" contained on the system PUT, and contacting IBM concerning the latest service activity, you can begin the installation of service to CP by issuing the CMS command:

```
vmserv nomemo noipl
```

You will be prompted for various information concerning your system (including staging area addresses and the products for which you wish to apply service). You should apply service for the SCP (5749-010). The address of your CP base staging area is 194. Reply "yes" to the question:

```
IS THIS THE INITIAL SYSGEN OF THIS SYSTEM? (YES|NO)
```

CMS is automatically loaded via ipl when the VMSERV EXEC completes.

The 192 TDSK should be accessed as the A disk with the command:

```
access 192 a
```

Step 14. Prepare the Service Programs

Use the GENERATE EXEC procedure to punch the service programs. Issue the command:

```
generate srvcpgm
```

These service programs are needed for standalone use; you should externally identify the decks and keep them intact.

The programs punched are:

- DMKFMT (a 3-card loader precedes the DMKFMT text deck)
- DMKDIR (a 3-card loader precedes the DMKDIR text deck)
- DMKDDR (a 3-card loader precedes the DMKDDR text deck)
- IBCDASDI (the IBCDASDI text deck is loadable)

----
[1]Where pp is the PUT number.

The GENERATE EXEC issues the following message:

```
THE FOLLOWING STANDALONE SERVICE PROGRAMS ARE BEING PUNCHED
** FORMAT - DIRECT - DUMP/RESTORE - IBCDASDI **

PUNCHING ' IPL FMT ' ******
PUNCHING ' IPL DIR ' ******
PUNCHING ' IPL DDR ' ******
PUNCHING ' IPL IBCDASDI ' ******
```

Each program deck is preceded by a CP userid card and several separator cards, all of which may be discarded. The format of these cards is described in the _VM/370 Operator's Guide_. For more information about the GENERATE EXEC procedure, see "Part 5. Updating VM/370."

Step 15. Print or Punch the Starter System Supplied Directory, DMKSNT, DMKSYS, and DMKFCB

After the service programs are punched, the GENERATE EXEC asks you whether you want a copy of the directory printed. Respond "yes."

```
PRINT COPY OF RELEASEn DIRECT? -- RESPOND (YES|NO): yes
```

This prints a copy of the directory, as well as copies of the DMKSYS ASSEMBLE (the CP system control file), DMKFCB ASSEMBLE (the forms control buffer file), and DMKSNT ASSEMBLE (the system name table) provided with the starter system.

The GENERATE EXEC procedure issues the following message:

```
A SAMPLE DIRECTORY IS BEING PRINTED TO AID YOU.
IT SHOWS WHERE THE VIRTUAL DISKS ARE LOCATED ON 'CPRnLO'
YOU MAY USE THESE MINIDISKS FOR OTHER VIRTUAL MACHINES,
IN PARTICULAR THE CMS SYSTEM DISK ( MAINT 190 ) AND
THE CP STAGING AREA DISK ( MAINT 194 )
INCLUDED IN THIS DIRECTORY IS THE USERID: MAINT
WHICH WILL BE USED FOR FUTURE SUPPORT OF THE SYSTEM.
THIS USERID SHOULD BE INCLUDED IN THE DIRECTORY
YOU BUILD FOR YOUR FLOOR USE.

** CAUTION ** IF YOU DESTROY USER MAINT'S AREAS, IT WILL BE
              NECESSARY TO RE-BUILD THE ENTIRE SYSTEM.

A SAMPLE OF DMKSYS, DMKFCB, AND DMKSNT ASSEMBLE ARE ALSO BEING
PRINTED TO AID YOU. THIS SAMPLE DMKSNT IS BASED ON THE
INFORMATION INCLUDED IN THE SAMPLE DMKSYS AS WELL AS THE
EXAMPLE ALLOCATIONS FOR VMRELn PROVIDED IN THE SYSGEN GUIDE.
A COPY OF THIS DMKSNT MODULE HAS BEEN INCLUDED IN THE CP NUCLEUS,
SUCH THAT IF ONE USES THE INCLUDED DMKSYS AND THE
SAMPLE ALLOCATION PROVIDED IN THE SYSTEM GENERATION GUIDE,
HE WILL BE ABLE TO SAVE HIS CMS SYSTEM UPON COMPLETION
OF THE SYSTEM GENERATION PROCEDURE. A COPY OF DMKFCB HAS BEEN
INCLUDED IN THE NUCLEUS AND NEED NOT BE RE-ASSEMBLED FOR
SYSTEM GENERATION. IT HAS BEEN INCLUDED FOR THE USER WHO WOULD LIKE
TO MODIFY OR ADD TO THE EXISTING BUFFER LOAD.

NOTE: IF THE USER WISHES TO MODIFY THE SAMPLE DMKSNT AND/OR DMKFCB
HE MAY INCLUDE THE UPDATED SOURCE WITH THE SOURCE INCLUDED UNDER
THE OPTION 'GENERATE VM370', OF THE SYSTEM GENERATION PROCEDURE.
IF PRESENT, IT WILL AUTOMATICALLY BE ASSEMBLED AND INCLUDED IN THE
NEW CP NUCLEUS.
```

Again, the GENERATE EXEC procedure prompts you:

```
DO YOU WISH TO HAVE A COPY OF DMKSNT, DMKSYS, DMKFCB, AND
RELEASEn DIRECT PUNCHED TO CARDS? -- RESPOND (YES|NO):
```

Enter "yes" to have these decks punched since you need to read in updated copies of these decks in Step 16. "Part 2. Defining Your VM/370 System" contains a listing of the directory and the DMKSNT, DMKSYS, and DMKFCB modules supplied with the starter system. The following messages are issued to indicate the files that are being punched:

```
PUNCHING ' DMKSNT ASSEMBLE ' ******
PUNCHING ' DMKSYS ASSEMBLE ' ******
PUNCHING ' DMKFCB ASSEMBLE ' ******
PUNCHING ' RELEASEn DIRECT ' ******
R;
```

Step 16. Build the VM/370 Directory and Assemble the Files Defining the Real I/O and System Devices

Create or update the following card files describing your installation's version of the VM/370 directory, real I/O configuration, and system devices and place them in the reader and invoke the GENERATE EXEC procedure to build the directory and assemble the files describing the configuration. Make sure your directory contains MDISK statements for MAINT's 193 and 294. Place the following cards in the card reader, in the sequence shown:

```
ID CPGEN
:READ  filename DIRECT
(Directory program control statements)
:READ  DMKRIO ASSEMBLE
(Real I/O configuration macros)
:READ  DMKSYS ASSEMBLE
(system control macro statements)
:READ  DMKSNT ASSEMBLE
(system name table macro statements)
```

The Directory program control statements, real I/O configuration macros, and system control macros are those you created according to the instructions in Part 2. "Defining Your VM/370 System." You can code your own system control macro statements or modify the sample files supplied with the starter system.

New requirements demand that the user define space in the directory for "Service Staging Areas". The minidisk addresses and suggested sizes are described in the Memo to Users for the SCP and later in this publication under "Updating VM/370". These areas are used for the AUX files and PTF files contained on the system PUT. The files are loaded by the individual service installation EXECs.

Since the starter system's directory does not allocate space for the minidisks, the "new user" service loaded in Step 13 did not attempt to load these files. They will be loaded, however, when the rest of the service is loaded, later in this procedure (Step 23).

Also, if you wish, you can change the printer forms control buffer module (DMKFCB) to conform to local requirements. If you want to change the printer forms control buffer, modify the file that was punched (in Step 15) and place it in the reader following the system name table macro statements:

:READ DMKFCB ASSEMBLE
(forms control macros)

:READ DMKSNT ASSEMBLE
(system name table macros)

Note: Ensure that the cylinders specified for the directory as well as the DMKSYS and DMKSNT source programs do not overlap each other.

Instructions for creating new forms control macro statements are in the *VM/370 System Programmer's Guide*.

FORMAT OF THE READ CONTROL STATEMENT

The READ control statements must be punched according to the format shown in Figure 26.

Column	Number of Characters	Contents	Meaning
1	1	colon':'	Identifies card as a control card
2-5	4	READ	Identifies card as a READ control card
6-7	2	blank	
8-15	8	fname	Filename of the file punched
16	1	blank	
17-24	8	ftype	Filetype of the file punched
25-80	56	blank	

Figure 26. Format of READ Control Statement

SPECIAL PROCEDURE IF YOU ARE USING ONLY ONE TAPE DRIVE

If you are using only one tape drive, issue the command:

define 181 as 182

Now mount the scratch tape in place of the system PUT and ready the device.

INVOKE THE GENERATE EXEC PROCEDURE

When all the files are placed in the reader, invoke the GENERATE EXEC procedure by issuing the following command:

    generate vm370

This procedure invokes the VM/370 directory program to build the disk-resident VM/370 directory, then assembles the DMKRIO and DMKSYS files that you placed in the real card reader. GENERATE prompts you for the filename of your directory file with the message

    ENTER DIRECTORY FILENAME

Enter the name you specified in the directory deck.

After the directory program execution completes, the DMKRIO and DMKSYS files are assembled in preparation for building the new CP system nucleus. GENERATE then checks for DMKFCB and DMKSNT source files. When new versions of the DMKFCB and DMKSNT modules are provided, GENERATE assembles the new modules and replaces the corresponding starter system supplied modules with the new modules. If any errors occur while the VM/370 directory is being built, the directory program issues error messages and the GENERATE EXEC procedure issues the following message:

    CORRECT THE DIRECTORY CARDS AND RELOAD THE CARD READER
      RESPOND WITH: GENERATE DIRECT

Correct the errors in the directory program control statements, and reload the card reader with only the ID card, the :READ statement, and the directory program control statements. Then respond with:

    generate direct

If errors are detected while the DMKRIO, DMKSYS, DMKFCB, or DMKSNT files are assembling, GENERATE issues a similar message:

    CORRECT THE filename ASSEMBLE FILE AND RELOAD THE CARD READER
      RESPOND WITH: GENERATE filename

Correct the errors in the indicated file, and reload the card reader with only the ID card, the :READ statement, and the appropriate file. Then, issue the GENERATE command with the appropriate option. For example:

    generate dmksys

Step 17. Attached Processor Support and Virtual=Real Machines

Once the directory is built and the files are assembled, the GENERATE EXEC asks:

    ARE YOU GENERATING AN AP SYSTEM?--(RESPOND YES|NO)

followed by:

    VIRTUAL=REAL OPTION REQUIRED (YES,NO):

Your responses to these questions will determine the CNTRL file and loadlist EXEC necessary to correctly build your system.

If you respond "yes" to this question, you are prompted to enter the amount of storage you wish to reserve in the CP nucleus for a virtual=real machine. (Part 1. "Planning for System Generation" contains formulas to help you determine how much storage you need to reserve.)

To generate a V=R system of size XM (megabytes), specify XM in the format nnnnk. nnnnk must be a multiple of 1024K. For example, to indicate 3M you should specify 3072K. The minimum size you can specify is 32K. The maximum specifiable size is 15360K.

A "yes" response displays the following messages:

```
STORAGE SIZE OF VIRT=REAL (MINIMUM IS 32K):
100k
0100K STORAGE SIZE FOR VIRTUAL=REAL
IS THE ABOVE ENTRY CORRECT (YES,NO):
yes
```

If you respond "no" to this message, the following message is displayed:

```
FILE 'DMKSLC TEXT A1' NOT FOUND
```

You must have enough virtual storage available to generate the CP nucleus with a Virtual=real area. If you do not have enough virtual storage available, redefine storage for your virtual machine.

Step 18. Load the CP Nucleus

Once you respond to the virtual=real generation questions, the GENERATE EXEC procedure builds and writes the CP nucleus on tape. To do this, GENERATE first issues the the VMFLOAD command to punch the loader and CP object modules to a virtual punch spool file. Then GENERATE transfers this file to the virtual card reader file and writes the file on tape. GENERATE issues the following messages during the processing:

```
hh:mm:ss NO FILES PURGED
SYSTEM LOAD DECK COMPLETE
hh:mm:ss PUN FILE nnnn TO CPGEN COPY 01 NOHOLD
IPLABLE NUCLEUS NOW ON TAPE ****
```

Note: If any errors are detected while the tape is being written, you must recreate the CP nucleus. To do this, issue the command:

```
generate cp nucleus
```

The procedure restarts at Step 17 where you are asked if you want the virtual=real option and support for the Attached Processor. The following message is issued:

```
THE NUCLEUS LOAD MAP MUST BE SAVED ON DISK FOR IPCS.
WHEN 'NUCLEUS LOADED ON XXXXXX' IS TYPED, ISSUE 'CLOSE PRT' AND IPL
CMS AND READ IN THE LOAD MAP.  IT WOULD BE EXPEDIENT TO LOAD IT ON
AN AREA ACCESSIBLE TO THE IPCS VIRTUAL MACHINE.
FOLLOWING THIS, THE NEW SYSTEM MAY BE IPL'ED.
```

When the nucleus is written on your system residence volume, the resultant load map is placed in your virtual reader by the CLOSE PRT command. This load map should be read in as a uniquely named CMS file. After the new system is operational, the disk-resident load map is required for IPCS. If you have limited space available on the starter system minidisks, read the load map onto the CMS system disk (190). Issue the following commands:

```
link cmssys 190 190 w write
access 190 a
read cpnuc loadmap
```

LOADING A CP NUCLEUS WITHOUT A VIRTUAL=REAL AREA

If you responded "no" when asked if you wanted the virtual=real area, the GENERATE EXEC procedure builds the CP nucleus, writes it to tape, and loads it from the tape.

When the nucleus is written on the system residence volume, the message

```
NUCLEUS LOADED ON volid
```

is issued, where volid is the volume serial number of your system residence volume. The volid is the serial number you specified on the SYSRES macro when you prepared the CP system control file (DMKSYS). If you followed the example in this manual, the serial number of your system residence volume is VMRELn.

The CP load map is placed in the virtual reader of CPGEN. This load map should be read in as a uniquely named CMS file. After your new system is operational, the disk resident load map is required for IPCS. To save the load map, issue the following commands:

```
close prt
ipl 190 parm seg=null
access 194 a
read cpipcs map a
```

If you wish, edit or print the load map. The contents of the load map are described in Part 5. "Updating VM/370".

You may now drain all spooling devices and shut down the system by entering:

```
drain all
shutdown
```

If an error occurs, and you do not receive the "NUCLEUS LOADED ON volid" message, issue the commands

```
cp spool prt off
close printer
```

to allow the error load map to be printed and examine the listing of the load map. A loader error may be indicated. See the VM/370 System Messages for a list of the loader wait state codes.

A loader failure may occur as the result of an error in the real I/O configuration (DMKRIO) file or the system control (DMKSYS) file. Check that the assemblies of these files completed without error. Also check that these files have CSECT cards and that macros are included in the proper sequence. If an "OVERLAY ERROR" occurs, a common cause is insufficient virtual storage for your virtual machine.

After correcting the error, you do not have to shut down the system, but reinvoke the GENERATE EXEC at the point at which the error occurred. (See Step 16.)

LOADING A CP NUCLEUS THAT HAS A VIRTUAL=REAL AREA

If you responded "yes" when asked if you wanted the virtual=real area, the GENERATE EXEC procedure issues the following message:

```
IF YOU HAVE ACCESS TO A DISK WITH THE SAME ADDRESS AS THE SYSRES
DEVICE, DETACH IT. IPL THE NUCLEUS JUST PLACED ON THE TAPE AND
THEN YOU WILL BE ABLE TO SAVE THE LOAD MAP AS DESCRIBED ABOVE.
YOU WILL RECEIVE AN ERROR MESSAGE BECAUSE YOU DO NOT HAVE ACCESS
TO THE SYSRES DEVICE, BUT THE LOAD MAP WILL HAVE BEEN CREATED.

TO LOAD THE CP NUCLEUS JUST CREATED, SHUTDOWN THE SYSTEM AND
THEN IPL THE TAPE.  ONCE THE NUCLEUS HAS BEEN LOADED,
YOU MAY IPL YOUR NEW CP SYSTEM RESIDENCE VOLUME.

NOTE: THERE MUST BE ENOUGH STORAGE ON THE SYSTEM (VIRTUAL OR REAL), TO
CONTAIN THE VIRT=REAL AREA AND THE CP NUCLEUS.
```

Be sure that there is enough storage to load the CP nucleus with a virtual=real area before you load it. The "Specifying a Virtual=Real Machine" section of Part 1 tells you how to determine the amount of real or virtual storage you need. You can load a CP nucleus that has a virtual=real area in either a real or virtual machine.

You IPL the tape containing the CP nucleus. The CP nucleus is on the tape at virtual address 182 for the CPGEN virtual machine.

EXAMINE THE CP LOAD MAP

Edit or print the load map if you wish. The contents of the load map are described in "Part 5. Updating VM/370."

Two external names may be listed as undefined on the load map. The external name DMKSLC is undefined if the virtual=real option is not specified. The external name DMKRNTBL is undefined if there is no entry in the system name table for a 3704/3705 control program (that is, if you did not code a NAMENCP macro for the DMKSNT file). Also, other names may be listed as undefined if other modules were deleted as described in the section "Reducing the Size of the CP Nucleus."

Step 19. IPL the Newly Generated VM/370

If you plan to keep the CPRnLO volume as allocated when using the various system directories, you should use the IPL FMT deck that you punched in Step 14 to reallocate the space on the volume. Put the deck in your card reader, set the load unit switches appropriately, and IPL the reader. Respond to the format/allocate messages as shown in the following example:

```
VM/370 FORMAT/ALLOCATE PROGRAM RELEASE n
ENTER FORMAT OR ALLOCATE:allocate
ALLOCATE FUNCTION SELECTED
ENTER DEVICE ADDRESS (CUU):130
ENTER DEVICE TYPE:device type¹
ENTER DEVICE LABEL:CPRnLO
ENTER ALLOCATION DATA FOR VOLUME CPRnLO
TYPE CYL CYL

        2314        3330      3340      3350
drct 000 000     000 000   000 000   000 000
perm 001 202     001 403   001 348   001 554
perm²            404 807   349 697
end

 ALLOCATION RESULTS

 DRCT 000 000    000 000   000 000   000 000
 PERM 001 202    001 403   001 348   001 554
 DEVICE 130 VOLUME CPRnLO ALLOCATION ENDED
 ENTER FORMAT OR ALLOCATE:
```

IPL the newly created system residence volume by setting the load unit address dials to the real address of the VMRELn system residence volume and pressing the LOAD button. The userid OPERATOR (or whatever userid you specified as the system operator on the SYSOPER macro) is logged on when you IPL. The following messages are issued. Respond as shown.

```
VM/370 VERSION vv LEVEL 00 PUT nnnn; mm/dd/yy hh:mm:ss

NOW hh:mm:ss EDT day mm/dd/yy
CHANGE TOD CLOCK (YES|NO):no
hh:mm:ss START ((COLD|WARM|CKPT|FORCE) (DRAIN))|(SHUTDOWN):cold
hh:mm:ss AUTO LOGON *** OPERATOR USERS=001 BY SYSTEM
hh:mm:ss

DMKCPI952I nnnnK SYSTEM STORAGE
DMKCPI957I STOR sssssK, NUC nnnK, DYN dddddK, TRA tttK, FREE ffffK,
     V=R vvvvvK

hh:mm:ss FILES:  NO RDR,  NO PRT,  NO PUN
hh:mm:ss FORMATTING ERROR RECORDING AREA
hh:mm:ss
```

----

[1]Device types accepted by the Format/Allocate program are: 2314, 2319, 3330, 3330-11, 3340-35, 3340-70, 3350, 2305-1, and 2305-2.

[2]This line gives the required specifications for the 3330 Model 11 and the 3340 Model 70.

At the time you IPL your newly generated VM/370, your system residence volume is formatted according to your specification on the SYSRES macro. Also, if you used the VM/370 directory supplied with the starter system, your starter system volume (CPRnL0) is set up as shown in Figures 27, 28, 29, and 30. Be careful not to modify the 190 and 194 minidisks for the MAINT user. These minidisks and the information they contain are required for applying PUT (Program Update Tape) services to VM/370.

Real Cylinder	Number of Cylinders	Contents
0	1	Unused
1	1	191 minidisk for the IVPM1 user
2	1	191 minidisk for the IVPM2 user
3-7	5	191 minidisk for the RSCS user
8-14	7	191 minidisk for the OPERATOR user
15-18	4	191 minidisk for the CE user
19-28	10	191 minidisk for the MAINT user
29-33	5	191 minidisk for the ECMODE user
34	1	199 minidisk for the MAINT user
35-169	135	190 minidisk for the MAINT user Note: The nucleus occupies the last two cylinders of the minidisk. (Specify cylinder 133).
170-202	33	194 minidisk for the MAINT user - it contains the CP object modules (text decks)

Figure 27. Allocation of the VM/370 Starter System Volume (CPRnL0) when the 2314 Starter System Directory Is Used

Real Cylinder	Number of Cylinders	Contents
0	1	Unused
1	1	191 minidisk for the IVPM1 user
2	1	191 minidisk for the IVPM2 user
3-7	5	191 minidisk for the RSCS user
8-12	5	191 minidisk for the OPERATOR user
13-16	4	191 minidisk for the CE user
17-23	7	191 minidisk for the MAINT user
24-28	5	191 minidisk for the ECMODE user
29	1	199 minidisk for the MAINT user
30-114	85	190 minidisk for the MAINT user Note: The nucleus occupies the last cylinder of the minidisk. (Specify cylinder 84.)
115-141	27	194 minidisk for the MAINT user
142-403	262	Not used.

Figure 28. Allocation of the Starter System Volume (CPRnL0) When the 3330 Starter System Directory Is Used

Real Cylinder	Number of Cylinders	Contents
0	1	Unused
1-2	2	191 minidisk for the IVPM1 user
3-4	2	191 minidisk for the IVPM2 user
5-10	6	191 minidisk for the RSCS user
11-20	10	191 minidisk for the OPERATOR user
21-25	5	191 minidisk for the CE user
26-40	15	191 minidisk for the MAINT user
41-45	5	191 minidisk for the ECMODE user
46-47	2	199 minidisk for the MAINT user
48-287	240	190 minidisk for the MAINT user Note: The nucleus occupies the last two cylinders of the minidisk. (Specify cylinder 238.)
288-347	60	194 minidisk for the MAINT user
Note: Cylinders 348-695 are not used for a 3340 Model 70 system residence volume.		

Figure 29. Allocation of the Starter System Volume (CPRnLO) When the 3340 Starter System Directory Is Used

Real Cylinder	Number of Cylinders	Contents
0	1	Unused
1	1	191 minidisk for the IVPM1 user
2	1	191 minidisk for the IVPM2 user
3	1	191 minidisk for the RSCS user
6-8	3	191 minidisk for the OPERATOR user
9-10	2	191 minidisk for the CE user
11-15	5	191 minidisk for the MAINT user
16-19	4	191 minidisk for the ECMODE user
20	1	199 minidisk for the MAINT user
21-56	35	190 minidisk for the MAINT user Note: The nucleus occupies the last cylinder of the minidisk. (Specify cylinder 34.)
57-64	9	194 minidisk for the MAINT user
65-554	490	Not used.

Figure 30. Allocation of the Starter System Volume (CPRnL0) When the 3350 Starter System Directory Is Used

Step 20. Back Up the Newly Generated VM/370

At this time, back up your new system residence volume. If your real machine has at least 448K bytes of real storage, the tape created in Step 18 is sufficient backup. However, that tape is not sufficient backup for real systems with less than 448K of real storage because that tape cannot be loaded on such systems. It may also be inadequate if you have a large V=R area.

VM/370 systems that run on a real machine with less than 448K bytes of real storage should use the DASD Dump Restore (DDR) service program to create a backup tape similar to the one created in Step 18. The DASD Dump Restore program is described in the *VM/370 Operator's Guide*. If your system residence volume is at address 131 and you labeled it VMRELn, you could use the following DDR control statements to back it up:

```
input 131 device type VMRELn
output 181 device type (tape drive)
dump cpvol
```

The DUMP CPVOL statement causes cylinder 0 and those disk cylinders allocated as PERM or DRCT in Step 2 to be dumped onto the tape.

If you do not wish to use the DDR program to backup your system, you can load the tape produced in Step 18 in a virtual machine. If you load the tape in a virtual machine that virtual machine must have (1) 512K of storage and (2) write-access to the system residence volume, at the address defined for system residence in the SYSRES macro of the CP system control (DMKSYS) file.

When you use the DDR program to backup your system, you do not get a load map when you restore the tape. You do get a load map if you load the tape produced in Step 18.

Step 21. Format the Operator's Virtual 191 Disk

Before any new minidisk area can be used for CMS files, it must be initialized with the CMS FORMAT command, which formats the area into 800-byte blocks. Take care not to format areas which contain data restored from the starter system (such as the 190 and 194 minidisks belonging to the user MAINT). The CMS FORMAT command is described in the _VM/370 CMS Command and Macro Reference_.

Note: After you complete this step, a portion of the starter system is overlaid by the operator's 191 minidisk. If for any reason you wish to IPL the starter system again, you must start from Step 1.

At this time you are logged on as the operator. Use the following procedure to format your virtual disk 191. First, if you have not already loaded CMS, issue:

```
ipl 190 parm seg=null CMS responds with:

CMS VERSION n.n - mm/dd/yy hh:mm
```

Next, enter the following command:

```
access (nodisk
```

The NODISK option prevents CMS from automatically accessing your virtual disk 191. (Accessing 191 at this time would cause an error message to be issued because 191 is not yet initialized, and therefore cannot be used.) After the Ready message is displayed, issue the command:

```
format 191 a
```

The CMS FORMAT command prompts you with the following message:

```
FORMAT WILL ERASE ALL FILES ON DISK 'A(191)'.
  DO YOU WISH TO CONTINUE? (YES|NO):
```

If you respond "yes", CMS prompts you with:

```
ENTER DISK LABEL:
opr191
```

Enter the one-to-six character alphameric label of the virtual disk. You can use whatever label you wish for this virtual disk. In this example, the label is OPR191. CMS then issues:

```
FORMATTING DISK 'A'.
'nn' CYLINDERS FORMATTED ON 'A(191)'.
```

and a Ready message.

Step 22. Format the MAINT User's Virtual 191 Disk

Initialize the 191 disk belonging to MAINT, in the same way you initialized the operator's virtual 191. Log off the system and log on again as userid MAINT, using the password CPCMS. Define your virtual storage to be larger than the location of any discontiguous saved segments that CMS may try to use at this time. (In this example, if you used the DMKSNT provided with the starter system, define your virtual storage as 2048K, or 2M). Then IPL CMS as follows:

```
define storage 2m
ipl 190 parm seg=null
```

When the CMS Ready message is displayed, issue:

```
access (nodisk
```

and continue to format MAINT's 191, using the same procedure you used to format OPR191.

Step 23. Complete the Application of Service

Now that the MAINT 191 minidisk is formatted, you (as the MAINT user) can complete the application of service with the changes supplied on the system Program Update Tape (PUT). Mount the system PUT if it is not already mounted. You must attach the real tape drive to your CMS virtual machine (MAINT); for example:

```
attach 280 to maint as 181
```

and, with the 191 minidisk that belongs to MAINT accessed as your A-disk, rewind and load the tape:

```
vmfplc2 rew
vmfplc2 load
```

CMS responds with the following message:

```
LOADING.....
5749010 06pp38   A1[1]
5749010   EXEC   A1
VMSERV    EXEC   A1
VMFPLC2 MODULE   A1
END-OF-FILE OR END-OF-TAPE
```

To resume the application of service started in Step 13, issue the commands:

```
access 191 c
vmserv restart 5749010 cp nomemo
```

----

[1]Where pp is the PUT number.

VMSERV remaps the PUT because the 191 disk of MAINT is new and does not contain the previously created map. You are prompted for various information concerning your system (including staging area addresses and the service that you want applied). Respond 'NO' when asked if you want service for RSCS. Service for RSCS is applied when RSCS is installed. The "Memo to Users" and the PUT document should be referenced concerning the application of service with the VMSERV EXEC.

Any further SCP service not applied in Step 13 is applied at this time (including service for CMS and IPCS). VMSERV prompts you with:

```
DO YOU WISH TO CONTINUE APPLYING SERVICE? (YES|NO)
```

Respond "No". The following messages are displayed:

```
hh:mm:ss  NO FILES PURGED
SYSTEM LOAD DECK COMPLETE
hh:mm:ss  PUN FILE nnnn TO MAINT  COPY 01 NOHOLD
hh:mm:ss  REWIND COMPLETE
          WHEN THE NEW CMS SYSTEM IS BUILT ISSUE:
          'CLOSE PRT'...(PRINTS THE LOAD MAP)
```

At this point, the CMS nucleus is loaded via IPL from your virtual card reader. When the new CMS nucleus is loaded, control passes to module DMSINI, which then prompts you to specify where to save the nucleus, what your CMS system residence address is to be, and so forth.

In the following example, the 190 you enter is the 190 minidisk that belongs to the user MAINT; it is equivalent to the 190 minidisk that belongs to CPGEN on the starter system. For instance, you specify cylinder 108 as the nucleus cylinder address for a 2314[1] when responding to message DMSINI609R.

The 19E you enter is the address of a user disk that may contain any user-written programs that run under CMS.

If you enter a null line when prompted for version identification and installation heading, CMS uses its own defaults.

```
DMSINI606R SYSTEM DISK ADDRESS = 190
DMSINI615R Y-DISK ADDRESS = 19e
DMSINI607R REWRITE THE NUCLEUS ? yes
DMSINI608R IPL DEVICE ADDRESS = 190
DMSINI609R NUCLEUS CYL ADDRESS = nucleus cylinder address¹
DMSINI610R ALSO IPL CYLINDER 0 ? yes
DMSINI611R VERSION IDENTIFICATION =
DMSINI612R INSTALLATION HEADING =
CMS V n.n - mm/dd/yy press ENTER
```

When the procedure completes its execution, the CMS system residence volume is updated with the most current object modules (text decks) and load modules, and the new CMS nucleus is written on the CMS system residence volume.

If there are any updates for the system assembler on the PUT, the procedure updates that program and creates the corresponding new auxiliary directory.

---

[1]The nucleus will reside on the last cylinder(s) of the minidisk 190. See the notes in the figures showing the allocation of the starter system volumes when the starter system directory is used to determine which nucleus cylinder address to specify.

If there are any updates to the EREP package on the PUT, an updated ERPTFLIB TXTLIB is loaded onto the CMS system disk. The CPEREP EXEC supplied with the system contains a statement:

```
GLOBAL TXTLIB ERPTFLIB EREPLIB
```

which ensures that the ERPTFLIB is searched first and the most current level of each individual EREP module is used.

Step 24. Save CMS

If you used the sample DMKSNT and DMKSYS files supplied with the starter system; and the sample allocations shown in Step 2, you can now save your CMS system.

To save the CMS system, load it and then save it as soon after loading as is possible. If you have not defined your virtual machine as 2M before, issue the command:

```
define storage 2M
```

Then IPL CMS:

```
ipl 190 parm seg=null
```

and press the carriage return to complete the IPL.

GENERATING THE CMSSEG SEGMENT

If you have defined a CMSSEG discontiguous saved segment in your DMKSNT (or used the DMKSNT supplied with the starter system), access your system disk as an extension and create one at this time by issuing:

```
access 190 B/A
cmsxgen 100000
```

where 100000 is the hexadecimal load address of the CMSSEG segment; this location must correspond to the CMSSEG page number in your DMKSNT entries. Figure 31 shows where the CMS segment will be loaded.

The segment name defaults to CMSSEG, but you can load an alternate by specifying the alternate's name (for example, cmsxgen 100000 cmsseg1). There must be an entry in the system name table for the alternate.

CMSXGEN checks that the address specified is greater than or equal to X'20000' and less than 16M. It also checks that only valid characters are specified. If an error is detected, the message

```
DMSCMS095E  INVALID ADDRESS 'address'
```

is issued and command execution is terminated.

Next, CMSXGEN checks that a read/write A-disk is accessed. If an A-disk is not available it issues the following error message and command execution is terminated.

```
DMSCMS064E  NO READ/WRITE A-DISK ACCESSED
```

Then CMSXGEN loads all the text files needed to create the CMS shared segment, starting at the address specified on the command line. If there are any unresolved external references, the CMSXGEN command terminates with the message:

    DMSCMS111E CMSXGEN FAILED DUE TO LOAD ERRORS

To ensure storage protection for the named segment, CMSXGEN assigns a storage key of X'D' (decimal 13) to the segment. CMSXGEN invokes the SETKEY command:

    SETKEY 13 segmentname

If any errors occur during the SETKEY command execution, the message:

    DMSCMS412S CMSXGEN FAILED DUE TO SETKEY ERROR

is issued and CMSXGEN execution is terminated. If no errors have occurred during CMSXGEN processing, the segment is saved via the CP SAVESYS command. If an error occurs at this point, the message:

    DMSCMS141S CMSXGEN FAILED DUE TO SAVESYS ERRORS

is issued and CMSXGEN execution is terminated. Otherwise, the segment is successfully saved, the load map is printed, and the completion message:

    DMSCMS715I CMSXGEN COMPLETE

is issued.

SAVING THE CMS SYSTEM

| Now, you should redefine your virtual storage to 960K and IPL CMS:

```
| define storage 960K
  ipl 190                  -or-       ipl 190 parm seg=segname
```

Where segname is the shared segment created here, if not named CMSSEG.

| When the terminal unlocks, do not press 'ENTER', but immediately issue the command:

    savesys cms

| Then press 'ENTER'. If CMS is successfully saved, the message

```
hh:mm:ss SYSTEM SAVED
CMS VERSION n.n - mm/dd/yy hh:mm
```

is displayed. Your CMS system is now saved; you can issue IPL CMS instead of IPL 190, when you wish to run CMS.

If you named your CMS system something other than CMS, such as CMS1, the entry you made in the system name table would be for CMS1; you would have to save CMS1 (SAVESYS CMS1) and then you could IPL CMS1. For more information about how to save CMS, see the VM/370 System Programmer's Guide or the "Saved Systems" section of this manual.

At this point, use the Installation Verification Procedure (IVP) to test the new system. Log off the userid MAINT and log on again as the userid OPERATOR, using the password OPERATOR. The IVP is described later in Part 3.

Step 25. Obtaining the MSS Communicator Program

If there is an MSS attached to your VM/370 system, and you plan to use the DMKMSS program for communicating between the VM/370 control program and the MSC, enabling VM/370 to dynamically mount and demount MSS volumes, you should obtain the file which will install the DMKMSS program in a VS system. The required file is distributed with the VM/370 control program object code, which you previously restored to MAINT's 194. The first step is to ensure that MAINT has access to its 194. Issue the CMS command:

```
access 194 d/a
```

Next, punch the file; this will install DMKMSS in your VS system. If your VS system is OS/VS1, issue the command:

```
punch mssvs1 jcl
```

If your system is OS/VS2, issue the command:

```
punch mssvs2 jcl
```

The punched output you receive is a series of OS/VS jobs. This file must be saved. When you execute the jobs in your OS/VS system, they will install the DMKMSS program and create a VS operator procedure called DMKMSS, 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 link edits 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/370, replaces the module IEHATLAS and DMKMSS in the program properties table; this adds DMKMSS as an authorized program and removes IEHATLAS. If your installation wishes 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 link edits 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 invoke 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.

Verifying CP and CMS Using the IVP

The Installation Verification Procedure (IVP) for VM/370 exercises CP and CMS to verify that they are working properly. The IVP is contained in two files using the EXEC facility of CMS, and uses two virtual machines in addition to the system operator's virtual machine.

The tests exercise the following areas of CP:

Multiple virtual machine support
I/O spooling
Transferring of spooled data to other virtual machines
Offline I/O operations
Sending of messages to the system operator
Paging operations
Task dispatching and scheduling
Disk I/O support
Automatic warm start following abnormal termination of VM/370
Verifying that the correct EC level is on machines with
 Extended Control-Program Support

The following facilities of CMS are exercised:

Normal CMS command processing
Disk formatting
Copying of files
Creation and modification of files via EDIT command
Assembly of executable programs
Execution of user programs
Creation and execution of user-written commands
Printing and punching of CMS files
Issuing of commands to CP
Use of multilevel nested EXEC procedures
Stacking and unstacking of command and data input from the terminal
Communication with user from EXEC procedures

Several other system facilities, incidental to the primary IVP tests, are exercised. Certain system facilities such as preferred execution options, virtual=real, OS ISAM, and RSCS (Remote Spooling Communications Subsystem) are not exercised by the IVP. The IVP requires operator intervention only when an operational decision is to be made, or to initiate the IVP tests themselves. All file creation, erasure, management, and logoff of the virtual machines (with the exception of the system operator) at test completion is performed without operator or user action.

The IVP tests use only the system-provided facilities. All unique test programs are created, assembled, and subsequently erased by the IVP.

Facilities Required for Each IVP Virtual Machine

All VM/370 configurations are supported. The IVP executes under the control of CMS. The other facilities required are:

- The assembler
- One virtual read/write disk accessed as the A-disk (usually 191)
- 320K of virtual storage (16M for IVPM1)

Starting the IVP

The IVP must be executed to formally complete the initial installation. (See "Variations of the IVP" for post installation testing.) It requires two virtual machines (IVPM1, IVPM2) which must be described in the VM/370 directory.

The directory entries for the IVP virtual machines, IVP1 and IVP2, are included in the VM/370 directory supplied with the starter system; these entries should be included in your own directory. The spooling classes for the reader and the punch must be the same.

You, as the system operator, execute the IVP. To initiate the IVP tests, enter the command:

```
ivp
```

and then answer "yes" or "no" to the following question:

```
ARE YOU THE SYSTEM OPERATOR? ENTER "YES" OR "NO":
```

If you enter the IVP command with no parameters specified and then reply that you are not the system operator, the IVP tests default to the single virtual machine verification procedure. (See "Variations of the IVP.")

Prompting instructions are displayed whenever you must perform an operation or issue a command.

Log on the virtual machine (IVPM1), using the password IVPASS, and IPL CMS to continue the testing procedure:

```
logon ivpm1
ENTER PASSWORD:
ivpass
LOGON AT 09:55:00 EST FRIDAY mm/dd/yy
define storage 16384k
STORAGE=16384K
```

If you created a CMSSEG discontiguous saved segment in Step 24 of the system generation procedure, IPL CMS by issuing:

```
ipl 190
```

If you did not create a CMSSEG segment in Step 24, IPL CMS by issuing:

```
ipl 190 parm seg=null
```

In either case, the system will respond:

```
CMS VERSION n.n mm/dd/yy hh.mm
```

Then you begin the IVP procedure by issuing:

```
ivp 1
```

At this point, the tests begin on virtual machine 1. After the disconnect message is displayed, follow the prompting messages that are displayed. These messages tell you to log on the IVPM2 virtual machine (this may be done on the same terminal), as shown:

```
logon ivpm2
ENTER PASSWORD:
ivpass
LOGON AT 09:58:30 EST FRIDAY mm/dd/yy
```

If you created a CMSSEG discontiguous saved segment in Step 24 of the system generation procedure, IPL CMS by issuing:

```
ipl 190
```

If you did not create a CMSSEG segment in Step 24, IPL CMS by issuing:

```
ipl 190 parm seg=null
```

In either case the system will respond:

```
CMS VERSION n.n mm/dd/yy hh.mm
```

Then you begin the IVP procedure by issuing:

```
ivp 2
```

At this point, the remainder of the tests begin on virtual machine 2. The final phase of the IVP tests consists of displaying, printing, and punching a file which contains the messages generated by IVPM1 after it is disconnected.

Upon completion of the tests, the IVP EXEC procedure logs off. The system abnormal termination test, which consists of forcing an ABEND dump of VM/370 and the subsequent warm start, is an option that you must specify in response to messages that are displayed. For the purpose of installation verification, you should select this option. You are instructed to delay starting the spooling devices (reader, printer, and punch) until after the warm start procedure.

Variations of the IVP

If you wish, you may run the IVP procedure after initial installation, using any one of the following methods:

- Executing the IVP without testing the system abnormal termination
- Executing the IVP using virtual machines other than IVPM1 and IVPM2
- Executing the IVP in a single virtual machine

When you execute the IVP in a single virtual machine, intermachine functions, such as transferring data between virtual machines, are not exercised.

To execute the IVP without testing system abnormal termination:

- Retain the created virtual machines in your VM/370 directory.
- Execute the IVP as described previously under "Starting the IVP," but do not select the "system abnormal termination" option.

To execute the IVP with virtual machines other than IVP1 and IVP2:

- Enter, in place of the command IVP 1:

  IVP 1 userid1

- Enter, in place of the command IVP 2:

  IVP 2 userid2

where userid1 and userid2 identify the two virtual machines in which the EXEC procedures IVP 1 and IVP 2 (respectively) are to be executed.

To execute the IVP in a single virtual machine, enter the command (from any logged-on virtual machine):

```
IVP *
```

This causes the IVP tests to be run in that single virtual machine. Intermachine transfer of data is simulated by transferring virtual punched output to the same virtual machine's virtual card reader.

Interpreting the Test Results

Messages at the end of the IVP test indicate successful completion. If any errors are detected by the IVP, call IBM for software support, because an error usually indicates a serious malfunction of the generated system. The IVP procedure identifies each command being tested just before the command is executed.

Error messages are displayed in a four-line format, for example:

```
*** IVP FAILURE HAS OCCURRED ***
*** COMMAND: STATE IVPTST *
*** EXPECTED RETURN CODE 28
*** RECEIVED RETURN CODE 0
```

These messages indicate that the CMS STATE command had a return code of 0, instead of the expected 28.

All information messages that originate within the IVP are preceded by three asterisks (***).

If any command fails, the IVP procedure terminates. Follow the instructions (if any are given) to log off the virtual machine.

Once the IVP procedure has executed successfully, continue the system generation process by loading and saving discontiguous saved segments, as described in the section that follows.

Loading and Saving Discontiguous Saved Segments

After you have finished generating and testing your new system, you may wish to load and save discontiguous saved segments. The DMKSNT module supplied with the starter system includes entries for saved segments called CMS, CMSSEG, CMSVSAM, CMSAMS, CMSDOS, and INSTVSAM. You may also create your own entries. See the section "Preparing the System Name Table File (DMKSNT)" in Part 2.

Throughout the following discussion, it will be helpful for you to refer to Figure 31, which shows how the CMS discontiguous saved segments are loaded in virtual storage if you use the DMKSNT module supplied with the starter system.

Before a discontiguous saved segment can be attached and detached by name, it must be loaded and saved. The discontiguous saved segment must be loaded at an address that is beyond the highest address of any virtual machine that will attach it. It is the system programmer's responsibility to make sure the saved segment is loaded at an address that does not overlay the defined virtual machine or any other saved segment that may be attached at the same time. The load addresses are determined by the entries you coded in your DMKSNT module.

The load address for the discontiguous saved segment should be just beyond the largest virtual machine that uses it. If the load address is unnecessarily high, real storage is wasted because CP must have segment table entries for storage that is never used.

For example, assume you have five CMS virtual machines in your installation. Also assume that all five use the CMS support for DOS program development and testing which is in a 32K segment named CMSDOS. If each of your five CMS virtual machines has a machine size of 320K, you should load the CMSDOS segment just beyond 320K but below 992K (so as to contain it within 1M). Otherwise real storage would be wasted because CP must maintain segment table entries for each 1024K of storage.

Once the named segment is loaded at the correct address, you can save it by issuing the CP SAVESYS command. To be sure that a discontiguous saved segment has storage protection, set the storage key for the segment accordingly. CMS has a new command, SETKEY, to do this. The CMS SETKEY command is described in the _VM/370 System Programmer's Guide_.

CMS has EXEC procedures that help you load, set storage keys for, and save the CMS discontiguous saved segments. The DOSGEN EXEC procedure loads and saves DOS segments. The VSAMGEN EXEC procedure loads and saves the CMS/VSAM and Access Method Services segments. The CMSXGEN EXEC procedure loads and saves CMSSEG, which contains the CMS Editor, EXEC processor, and OS simulation routines. You used the CMSXGEN EXEC procedure to save CMSSEG in Step 24 of the system generation procedure. The DOSGEN and VMSAMGEN EXEC procedures are described later in this section.

Note: These procedures for loading and saving discontiguous saved segments and the associated text files are 'mode 1' to reduce the amount of storage needed for the master file directory in the user's virtual machine. The system disk must be accessed (any mode, A through G) before loading any discontiguous saved segment. Make sure that this is done _after_ any intermediate IPL of CMS.

Decimal Load Address	Segment Name	Contents			
16320K					
16256K	INSTVSAM	Contains the CMS/DOS discontiguous saved segment used to install VSAM and Access Method Services.	FE0000 (1)	4064 (2)	254 (3)
2112K		Storage unaddressable by the virtual machine.	210000	528	33
		END OF DEFINED VIRTUAL STORAGE			
2048K		Contains the CMS control blocks and free storage used during installation of the segments.	200000	512	32
1984K	CMSDOS	Contains DOS/VS Simulation Routines The area from 1984K to 2016K is shared. The area from 2016K to 2048K is unused.	1F0000	496	31
1472K	CMSAMS	Contains CMS Access Method Services support The area from 1472K to 1856K is shared; the area from 1856K to 1984K is not.	170000	368	23
1088K	CMSVSAM	Contains CMS VSAM support The area from 1088K to 1408K is shared; the area from 1408K to 1472K is not.	110000	272	17
1024K	CMSSEG	Contains the Editor, EXEC, and OS Simulation Routines The entire segment is shared.	100000	256	16
0K		CMS Virtual Machine's Area	000000	0	0
	LEGEND:		(1) HEX LOAD ADDRESS	(2) STARTING PAGE NUMBER	(3) STARTING SEGMENT NUMBER

Figure 31. Sample Layout of Storage for CMS Discontiguous Saved Segments

You may want to compare this storage layout with the sample DMKSNT shown in the section "Preparing the System Name Table File (DMKSNT)" in Part 2. Note that as new releases of VSAM and AMS become available, the number of segments required by these systems may be increased, thus requiring all segment addresses above these segments to be increased accordingly.

Note: Refer to the latest Memo to Users for any possible changes to the sample layout of storage for CMS discontiguous saved segments.

Relationship of Page Numbers, Segment Numbers, and Hexadecimal Addresses:

Since the NAMESYS macro requires you to specify page and segment numbers, and the CMSXGEN, DOSGEN, and VSAMGEN procedures require you to enter hexadecimal addresses, you may find the following reference information useful.

1 Page = 4K = X'1000'
1 Segment = 64K = X'10000'

To convert a page number to a segment number, divide the page number by 16.

Since one segment is 10000 in hexadecimal, then 20000 is segment 2, 100000 is segment 16, 1C0000 is segment 28, and so on.

The recommended procedure for loading and saving the discontiguous saved segments as described in Figure 31 is:

1. During the system generation procedure (Step 23), invoke the CMSXGEN EXEC procedure to load and save the CMSSEG segment at address 100000.

2. Perform the Installation Verification Procedure.

3. Redefine storage to 16M, IPL CMS and access the system disk again.

4. Invoke the DOSGEN procedure to load and save the INSTVSAM segment at address FE0000.

5. Redefine storage to 2112K, IPL CMS and access the system disk again.

6. Define the system name of the CMS/DOS segment to INSTVSAM by issuing the CMS command:

   set sysname cmsdos instvsam

7. Invoke the VSAMGEN EXEC procedure to load and save the CMSVSAM and CMSAMS segments at addresses 110000 and 170000, respectively.

8. Invoke the DOSGEN EXEC procedure to save the CMSDOS segment at address 1F0000. The system name entry in the SYSNAMES table defaults to CMSDOS.

9. Text files must have a filetype of TEXT. For example, after you have updated an object module using VMFASM, the most recent object file has a filetype such as TXTLOCAL. To use that text file here, you must rename it to a filetype of TEXT. If there is currently a text file on the system disk, you may want to rename it too, so that your updated text file (which may reside on another disk) is the one that is loaded.

Loading and Saving the CMS/DOS Segment Called INSTVSAM

Use the DOSGEN EXEC procedure to load and save the CMS/DOS segment called INSTVSAM. This CMS/DOS segment is used only for the installation of VSAM and Access Method Services.

Before you invoke DOSGEN to load the INSTVSAM segment, IPL CMS in a virtual machine with 16M (16384K) of storage. The INSTVSAM segment will be loaded near the top of storage, at 16256K (hexadecimal address FE0000).

You can increase your virtual machine storage size, up to the maximum size defined for it in the VM/370 directory, by entering the DEFINE STORAGE command. After the DEFINE STORAGE command executes, you must reload CMS.

```
define storage 16m
ipl 190
```

At this point, you save the CMS/DOS segment called INSTVSAM by issuing the commands:

```
dosgen fe0000 instvsam
```

Be sure the DOSGEN EXEC is formatted on a CMS minidisk in either 800 or 1K blocks. The format of the DOSGEN command is:

```
DOSGEN address [segmentname]
```

where:

address — is the virtual storage location where the CMS/DOS segment is to be loaded. This address is specified in hexadecimal digits.

segmentname — is the name of the segment to be loaded. You must have previously assigned a name to the CMS/DOS segment with the NAMESYS macro.

DOSGEN checks that the address contains valid characters, is greater than X'20000', and less than 16M. If an error is detected, the message

```
DMSGEN095E INVALID ADDRESS 'address'
```

is issued and the command is terminated.

DOSGEN then checks for a read/write A-disk on which to write the CMS loader work file; if none is accessed, it issues an error message and the command terminates.

```
DMSGEN006E NO READ/WRITE A-DISK ACCESSED
```

Next, DOSGEN loads all the text files needed for DOS simulation. The text files are loaded starting at the address specified on the DOSGEN command. If there are any unresolved external references, DOSGEN terminates with the message

```
DMSGEN111E DOSGEN FAILED DUE TO LOAD ERRORS
```

DOSGEN then assigns a storage key of X'D' to the segment and saves it. If an error is detected, one of the following messages is issued and DOSGEN terminates:

```
DMSGEN412S DOSGEN FAILED DUE TO SETKEY ERRORS
DMSGEN141S DOSGEN FAILED DUE TO SAVESYS ERRORS
```

Otherwise, the segment is successfully saved, the load map is printed, and the completion message:

    DMSGEN715I DOSGEN COMPLETE

is issued.

Note: In the DOSGEN EXEC procedure, the default setting for EMSG is TEXT. With this default, error messages are displayed with the message text only, unless the type code of the message is S or T. To have all messages displayed with both text and identifier, issue the command:

    set emsg on

at the beginning of the procedure.

Loading and Saving the CMSVSAM and CMSAMS Segments

When a VSAM routine is invoked, CMS attaches the discontiguous segments that contain the VSAM simulation routines. When an access method services routine is invoked, CMS attaches the discontiguous segments that contain the access method services routines.

The VSAMGEN installation EXEC procedure helps you load and save the VSAM and access method services segments. The VSAMGEN installation procedure can be used to:

- Install VSAM and access method services for DOS users
- Install VSAM amd access method services for OS users
- Update VSAM amd access method services for DOS users
- Update VSAM and access method services for OS users

Before you invoke VSAMGEN, you must:

- Restore a Release 31 (or later) DOS/VS starter tape to disk. The disks supported by DOS/VS Release 31, 32, and 33 are: 2314/2319, 3330 Model 1 and 11, 3340, 3344, and 3350. In DOS/VS Release 34 the 3330 Model 11 and the 3350 (native mode) are also supported. For a description of the procedure for restoring a starter tape, see the publication DOS/VS System Generation.

- Generate VM/370 and apply the latest level of PLC updates.

- Install the CMS/DOS saved segment called INSTVSAM.

- Define the size of your virtual machine so that it is large enough to contain the six VSAM segments and the eight access method services segments, plus at least two additional segments: one for the CMS control blocks and free storage used during the generation process, and one for the CMS/DOS segment called CMSDOS. However, your machine must not be so large that it overlays the CMS/DOS segment called INSTVSAM.

  The CMSVSAM and CMSAMS segments must be loaded so that they do not overlay each other, the virtual machine that is loading them, or any other discontiguous segment that will be loaded at the same time. CMSVSAM and CMSAMS also must be loaded at addresses lower than the CMS/DOS segment called INSTVSAM and beyond the area where you loaded the CMSSEG segment. CMSSEG contains the CMS Editor, EXEC processor, and OS simulation routines.

  If you follow the example shown in Figure 31, define your virtual machine storage size as 2112K.

- Access your A-disk in read/write mode. The VSAMGEN EXEC procedure writes DOSLIB files, updated object modules, and (for the OS user) CMS text files created from VSAM and access method services object modules to the A-disk. Before the VSAMGEN EXEC procedure completes, it prompts you to specify whether you want to save the DOSLIBs created; if not, it erases them. If you are following the example shown, answer "yes" to this prompting message. The amount of space you need on your A-disk depends on (1) the device type of your A-disk and (2) whether you are an OS or DOS user. The amount of A-disk space required is:

Device Type	Number of Cylinders Required: DOS User	OS User
2314	10	20
2319	10	20
3330 Model 1	6	12
3330 Model 11	6	12
3340	15	30
3344	15	30
3350	3	6

Note: DOS users can use the MAINT 191 supplied with the starter system to generate VSAM and access method services without obtaining additional minidisk space, but they must erase the DOSLIBs when asked. OS users should define sufficient A-disk space as indicated in the table.

- IPL a CMS system with sufficient virtual storage (in the example shown in Figure 31, 2112K):

```
define storage 2112K
ipl 190 parm seg=null
```

- Since the CMS/DOS segment called INSTVSAM is used to install the CMSVSAM and CMSAMS segments, you must define the system name of the CMS/DOS segment. To do this, issue the command:

```
set sysname cmsdos instvsam
```

Then link to and access the restored DOS/VS starter system.

```
cp link dos 342 342 rr read
access 342 d
set dos on d
```

The S-disk must be accessed as a read-only extension of the A-disk.

```
access 190 b/a
```

Invoke the VSAMGEN EXEC procedure by entering:

```
vsamgen
```

VSAMGEN prompts you to enter INSTALL if you are installing VSAM and access method services or UPDATE if you are updating the already installed VSAM and access method services routines. VSAMGEN also requires that you indicate whether you are an OS or DOS user. The prompting messages are:

```
DMSVGN360R ENTER 'INSTALL' 'UPDATE' OR 'RESTART'
DMSVGN361R ENTER EITHER 'DOS' OR 'OS'
```

Respond INSTALL and DOS or OS. If INSTALL is specified for an OS user, all system modules are transferred from the DOS relocatable library to the CMS disk in TEXT file format. This allows the OS installation to dispose of the DOS starter system. If you are both an OS and a DOS user, reply as a DOS user.

VSAMGEN requires a read/write A-disk and checks that one is available. If an A-disk is not available, VSAMGEN issues one of the following error messages and terminates:

```
DMSVGN069E DISK 'A' NOT ACCESSED.
DMSVGN361E DISK 'A' IS NOT A CMS DISK.
```

You are prompted to enter the release level of the DOS/VS starter system you are using:

    DMSVGN369R ENTER RELEASE NUMBER OF THE DOS/VS STARTER SYSTEM:

Enter 34. If you enter an unsupported release number, you receive the message

    DMSVGN379E INVALID - RELEASE 31 OR LATER REQUIRED

and the VSAMGEN EXEC procedure is terminated.

You are now prompted to indicate whether you want to install VSAM or access method services or both:

    DMSVGN264R ENTER 'CMSVSAM' OR 'CMSAMS' OR 'BOTH' FOR GENERATION OF
               NEW SYSTEM(S)

If you receive the message:

    DMSCPY002E INPUT FILE 'VSAM33 DOSLNK *' NOT FOUND

the probable cause is that the system disk is not accessed as a read-only extension of your A-disk.

Next, VSAMGEN issues a message asking you to identify the DOS system relocatable library:

    DMSVGN362R ENTER MODE OF DOS SYSTEM RELOCATABLE LIBRARY DISK:

Respond with the filemode of the DOS/VS starter system disk, which you linked to and accessed before invoking VSAMGEN. If the filemode you enter is not the filemode of a DOS disk, you receive an error message and VSAMGEN processing is terminated.

    DMSVGN361E DISK 'fm' IS NOT A DOS DISK.

The System Name Table (DMKSNT) provided with the VM/370 Starter System was created assuming DOS/VS Release 34.

VSAMGEN checks that the files it requires are on an accessed disk; these required files are shipped on the CMS system disk. VSAMGEN next invokes the CMS/DOS environment and makes the DOS/VS starter system disk available as the DOS/VS system residence volume.

For OS users only, VSAMGEN invokes the RSERV command and creates CMS text files for all the required VSAM and access method services modules in the DOS/VS relocatable library. The OS users of VSAM and access method services thus are not required to keep the DOS/VS starter system disk. OS users can update VSAM and access method services directly on the CMS A-disk. OS users receive the following messages:

    DMSVGN361I CREATING CMS TEXT FILES...
    DMSVGN360I CMS/VSAM TEXT FILES CREATED ON DISK 'A'.

For both OS and DOS users, VSAMGEN then link-edits the VSAM modules and places them in a CMS phase library called CMSVSAM DOSLIB. At this time you receive the following information messages:

    DMSVGN362I LINK-EDITING CMSVSAM...
    DMSVGN363I CMSVSAM DOSLIB CREATED ON DISK 'A'.

Note that DOS linkage editor messages (if they occur) are written to the virtual console as well as the linkage editor map during VSAMGEN processing. Information messages (I-level, return code = 4) may occur due to CMS phase construction and will not cause the EXEC to terminate execution.

Next you must respond to the VSAMGEN message:

```
DMSVGN370R ENTER 'GO' IF SAVED SYSTEM IS TO BE CREATED,
           OTHERWISE 'QUIT'
```

At this time the DOSLIB has been created and if you do not require the link-edit listing, enter 'GO'. However, if you desire to modify the VSAM or AMS systems, enter 'QUIT'. This will preserve the DOSLIB on your A-disk and you can wait until you have the link-edit listing before continuing. If you responded 'BOTH' to message DMSVGN264R, the EXEC will then create the CMSAMS DOSLIB. When you are ready to continue, invoke the VSAMGEN EXEC and reply RESTART to the message:

```
DMSVGN360R ENTER 'INSTALL', 'UPDATE' OR 'RESTART'
```

making sure that the required DOSLIBs are on your A-disk and reply to the succeeding messages as before.

Next you must respond to a VSAMGEN message and enter the load address for CMSVSAM. This address must be entered in hexadecimal characters; it must be an address beyond the size of the virtual machines that will attach it. The prompting message is:

```
DMSVGN363R ENTER LOCATION WHERE CMSVSAM WILL BE LOADED AND SAVED:
```

If your installation DMKSNT was defined to be identical to the example shown in Figure 31 you would enter: 110000 (1088K). If you enter the load address incorrectly, you receive the message:

```
DMSVGN360E INVALID RESPONSE 'location'.
```

VSAMGEN fetches the VSAM modules and loads them at the address you specified. You receive the message:

```
DMSVGN364I FETCHING CMSVSAM...
```

followed by messages from the FETCH command describing the entry point of the modules being fetched. Now the message:

```
DMSVGN371R system IS LOADED, IF ZAPS ARE TO BE APPLIED
           GO INTO 'CP' MODE, ELSE 'NULL'
```

is issued. The system has been loaded into the location requested and if no modifications are required press the ENTER key. If modifications to the system are required, press the ATTENTION key to enter CP mode and make the desired modifications; then enter BEGIN to return to CMS mode and press the ENTER key. Next you must respond to a VSAMGEN message with the name of your installation's VSAM segment.

```
DMSVGN366R ENTER NAME OF SYSTEM TO BE SAVED:
```

In this example you respond:

```
cmsvsam
```

VSAMGEN assigns a storage protection key of X'F' to the first five segments, the shared portion of VSAM, and a storage protection key of X'E' to the last segment, the nonshared portion of VSAM. Then VSAMGEN saves VSAM.

VSAMGEN issues the following message, which indicates that the VSAM discontiguous segments are saved.

DMSVGN365I SYSTEM segmentname SAVED

VSAMGEN then issues the following message to ask you if you want it to erase the DOSLIB that it created:

DMSVGN368R ERASE CMSVSAM DOSLIB? ENTER "YES" OR "NO":

Answer "yes" unless you are using a larger minidisk than MAINT 191. Otherwise, there will not be enough disk space for CMSAMS construction.

Note: If any errors occur during the VSAMGEN procedure, the action required to complete the procedure is as follows:

1. If an error occurred before message DMKSVGN364R was issued you must reinvoke the VSAMGEN EXEC.

2. If an error occurred before any DOSLIBs were created, you can reinvoke the VSAMGEN EXEC and respond 'UPDATE' to message DMSVGN360R. The system will then prompt you for the systems to be generated and ask if tape or cards are to be used for the PTF application. Respond 'CARDS' and when a request for the module name is presented, enter 'END'. The system will then start creating DOSLIBs.

3. If an error occurred after a DOSLIB was created you can reinvoke the VSAMGEN EXEC and respond 'RESTART' to message DMSVGN360R.

4. If an error occurred after the CMSVSAM DOSLIB was created and you had entered 'BOTH' to message DMSVGN264R, you can reinvoke the VSAMGEN EXEC, respond 'CMSVSAM' to message DMSVGN264R and respond 'RESTART' to message DMSVGN360R. Once the CMSVSAM segment is saved, reinvoke the VSAMGEN EXEC, respond 'UPDATE' to message DMKSVGN360R and proceed as in step 2 to create the CMSAMS segment.

VSAMGEN then continues by installing the access method services segments. VSAMGEN link-edits the access method services modules and places them in a CMS phase library called CMSAMS DOSLIB. You receive the following messages:

DMSVGN365I LINK-EDITING CMSAMS...
DMSVGN363I CMSAMS DOSLIB CREATED ON DISK 'A'.

Note: While VSAMGEN is installing access method services, you receive information messages and the linkage editor continues. A return code greater than 4 causes VSAMGEN to terminate.

You are then prompted to enter the load address for the access method services segments.

DMSVGN363R ENTER LOCATION WHERE CMSAMS WILL BE LOADED AND SAVED:

In this example, enter: 170000 (1472K).

VSAMGEN fetches the access method service modules, loads the modules at the designated address, assigns a storage protection key of X'F' to the first six segments, and a key of X'E' to the last two segments. Then VSAMGEN saves the access method services segments. You receive the message

DMSVGN364I FETCHING CMSAMS...

followed by messages from the FETCH command describing the entry point of the modules being fetched.

VSAMGEN then prompts you for the name of the access method services segments.

DMSVGN366R ENTER NAME OF SYSTEM TO BE SAVED:

In this example, enter:

cmsams

Then VSAMGEN saves the access method services segments. A message:

DMSVGN365I SYSTEM segmentname SAVED

is issued to indicate that the access method services segments are successfully saved.

VSAMGEN then issues the following message to ask you if you want it to erase the DOSLIB that it created:

DMSVGN368R ERASE CMSAMS DOSLIB? ENTER "YES" OR "NO":

Answer "yes" unless you are using a larger minidisk than MAINT 191.

Loading and Saving the CMS/DOS Segment Called CMSDOS

To load and save the CMS/DOS segment called CMSDOS, use the same EXEC procedure, DOSGEN, that you used to load and save the CMS/DOS segment called INSTVSAM.

If you plan to load the 64K CMS/DOS segment called CMSDOS at 1984K (1F0000), as in the example shown in Figure 31, your virtual machine size must be at least 2112K. Access the CMS system disk as anything (A through G).

Save the CMS/DOS segment called CMSDOS by issuing the command:

dosgen 1F0000 cmsdos

DOSGEN checks that the address contains valid characters, is greater than X'20000', and less than 16M. If an error is detected, the message

DMSGEN095E INVALID ADDRESS 'address'

is issued and the command is terminated.

DOSGEN then checks for a read/write A-disk on which to write the CMS loader work file; if none is accessed, it issues an error message and terminates.

DMSGEN006E NO READ/WRITE A-DISK ACCESSED

Next, DOSGEN loads all the text files needed for DOS simulation. The text files are loaded starting at the address specified on the DOSGEN command. If there are any unresolved external references, DOSGEN terminates execution with the message

    DMSGEN111E DOSGEN FAILED DUE TO LOAD ERRORS

DOSGEN then assigns a storage key of X'D' to the segment and saves it. If an error is detected, one of the following messages is issued and DOSGEN terminates execution:

    DMSGEN412S DOSGEN FAILED DUE TO SETKEY ERRORS
    DMSGEN141S DOSGEN FAILED DUE TO SAVESYS ERRORS

Otherwise, the segment is successfully saved, the load map is printed, and the completion message:

    DMSGEN715I DOSGEN COMPLETE

is issued.

The system name entry in the SYSNAMES table defaults to CMSDOS.

At this point you have completed the procedures for loading and saving the discontiguous saved segments that are defined in the DMKSNT module supplied with the starter system. If you wish to load and save other discontiguous saved segments, you must have created other DMKSNT entries for them, and you now must repeat these procedures for those entries.

You can now go on to generate and install the RSCS component of the VM/370 SCP, if you wish.

Generating and Installing RSCS

General Information

The data and control files required to generate and install RSCS are contained in the starter system maintenance areas of the VM/370 system.

RSCS data, consists of the following. On MAINT 194:

File	Contents
DMTxxx TEXT	The pre-assembled nucleus modules and supervisor routines required to generate RSCS are: DMTAKE, DMTASK, DMTASY, DMTCMX, DMTCOM, DMTCRE, DMTDSP, DMTEXT, DMTGIV, DMTINI, DMTIOM, DMTMAP, DMTMGX, DMTMSG, DMTPST, DMTQRQ, DMTREX, DMTSIG, DMTSTO, DMTSVC, DMTVEC, and DMTWAT.
DMTAXS TEXT	The spool file access method supervisor task.
DMTLAX TEXT	The communication line allocation supervisor task.
DMTNPT TEXT	The Nonprogrammable Terminal (NPT) line driver module.
DMTSML TEXT	The Spool MULTI-LEAVING (SML) line driver module.
DMTLOAD EXEC	The loadlist EXEC file. This file is required to generate an RSCS nucleus on the RSCS system disk.
DMTSYS ASSEMBLE	The RSCS configuration table module. This file must be assembled with the COPY files you create to define your RSCS configuration (the AXSLINKS, LAXLINES, and TAGQUEUE COPY files).
DMTMAC MACLIB	The file containing all the macros needed to assemble the RSCS source files.
DMTRn0[1] CNTRL	The control file that is needed to assemble the configuration table via the VMFASM EXEC procedure.
DMTMAC EXEC	An EXEC file used to generate the DMTMAC MACLIB.

On the VM/370 SOURCE tape:

DMTxxx ASSEMBLE	All the source files for RSCS. There is an ASSEMBLE file for each TEXT file previously listed.

Also, the SOURCE tape contains all the macro and copy files included in the DMTMAC macro library.

---

[1] DMTRn0 may be DMTR10, DMTR20, DMTR30 and so forth, depending on the release level.

The AXSLINKS COPY file is a list of 1 to 64 GENLINK macro statements. The GENLINK macro defines the attributes of a link. The first GENLINK macro in the AXSLINKS file must contain the ID of the local RSCS station. You must also code the TYPE=driverid operand with a valid filename on this first GENLINK macro. You should code additional GENLINK macros, with no operands, for links you may want to define temporarily during an operating session.

The format of the GENLINK macro is:

```
GENLINK  ID=linkid,TYPE=driverid[,CLASS=c]
                               [,KEEP=holdslot]
                               [,LINE=vaddr]
                               [,TASK=taskname]
```

where:

ID=linkid is a 1- to 8-character alphameric location ID of the remote location to be served by the link. If this operand is not specified, the ID defaults to "undefined."

TYPE=driverid
: is a CMS filename of a file which is the TEXT file for the line driver program to be used to process files for the link. The appropriate line driver program to be specified depends on the type of remote telecommunications facilities to be used.

  The TYPE operand must be specified if ID=linkid is coded. If the TYPE operand is omitted, TYPE defaults to "undefined".

CLASS=c is the spooling class(es) of the files which can be processed by the active link. You can specify up to four spooling classes (single alphameric characters from A to Z and 0 to 9) with no intervening blanks, or *, which means all spool file classes may be processed. If the CLASS operand is not specified, the default is "*".

KEEP=holdslot
: is a decimal number from 0 to 16 which designates the number of virtual storage file tag slots to be reserved for exclusive use by the link. If the KEEP operand is omitted, a default "holdslot" value of 2 is assumed.

LINE=vaddr
: designates the virtual device address of a permanent telecommunications line port to be used for processing files on the link. If the LINE operand is omitted, the default is "undefined".

TASK=taskname
: is a 1- to 4-character alphameric identifier. It specifies the task name to be used by the line driver associated with the link. If the TASK operand is omitted, the default is "undefined".

CREATING THE LAXLINES COPY FILE

The LAXLINES COPY file defines the virtual device addresses of the telecommunications line ports (attached to the RSCS virtual machine) which may be shared among the various active links. Such line ports must be switchable, and the links which are to use these line ports must have switched line ports available at their remote stations.

The LAXLINES COPY file consists of a list of GENLINE macro statements, one for each line port. The format of the GENLINE macro is as follows:

GENLINE	LINE=vaddr

<u>where:</u>

LINE=vaddr
: is the virtual device address of a switched telecommunications line port available to RSCS.

CREATING THE TAGQUEUE COPY FILE

The TAGQUEUE COPY file specifies the total number of virtual storage tag slots to be available to RSCS. The format of the GENTAGQ macro is:

GENTAGQ	NUM=totslots

<u>where:</u>

NUM=totslots
: is a decimal number from 32 to 512 that defines the total number of virtual storage tag slots to be made available to RSCS for storing information on files enqueued for transmission or received from remote stations. Files which cannot be enqueued for transmission because no free virtual storage tag slots are available are left pending, and are automatically accepted and enqueued at a later time as virtual storage tag slots become available.

: You must specify a number for totslots that is at least as large as the sum of linkid tag slots defined or implied by the KEEP=holdslot operand of all the GENLINK macros.

Generation Procedure for RSCS

Before you perform the RSCS generation procedure, be sure you have the following:

- The VM/370 system PUT tape
- A VM/370 directory entry for your RSCS virtual machine
- A VM/370 directory entry for the software system support virtual machine (for example, the MAINT entry supplied with the VM/370 starter system directory).

You can use a 2314, 3330, 3340, or 3350 disk as the RSCS system disk. The RSCS nucleus occupies two cylinders on a 2314 or 3340, and one cylinder on a 3330 or 3350 disk.

The following system generation procedure for RSCS assumes you have the MAINT virtual machine supplied with the VM/370 starter system in your VM/370 directory.

STEP 1. LOG ON AS MAINT AND IPL CMS

To build the RSCS nucleus, you must log on the software system support virtual machine (MAINT) and IPL the CMS system.

```
logon maint cpcms
ipl 190
```

STEP 2. APPLY THE RSCS UPDATES FROM THE VM/370 PROGRAM UPDATE TAPE

Press the carriage return to complete IPL.

Mount the system PUT and attach to MAINT as 181. Service for RSCS is initiated by issuing the commands:

```
access 191 c
vmserv restart 5749010 rscs noipl
```

The keyword "rscs" allows the procedure to load what is required (from the system PUT) for RSCS.

You are prompted for various information concerning your system (including staging area addresses and the service that you want applied). Respond 'NO' when asked if you wish service for IPCS. Service for IPCS has already been applied. The user memos and the PUT document should be referenced concerning the application of service with the VMSERV EXEC.

After service has been applied, you will receive this message:

```
DO YOU WISH TO BUILD RSCS SYSTEM -- RESPOND (YES|NO)
```

You must respond "no".

STEP 3. FORMAT THE RSCS SYSTEM DISK

You must link to the RSCS system disk and format it. If you used the VM/370 directory entry for the RSCS virtual machine supplied with the starter system, your LINK command is:

```
link to rscs 191 as 195 w pass= password
```

This makes the RSCS system disk (address 191 in the RSCS virtual machine) available at virtual address 195 in the MAINT virtual machine. Remember the address you specify for MAINT. You must use this same virtual address later in the RSCS generation procedure (you use 195 when you invoke GENERATE again to build the RSCS nucleus in Step 7). Then, format the RSCS system disk.

```
format 195 a
```

Next, format the RSCS system disk again. You must use the recompute function of the FORMAT command to make the last cylinders of the RSCS system disk unavailable to the CMS file system. The last one or two cylinders contain the RSCS nucleus. If the RSCS system disk is a 2314 or 3340, the last two cylinders are needed for the nucleus. For a 3330 or 3350, only the last cylinder is needed. The FORMAT command for a 2314 or 3340 RSCS system disk is:

```
format 195 a 3 (recomp
```

The FORMAT command for a 3330 or 3350 RSCS system disk is:

```
format 195 a 4 (recomp
```

STEP 4. CREATE THE COPY FILES THAT DEFINE YOUR RSCS SYSTEM

You can use the CMS Editor to create the three COPY files you need: AXSLINKS, LAXLINES, and TAGQUEUE. The macros you need to include in these files are described in a preceding section, "Defining Your RSCS System."

When you code the GENLINK macros for the AXSLINKS COPY file, you may include GENLINK macros with no operands. These macros reserve extra link table entries which can be defined later with the RSCS DEFINE command. Also, the local linkid (local location ID) must be specified on the first GENLINK macro in the AXSLINKS file.

When you code the GENTAGQ macro for the TAGQUEUE COPY file, remember that the total number of tag slots must be equal to or greater than the number of holdslots defined either explicitly or implicitly in the AXSLINKS COPY file.

The following example shows the creation of the AXSLINKS, LAXLINES, and TAGQUEUE COPY files:

```
edit axslinks copy
NEW FILE:
EDIT:
input
INPUT:
*copy axslinks
 genlink id=mylocid,type=dmtnpt
 genlink id=newyork,class=a,keep=2,line=079,task=m1,type=dmtnpt
 genlink id=sanfran,class=a,keep=2,line=07a,task=m2,type=dmtsml
 genlink id=london,class=a,keep=4,line=07b,type=dmtsml
 genlink
 genlink
 genlink
EDIT:
file
R;
```

On the TYPE operand, DMTNPT refers to the Nonprogrammable Terminal (NPT) line driver program and DMTSML refers to the Spool MULTI-LEAVING (SML) line driver program.

```
edit laxlines copy
NEW FILE:
EDIT:
input
INPUT:
*copy laxlines
 genline line=079
 genline line=07a
 genline line=07b
 genline line=07c
 genline line=07d
 genline line=07e
 genline line=07f
EDIT:
file
R;

edit tagqueue copy
NEW FILE:
EDIT:
input
INPUT:
*copy tagqueue
 gentagq num=32
EDIT:
file
R;
```

STEP 5. CREATE THE DMTLOC MACRO LIBRARY

Using the COPY files created in Step 4, generate a CMS macro library called DMTLOC MACLIB, as follows:

```
maclib gen dmtloc axslinks laxlines tagqueue
```

If you must change your RSCS configuration at a later time, you have to change the COPY file and then generate a new DMTLOC macro library.

STEP 6. ASSEMBLE THE CONFIGURATION TABLE (DMTSYS)

Before you assemble the configuration table module, access the staging area disk (194) as an extension of the RSCS system disk.

```
access 194 b/a
DMSACC723I B (194) R/O
```

Now, assemble the RSCS configuration table module (DMTSYS) using the VMFASM EXEC procedure. Specify DMTRn0 as the control file. This control file identifies DMTLOC as the macro library to be used during the assembly. The DMTRn0 control file is on the 194 minidisk. Invoke VMFASM as follows:

```
VMFASM DMTSYS DMTRn0
```

Note: If an error occurs due to an incorrectly coded macro, correct the macro and restart at Step 5.

STEP 7. INVOKE GENERATE TO CREATE THE RSCS NUCLEUS

Invoke GENERATE to build the RSCS system nucleus as follows:

```
generate rscs build
```

GENERATE prompts you for the operands it needs to link to the RSCS system disk:

```
ENTER RSCS SYSTEM DISK LINK PARAMETERS:
USERID VADDR1 VADDR2
```

If you are following the examples in this procedure, you enter

```
rscs 191 195
```

The value you enter for vaddr2 must be the same value you specified for vaddr2 on the LINK command issued in Step 3; otherwise, the link is not allowed. The userid you specify is the userid of your RSCS virtual machine and the virtual address you specify for vaddr1 is the virtual device address of the RSCS system disk in the RSCS virtual machine. You will be prompted for the write password of the disk.

GENERATE then links to the RSCS system disk and copies four files (DMTNPT, DMTSML, DMTAXS, and DMTLAX) to it. You receive the following message:

```
TRANSFERRING 'RSCS' DISK RESIDENT TEXT...
```

Finally, GENERATE builds and loads the RSCS nucleus. You receive the following messages:

```
DMTINI406R SYSTEM DISK ADDRESS = 195
DMTINI407R REWRITE THE NUCLEUS ? yes
DMTINI409R NUCLEUS CYL ADDRESS = 003   (for a 2314 or 3340, or 004
                                        for a 3330 or 3350)
DMTINI410R ALSO IPL CYLINDER 0 (YES|NO) ? yes
```

For this example, respond as shown. You respond with the address of the RSCS system disk, in this case 195. You always respond with the same address you specified as vaddr2 when you were prompted for link parameters. The nucleus cylinder address depends on the device type of the RSCS system disk.

You receive the message

```
DMTAXS103E FILE 'spoolid' REJECTED -- INVALID DESTINATION ADDRESS
```

This message reflects the purging of the RSCS nucleus from the card reader.

At this time you have an RSCS system generated and written to the RSCS system disk, as indicated by the message:

```
DMTREX000I RSCS (VER n, LEV n, mm/dd/yy) READY
```

You can now log on the RSCS virtual machine, IPL the RSCS system disk and start your RSCS operations. See the _VM/370 RSCS User's Guide_ for information about using RSCS.

Generating and Installing IPCS

The VM/Interactive Problem Control System Extension (VM/IPCS Extension) program product can be ordered separately. It is not to be confused with the Interactive Problem Control System (IPCS) component of VM/370. VM/IPCS Extension provides installations with expanded facilities for reporting and diagnosing software failure. If you have installed this program, see the _VM/370 Interactive Problem Control System Extension User's Guide and Reference,_ Order No. SC34-2020.

Generation Procedure for IPCS

The IPCS function exists on the Starter System S-disk and requires no special installation procedure. Should it become necessary to update the IPCS function the following procedure may be used.

Before you perform IPCS generation procedure, be sure you have the VM/370 directory entry for your IPCS virtual machine.

STEP 1. LOG ON AS MAINT AND IPL CMS

To load the IPCS command modules and EXEC procedures, log on as the software support virtual machine (MAINT) and IPL the CMS system.

```
logon maint cpcms
ipl 190
```

STEP 2. FORMAT THE IPCS VIRTUAL MACHINE 191 DISK

If the IPCS virtual machine (OPERATNS in our example) 191 disk has already been formatted, you may proceed directly to Step 3. If it has not, you must link to the IPCS virtual machine 191 disk and format it. The LINK command you use is:

```
link to operatns 191 as 195 w password
```

where password is the WRITE password of the OPERATNS 191 disk.

This makes the IPCS virtual machine 191 disk available at virtual address 195 to the MAINT virtual machine. Next, format the IPCS virtual machine 191 disk.

```
format 195 a
```

Reply 'YES' to the prompt "DO YOU WISH TO CONTINUE" and give a six-character label when requested.

STEP 3. APPLY THE IPCS UPDATED FILES FROM THE PUT

The required service for IPCS has been applied in Step 23. The IPCS system is now ready for use.

Part 4. Generating the 3704/3705 Control Program

If you do not want to support a 3704/3705 control program under VM/370 control, disregard Part 4.

Part 4 describes the procedures you must follow to generate, test, and run a 3704/3705 control program with VM/370. It includes the following information:

- Introduction
- Planning Considerations
- Generating and Loading the 3704/3705 Control Program

You should generate a VM/370 system that supports the 3704/3705 first. "Part 1. Planning for System Generation" contains a section, "Generating a VM/370 System that Supports the 3704 and 3705," that tells you what you must include in your VM/370 system. When you have a VM/370 system that supports the 3704/3705, use the information in this part to generate and test a 3704/3705 control program that runs under VM/370 control.

Introduction to the IBM 3704 and 3705 Communications Controllers

The IBM 3704 and 3705 Communications Controllers are programmable units. The Emulation program can be generated to execute in 3704/3705 storage.

The Emulation program (EP) permits existing teleprocessing systems, including VM/370, that use the IBM 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 execute without change on the 3704/3705.

In this publication, the term "3704/3705 control program" refers to the EP control program.

VM/370 supports the:

- IBM 3704 Communications Controller, Models A1-A4
- IBM 3705-I Communications Controller, Models A1-D8
- IBM 3705-II Communications Controller, Models E1-H8

when attached to a VM/370 processor. Three terminals are supported: 1050, 2741, and CPT-TWX 33/35. The 3767 terminal (operating as a 2741) is supported by lines in EP mode, and the 3101 display terminals are supported as CPT-TWX 33/35.

The minimum required by an EP control program is 16K.

Planning Considerations for the 3704/3705 Control Program

When planning for the installation of IBM 3704 and 3705 Communications Controllers, be sure that you are familiar with device characteristics, have the the appropriate publications and support package, and have a VM/370 system that supports the 3704/3705.

Related Publications

The _Introduction to the IBM 3704 and 3705 Communications Controllers_, Order No. GA27-3051, describes the general functions of the 3704 and 3705. It is a prerequisite publications for generating a 3704/3705 control program under VM/370.

If you are installing Version 4 of the Network Control Program/VS, the _IBM 3704 and 3705 Communications Controllers, Network Control Program/VS Generation and Utilities Guide and Reference Manual (for OS/VS and DOS/VS TCAM Users)_, Order No. GC30-3007, is a corequisite publication. If you are installing Version 4 of the Network Control Program/VS, the _IBM 3704 and 3705 Communications Controllers, Network Control Program/VS Generation and Utilities, Guide and Reference Manual (for OS/VS and DOS/VS VTAM Users)_, Order No. GC20-3008, is a corequisite publication. You must refer to one of these publications in order to code the 3704/3705 control program generation macros. Throughout Part 4, these publications are referred to as the _3704 and 3705 Generation and Utilities Guide_.

3704 and 3705 Support Package

Before you can generate a 3704/3705 control program, you must have the following OS/VS Network Control Program Support Package. This is the only 3704/3705 support package that contains the CMS file required for generating and loading the 3704/3705 control program under VM/370. The support package is:

- IBM 3704/3705 Emulation Support and System Support Package (EP/VS SCP) for OS/VS (order No. 5744-AN1). VM/370 supports this package in emulation mode only.

This package contains the following basic material:

- A Program Directory
- _IBM 3704 and 3705 Communications Controllers, Network Control Program/VS Generation and Utilities, Guide and Reference Manual (for OS/VS and DOS/VS TCAM Users,_ Order No. GC30-3007

  -- or --

  IBM 3704 and 3705 Communications Controllers, Network Control Program/VS Generation and Utilities, Guide and Reference Manual (for OS/VS and DOS/VS VTAM Users), Order No. GC30-3008

- *IBM 3704 Control Panel Guide*, Order No. GA27-3086.
- *IBM 3705 Control Panel Guide*, Order No. GA27-3087.

Machine Readable Material

- Magnetic tape containing the macros and modules of the 3704/3705 control program and the OS/VS system support programs.

VM/370 Support of the 3704 and 3705

The IBM 3704/3705 Communications Controllers can support:

- Up to 352 low speed start-stop lines
- Up to 60 medium speed synchronous lines
- Line speeds from 45.2 baud to 56.0K baud
- Modem capability within the 3704/3705
- Limited-distance "hard-wire" capability
- 16K to 256K internal storage
- Remote 3275, 3276, 3277 and 3278 terminals with optional 3284, 3286, 3277, 3288 and 3279 printers (EP mode only)
- Remote 2780 terminals (EP mode only)
- Emulator Program (EP) Version 3.0.

VM/370's support of the 3704/3705 does not include:

- Remote 3704/3705 Communications Controllers

EMULATION PROGRAM (EP) WITH VM/370

The EP 3704/3705 control program under VM/370:

- Emulates 2701, 2702, and 2703 operations
- Attaches to a System/370 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

This support is equivalent to that provided in Release 1 of VM/370. The CP DIAL command and the TERMINAL APL ON and APL OFF command lines are supported. However, Release 2 and above of VM/370 provides additional support:

- Service programs and special CMS commands allow you to easily generate the EP control program in a CMS virtual machine.
- The CP NETWORK command allows you to load or dump the 3704/3705 and provides for automatic dumping and reloading if a fatal error occurs.

Generating and Loading the 3704/3705 Control Program

Several commands and EXEC procedures generate and load the 3704/3705 control program. These commands and EXEC procedures are executed in a CMS virtual machine. The commands are a part of the VM/370 system and are distributed with it.

A special version of the IBM 3704/3705 Network Control Program Support Package for OS/VS, Order No. 5744-AN1 EP/VS SCP is available from PID for use under VM/370. This version of the 3704/3705 package contains two CMS EXEC procedures for generating and loading the 3704/3705 control programs.

This section describes the step by step procedure for generating and loading the 3704/3705 control program. Each EXEC procedure and command is described as it is used. The action required at each step is summarized first and then explained in detail. The preceding "Planning Considerations" section, lists all the documentation, physical devices, programming, and other materials you need before starting to generate the 3704/3705 control program.

Step 1. Log On the VM/370 System

VM/370 supports the EP type of control program. The VM/370 system that you load also must have been generated with:

- The IBM 3704 or 3705 Communications Controllers specified on a RDEVICE system generation macro.
- The NAMENCP macro coded to create an entry in the VM/370 system name table (DMKSNT) for the 3704/3705 control program.
- Space reserved on a CP-owned volume to contain a copy of the 3704/3705 control program.

These VM/370 system generation requirements are described in Part 1.

Step 2. Set Up a CMS Virtual Machine

You must IPL a CMS virtual machine and be sure that the necessary devices are attached.

The 3704/3705 control program is generated using commands and EXEC procedures that execute in a CMS virtual machine. The CMS virtual machine must have the following resources:

- At least 1024K of virtual storage.
- One tape drive (9 track, 800 or 1600 bpi).
- Space available on the CMS A-disk (120 cylinders of a 3330 disk, all 203 cylinders of a 2314 disk, 300 cylinders of a 3340 disk, or 60 cylinders of a 3350 disk.

If the CMS virtual machine does not have these resources, use the CP DEFINE command to redefine the size of your virtual storage or send a message to the operator requesting him to attach the tape or disk device you need.

Be sure that there are no files on the A-disk with a filetype of COPY or TEXT. Use the CMS RENAME command to temporarily change such filetypes. A naming conflict can terminate the installation procedure for the distribution tape.

You need CP command privilege classes A, B, and G to install the 3704/3705 control program and, if you use the NETWORK TRACE command while testing the 3704/3705 control program, you need command privilege class F. Do not use class F unless you need it; for class F, I/O error recording is not done automatically. Check with the system administrator to ensure that your VM/370 directory entry has the appropriate command privileges.

Step 3. Load the IBM 3704/3705 Control Program Distribution Tape Files onto a CMS Disk

Use CMS commands to position the distribution tape at the proper file and to create CMS disk files from the tape files. The first file created from the tape files is an EXEC procedure that processes the rest of the tape files and creates the CMS disk files.

If you cannot mount the distribution tape yourself, send a message to the operator and have him mount the correct tape. The distribution tape contains ten files. The tenth file contains the INSTEP and ARNGEND EXEC procedures, which create the necessary CMS files from the other tape files.

Use the CMS TAPE command to position the tape at the beginning of the tenth file:

```
tape fsf 9
```

Then, use the CMS TAPPDS command to create the INSTEP EXEC A1 and ARNGEND EXEC A1 files from the tenth file:

```
tappds * exec
```

If the files are successfully created, the responses

```
FILE 'ARNGEND EXEC A1' COPIED
FILE 'DMSARD  EXEC A1' COPIED
FILE 'DMSARX  EXEC A1' COPIED
FILE 'DMSGRN  EXEC A1' COPIED
FILE 'DMSTMA  EXEC A1' COPIED
FILE 'INSTEP  EXEC A1' COPIED
```

appear on the terminal. Before you invoke the INSTEP EXEC procedure, you should obtain access to mode 1 files on the CMS system disk. You can do this by accessing it as an extension of your A-disk; for example, if the S-disk is at virtual address 190, and you currently have a disk accessed as mode C, you issue the command

```
access 190 c/a
```

Invoke the INSTEP EXEC procedure to load all the necessary files and generate the 3705 Assembler:

```
instep
```

The INSTEP EXEC procedure generates the 3705 Assembler and creates the macro and text libraries that are needed to generate a 3704/3705 control program. The INSTEP EXEC procedure sends messages to the terminal to indicate its progress.

INSTEP issues the message

```
BUILD STAGE ONE MACLIB
LOADING 'GEN3705 MACLIB'
```

and uses the third tape file to create the CMS file GEN3705 MACLIB A1. It issues the messages

```
BUILD STAGE TWO MACLIBS
LOADING 'MAC3705 MACLIB'
```

using the fifth tape file to create the CMS file MAC3705 MACLIB A1. Using the sixth tape file, INSTEP creates the CMS file OBJ3705 MACLIB A1, and issues the messages

```
BUILD STAGE TWO TXTLIB

LOADING 'OBJ3705 MACLIB'
RENAME OBJ3705 MACLIB A1 OBJ3705 TXTLIB A1
```

Finally, INSTEP issues the message

```
LOAD 3705 ASSEMBLR FILES
```

and loads the assembler text files from tape via the TAPPDS command. The files copied are listed off in messages in the form:

```
FILE 'fn EPTAPE A1' COPIED
```

The ARNGEND EXEC procedure is invoked by INSTEP to generate the 3705 Assembler, after issuing the message

```
BUILD 3705 ASSEMBLR MODULES.
```

The ARNGEND EXEC procedure displays the following status and error messages:

```
ENTER TARGET DISK MODE FOR 3705 ASSEMBLR MODULES
DEFAULTS TO S-DISK IF NONE ENTERED
```

You enter the mode letter of the disk that will contain the 3705 assembler modules when the assembler is used. This may be a different disk then the one on which the modules now reside. If you enter a mode letter, ARNGEND uses that mode letter as the "targetmode" operand of the GENDIRT command when it creates the auxiliary directory for the 3705 assembler. If you do not specify a mode letter, S is assumed by the GENDIRT command.

If the 3705 assembler text files are not loaded successfully, or if the assembler generation procedure fails, the following message appears.

    ASM3705 GEND FAILED

When the last message

| END OF EPTAPE INSTALL

appears on the terminal, the distribution tape is no longer needed. At this time, the 3705 Assembler program, the macro libraries for the Stage 1 and Stage 2 generation procedures, and the text library for the Stage 2 generation procedure all exist on the CMS A-disk.

Note: You may find it helpful to dump the contents of the A-disk to tape at this time. If you save the tape dump, you have the pre-Stage 1 files. If errors are later encountered, you may need these files.

Step 4. Code the 3704/3705 Control Program Macro Instructions

Code the 3704/3705 control program macro instructions and place them in a CMS file. Use the CMS Editor to create the file, which must have a filetype of ASM3705. VM/370 recommends that you assign the same filename to this CMS file as was specified previously in the NAMENCP macro. If the SAVE option is to be specified on the GEN3705 command, the filename must be the same. This ASM3705 file is used as input to Stage 1 of the 3704/3705 control program generation procedure.

Use the *3704 and 3705 Generation and Utilities Guide* to code the macro instructions. Follow the macro instruction formats described in that publication except where suggestions and requirements are indicated in the following paragraphs.

BUILD MACRO INSTRUCTION

The BUILD macro must be the first macro in the CMS file. Figure 32 lists the operands which VM/370 requires, recommends, or does not support. For all other operands, refer to the *3704 and 3705 Generation and Utilities Guide*.

Operand	Comments
LOADLIB=dsname OBJLIB=dsname	Required by the BUILD macro, but does not apply to VM/370. Specify a valid dsname.
JOBCARD={YES (default), NO}	VM/370 recommends JOBCARD=YES for EP.
NEWNAME={NCP001, PEP001, symbol}	VM/370 requires that the value of NEWNAME be the same as the name previously specified in the NAMENCP macro and the name that subsequently will be specified in the SAVENCP command. Also, if the GEN3705 command is to be issued with the SAVE option, the value of NEWNAME must be the same as the "fname" specified on the GEN3705 command.
QUALIFY={symbol, NONE, SYS1 (default)}	VM/370 requires the default value.
UT1=dsname UT2=dsname UT3=dsname	VM/370 ignores these operands.

Figure 32. BUILD Macro Operands for VM/370

CSB MACRO INSTRUCTION

The CSB macro instruction is required. See the _3704 and 3705 Generation and Utilities Guide_ for more information about coding the CSB macro instruction.

GROUP AND LINE MACRO INSTRUCTIONS

These macros describe the physical and logical configuration of the communications network accessed through the 3704/3705 control program. Since VM/370 does not support either multi-drop lines or cluster control units, the 3704/3705 configuration for VM/370 is generally simple, with only one GROUP macro for each communications scanner. For VM/370, it is often easiest to specify most of the operands of the LINE macro on the GROUP macro. The _3704 and 3705 Generation and Utilities Guide_ describes the GROUP and LINE macro instructions in detail and lists all the operands of the configuration macros, telling you where each operand is described and also where it may be coded.

VM/370 requires the DUPLEX and FEATURE operands. These operands allow VM/370 to detect and respond to a terminal attention interrupt and to recognize when a data set has been hung up. For the GROUP macro, VM/370 requires the default value for the REPLYTO operands and recommends the default value for the TEXTTO operand.

GENEND MACRO INSTRUCTION

The GENEND macro indicates the end of the 3704/3705 macro input file. It must be the last macro in the CMS file you are building as input to Stage 1.

SPECIAL MACRO CODING CONSIDERATIONS FOR THE EMULATION PROGRAM (EP)

There are no strict dependencies between the host access method and the emulation program; consequently, few guidelines are necessary for an emulation program generation. However, be careful when configuring emulator lines for CPT-TWX terminals. While VM/370 normally accepts incoming calls from either 1050 or 2741 terminals on the same physical line, that same line cannot be used for CPT-TWX terminals. When generating the VM/370 system, ensure that the hardware configuration specified in the CP module DMKRIO matches the configuration of the Emulation Program for CPT-TWX lines; the exact configuration of 1050 and 2741 lines is not critical.

Note: The base address of the 3704/3705 (the address used to load and/or dump the control program) can never be specified for use as a telecommunications line. VM/370 treats the base address as a separate entity for use only during the load and dump operation.

Step 5. Define the Macro and Text Libraries

The macro and text libraries created from the distribution tape in Step 3 must be made available to CMS. One macro library (GEN3705) is needed for the Stage 1 generation procedure and one macro library (MAC3705) and one text library (OBJ3705) are needed for the Stage 2 generation procedure. It is easiest to define all the libraries before starting Stage 1. Use the CMS GLOBAL command:

```
global maclib gen3705 mac3705
global txtlib obj3705
```

Step 6. The Stage 1 Generation Procedure

The Stage 1 generation procedure accepts the CMS file you created in Step 4 as input and produces the Stage 2 input file that is needed in Step 7.

The Stage 1 generation procedure is performed by invoking the 3705 Assembler to process the 3704/3705 control program macro instructions. It produces one file with the same filename as the input file and with a filetype of TEXT. This TEXT file contains 3705 Assembler source statements and job control language (JCL) statements.

THE ASM3705 COMMAND

Use the CMS ASM3705 command to invoke the 3705 Assembler to assemble the macro instruction file. The 3705 Assembler processing and output are controlled by the options selected. The format of the ASM3705 command is:

```
┌──────────────────────────────────────────────────────────────────────────────┐
| ASM3705 | fn [ (options...[) ]]                                              |
|         |                                                                    |
|         |  options:                                                          |
|         |  ┌      ┐  ┌      ┐  ┌      ┐  ┌      ┐  ┌      ┐                  |
|         |  |XREF  |  |RENT  |  |DECK  |  |LOAD  |  |LIST  |                  |
|         |  |NOXREF|  |NORENT|  |NODECK|  |NOLOAD|  |NOLIST|                  |
|         |  └      ┘  └      ┘  └      ┘  └      ┘  └      ┘                  |
|         |            ┌             ┐  ┌       ┐                              |
|         |            |LINECOUN nn  |  |PRINT  |                              |
|         |            |LINECOUN 55  |  |DISK   |                              |
|         |            └             ┘  |NOPRINT|                              |
|         |                               └       ┘                              |
└──────────────────────────────────────────────────────────────────────────────┘
```

where:

fn specifies the filename of the source file to be assembled. This source file contains the 3704/3705 control program macro instructions. The file must have a filetype of ASM3705 and fixed-length, 80-character records.

Options

If duplicate or conflicting options are specified, the last one entered in the command line is the one in effect.

XREF	includes a cross-reference symbol table in the LISTING file.
NOXREF	suppresses the cross-reference symbol table.
RENT	checks the source file to see if it satisfies reentrancy requirements.
NORENT	suppresses the check for satisfaction of reentrancy requirements.
DECK	spools the output object module, fn TEXT, to the punch.
NODECK	suppresses the spooling of the output object module, fn TEXT, to the punch.
LOAD	creates a TEXT file on disk for the program that was assembled.
NOLOAD	suppresses the creation of a TEXT file on disk for the program that was assembled.
LIST	produces a LISTING file.
NOLIST	produces no LISTING file.
PRINT	spools the LISTING file to the printer.
DISK	puts the LISTING file on disk.
NOPRINT	produces no LISTING file.
LINECOUN nn	specifies the number of lines per output printer page. A default of 55 lines is assumed.

Note: All of the options of the 3705 XF Assembler are supported and may be used with the ASM3705 command, with the exception of ALIGN/NOALIGN and TEST/NOTEST.

Files Created by the ASM3705 Command

TEMPORARY WORKFILES Three files are temporarily created for each assembly:

fn SYSUT1
fn SYSUT2
fn SYSUT3

Any existing files with the same file identifiers are erased at the beginning of the assembly. These files are placed on the read/write disk with the most available space. Work space is automatically allocated as needed during the assembly and returned to available status when the assembly is complete. Insufficient space causes abnormal termination of the assembly.

PERMANENT FILES One or two permanent files may be created during a successful assembly:

fn TEXT
fn LISTING

The fn TEXT file contains the output object module if the LOAD option is in effect. The fn LISTING file contains a listing of source statements, assembled machine code, and other associated information based on the options selected. This file is created unless the NOPRINT or NOLIST options are selected. The LISTING and TEXT files are placed on (1) the disk from which the source file was read, (2) its parent or (3) the primary disk, unless you created a file definition for these files placing them on a non-DASD device. Failure to obtain sufficient space for these files results in abnormal termination of the assembly.

SPECIAL CONSIDERATIONS FOR THE STAGE 1 ASSEMBLY

The Stage 1 assembly can be very lengthy. The amount of time the Assembler takes depends upon the macro options selected and the number of users on the VM/370 system.

The LISTING file produced by the Stage 1 assembly is quite large. If you let the ASM3705 command option default to DISK, much of the space on your A-disk is used. Therefore, VM/370 recommends that you specify the PRINT option when you issue the ASM3705 command. Also, there are many macro expansions that make the LISTING file larger. VM/370 recommends that you insert a 'PRINT NOGEN' assembler statement in front of the first macro instruction in the input file to suppress the printing of the macro expansions and reduce the size of the LISTING file.

You should examine the output of the Stage 1 assembly carefully and produce a list of resource IDs, with their characteristics, for the operations personnel. The cross-reference list for operations should include:

- Resource ID
- Type of resource (line or terminal)
- Type of line (EP-mode or variable)
- Location

Step 7. The Stage 2 Generation Procedure

During the Stage 2 generation procedure the TEXT file produced in Step 6 is scanned. That TEXT file contains several job steps of 3705 Assembler source statements with embedded OS JCL statements. The JCL statements are removed and a unique CMS 3705 Assembler source file is created for each job step in the input file. An EXEC procedure is also created to assemble and link edit the source files. When the EXEC procedure is invoked, it produces the load module file (and, optionally, saves a copy of the control program in page-format on a CP-owned volume).

THE GEN3705 COMMAND

Use the CMS GEN3705 command to invoke the Stage 2 service program. Command options let you determine whether or not GEN3705 includes a command in the EXEC procedure to save a copy of the load module on disk, or if GEN3705 invokes the EXEC procedure automatically. The format of the GEN3705 command is:

```
┌──────────────────────────────────────────────────────────────────────┐
| GEN3705 | fname ftype [fmode] [ (options...[) ]]                      |
|         |                                                             |
|         | Options:                                                    |
|         | ┌      ┐  ┌       ┐                                         |
|         | |RUN   |  |SAVE   |                                         |
|         | |NORUN |  |NOSAVE |                                         |
|         | └      ┘  └       ┘                                         |
└──────────────────────────────────────────────────────────────────────┘
```

where:

fname specifies the filename of the Stage 2 input stream produced by the Stage 1 assembly. The file must contain fixed-length, 80-character records.

ftype specifies the filetype of the Stage 2 input stream. The filetype is normally TEXT.

fmode specifies the filemode.

Options:

If duplicate or conflicting options are specified, the last option entered on the command line is in effect.

RUN causes the output EXEC file to be executed at the conclusion of the GEN3705 processing.

NORUN suppresses the execution of the output EXEC file.

SAVE includes a SAVENCP command in the output EXEC file to create a page-format copy of the 3704/3705 control program on a VM/370 CP-owned volume.

If you are generating a 3705 Emulator control program with a Type 4 channel adapter, do not use the SAVE option; an error message will result from the SAVENCP command. In this case, you must specify the SAVENCP command yourself, specifying the CAMOD option.

NOSAVE does not include the SAVENCP command in the output EXEC file.

Files Created by the GEN3705 Command

Three types of permanent files are created when the GEN3705 command successfully executes: ASM3705, TEXT, and EXEC files.

```
fname00  ASM3705        fnameL0  TEXT        fname EXEC
fname01  ASM3705        fnameL0  TEXT
   .        .              .       .
   .        .              .       .
   .        .              .       .
fnamenn  ASM3705        fnameLn  TEXT
```

A separate ASM3705 file is created for each assembly job step in the Stage 2 input file. Each ASM3705 file created by GEN3705 is given a unique filename of the form 'fnamenn'. The first six characters of the input filename are concatenated with a two-digit number. For example, if the input file is NCP320 TEXT, the output files are NCP32000 ASM3705, NCP32001 ASM3705, ..., NCP320nn ASM3705. These files are used as input to the 3705 Assembler when it is invoked by the Stage 2 EXEC procedure.

The GEN3705 program creates several TEXT files. These files contain only linkage editor control statements, those statements necessary to build the load module file for the 3704/3705 control program. Each of the TEXT files created is given a unique filename of the form 'fnameLn'. The first six characters of the input filename are concatenated with the letter L and a one-digit number. For example, if the input file is NCP320 TEXT, the linkage editor output files are NCP320L0 TEXT, NCP320L1 TEXT, ..., NCP320Ln TEXT.

The filenames assigned to the linkage editor and assembler files must be different. If the filenames were the same, when the ASM3705 files are later assembled, TEXT files would be produced which would have file identifiers that conflict with the linkage editor files.

The EXEC macro file created contains the CMS commands necessary to invoke the ASM3705 command for each of the ASM3705 files, and to subsequently invoke the linkage editor for each of the Assembler TEXT files. If the SAVE option is specified, the EXEC file also contains the SAVENCP command which loads the 3704/3705 control program image into virtual storage and creates the page-format copy of it on a CP-owned volume. The filename of the Stage 2 input file is used as the 'ncpname' operand for the SAVENCP command.

SPECIAL CONSIDERATIONS FOR THE STAGE 2 GENERATION PROCEDURE

VM/370 recommends that you specify the RUN option. When the RUN option is specified, GEN3705 stacks a CMS command line to cause the EXEC file to execute following the completion of the GEN3705 program. This technique minimizes the virtual storage overhead during the EXEC file execution.

If you do not specify the SAVE option, you have to explicitly issue the SAVENCP command. If you do specify the SAVE option, be sure that the input file has the same filename as the entry reserved in the system name table. The system name table is created when a NAMENCP macro is issued during a VM/370 system generation. The NAMENCP macro is described in "Part 2. Defining Your VM/370 System" and the building of the system name table is described in "Part 3. Generating VM/370 (CP, CMS, RSCS, and IPCS)."

Step 8. Invoke the EXEC Procedure Created in Step 7

If you specified RUN on the GEN3705 command, this step is executed for you. If you did not specify RUN on the GEN3705 command, you must invoke the EXEC procedure that the GEN3705 program created.

The EXEC procedure is given the same filename as the GEN3705 input file. It is invoked by entering that filename at the terminal. For example, if the input file is NCP320 TEXT, the EXEC file is named NCP320 EXEC, and can be invoked by issuing:

```
NCP320
```

at the terminal.

This EXEC procedure contains CMS commands that:

- Assemble the 3705 source files (ASM3705 commands).
- Build the TXTLIB that the 3705 Assembler needs (TXTLIB commands).
- Define all the necessary files; such as, the SYSLIB and SYSLMOD files, load libraries, and text libraries (FILEDEF commands).
- Link edit the 3705 text files creating a load module (LKED commands).

You need not issue the ASM3705 and LKED commands that create the 3704/3705 control program load module; the EXEC procedure does that for you. The ASM3705 command is described in Step 6. The FILEDEF and TXTLIB commands are described in the _VM/370 CMS Command and Macro Reference_.

THE LKED COMMAND

Use the CMS LKED command to create the 3704/3705 control program load module from the 3705 Assembler object files. The format of the LKED command is:

```
┌──────────────────────────────────────────────────────────────────────┐
│ LKED    │ fname    [(options...[)]]                                   │
│         │                                                            │
│         │ Options:                                                   │
│         │ [NCAL] [LET] [ALIGN2] [NE] [OL] [RENT]                     │
│         │                                                            │
│         │ [REUS] [REFR] [OVLY] [XCAL]                                │
│         │                                                            │
│         │ [NAME membername] [LIBE libraryname]                       │
│         │ ┌     ┐  ┌        ┐  ┌        ┐                            │
│         │ │XREF│  │TERM    │  │PRINT   │                            │
│         │ │MAP │  │NOTERM  │  │DISK    │                            │
│         │ │LIST│  └        ┘  │NOPRINT│                            │
│         │ └     ┘              └        ┘                            │
└──────────────────────────────────────────────────────────────────────┘
```

where:

fname specifies the filename of the object file to be processed. The file must have a filetype of TEXT and fixed-length, 80-character records.

Options:

If duplicate or conflicting linkage editor options are specified, the resolution is performed by the linkage editor in accordance with its normal procedures. If duplicate or conflicting CMS-related options are specified, the last one entered on the command line is in effect. The CMS-related options are: TERM, NOTERM, PRINT, DISK, NOPRINT, NAME, and LIBE.

NCAL — suppresses the automatic library call function of the linkage editor.

LET — suppresses marking of the load module "not executable" in the event of some linkage editor error condition.

ALIGN2 — indicates that boundary alignment specified in the linkage editor input file is to be performed on the basis of 2048-byte boundaries. If this option is omitted, alignment is performed on the basis of 4096-byte boundaries.

NE — marks the load module output as "not to be edited" such that it cannot be processed again by the linkage editor.

OL — marks the load module output "only loadable".

RENT — marks the load module reenterable.

REUS — marks the load module reuseable.

REFR — marks the load module refreshable.

OVLY — processes an overlay structure.

XCAL — allows valid exclusive CALLs in the overlay structure.

NAME membername
: is the member name to be used for the load module created. The member name specified here overrides the default name, but it cannot override a name specified via the linkage editor NAME control statement.

LIBE libraryname
: is the filename of a LOADLIB file where the output load module is to be placed. The LOADLIB file specified here may also be used for auxiliary input to the linkage editor via the INCLUDE statement.

XREF — produces an external symbol cross-reference for the modules being processed.

MAP — produces only a module map for the processed module(s).

<u>LIST</u> — includes only linkage editor control messages in the printed output file.

<u>TERM</u> — displays any linkage editor diagnostic messages at the user terminal.

NOTERM — suppresses the displaying of diagnostic messages.

PRINT — spools the linkage editor printed output file to the printer.

<u>DISK</u> — stores the linkage editor output in a CMS disk file with a filetype of LKEDIT.

NOPRINT — produces no output file.

Linkage Editor Control Statements

Only a subset of the possible linkage editor control statements are meaningful in CMS. Since the CMS interface program cannot examine the input data for the LKED command, all of the control statements are allowed, even though several of them result in the creation of a load module file which cannot be used under CMS. For both command options and control statements, see the publication *OS/VS Linkage Editor and Loader*.

Files Created by the LKED Command

TEMPORARY WORKFILE The LKED command produces one temporary file:

    fname SYSUT1

This file is temporarily created for each link-edit step; any existing file with the same file identifier is erased at the beginning of the link edit. This file is placed on the read/write disk with the most available space. Work space is automatically allocated as needed during the link edit and returned to available status when the link edit is complete. Insufficient space causes abnormal termination of the link edit.

PERMANENT FILES The LKED command produces two permanent files:

    fname LOADLIB
    fname LKEDIT

The 'fname LOADLIB' file contains the load module(s) created by the linkage editor. This file is in CMS simulated partitioned data set format, as created by the CMS OS data management macros. The filename of the input file becomes the filename of the LOADLIB file, unless the LIBE option is specified. The filename of the input file also becomes the member name of the output load module, unless either the NAME option or a NAME control statement is used. One or more load modules may be created during a single LKED command execution if the NAME linkage editor control statement is used in the input file. When the NAME control statement is used, that name becomes the member name in the LOADLIB file. The replace option of the NAME statement determines whether existing members with the same name are replaced or retained.

The 'fname LKEDIT' file contains the printed output listing produced according to the XREF, MAP, or LIST options. This file is created on disk unless the PRINT or NOPRINT option is specified. The LOADLIB and LKEDIT files are placed on (1) the disk from which the input file was read, (2) the parent disk, or (3) the primary disk. Failure to obtain sufficient space for these files results in abnormal termination of the linkage editor.

Step 9. Save the 3704/3705 Control Program Image on Disk

If you specified SAVE on the GEN3705 command, this step is executed for you. If you did not specify SAVE on the GEN3705 command, you must issue the SAVENCP command yourself.

Note: The VM/370 command privilege class A, B, or C is required to use the SAVENCP command.

THE SAVENCP COMMAND

Use the CMS SAVENCP command to read a 3704/3705 control program load module created by the LKED command, and to load it into virtual storage in the CMS user area. Once the load is performed, SAVENCP scans the control program image and extracts the control information required by CP. The control information is accumulated in one or more 4096-byte pages in the CMS user area. When all of the necessary control information is extracted, SAVENCP builds the Communications Controllers Parameter List (CCPARM) and issues the DIAGNOSE X'50' instruction to create the page-format copy of the control program on a CP-owned volume. The format of the SAVENCP command is:

```
┌──────────────────────────────────────────────────────────────────────────┐
│ SAVENCP   │   fname    [  (options.. [) ]]                               │
│           │                                                              │
│           │   Options:                                                   │
│           │   ┌              ┐  ┌             ┐  ┌                   ┐   │
│           │   │ENTRY symbol│  │NAME ncpname│  │LIBE libraryname│   │
│           │   │CXFINIT      │  │fname        │  │fname             │   │
│           │   └              ┘  └             ┘  └                   ┘   │
│           │   ┌          ┐                                               │
│           │   │CAMOD {0}│                                               │
│           │   │      {1}│                                               │
│           │   └          ┘                                               │
└──────────────────────────────────────────────────────────────────────────┘
```

where:

fname is the filename of the LOADLIB file where the 3704/3705 control program load module resides; unless LIBE is specified, in which case, it specifies the member name of the image within the LOADLIB. This name is used as the ncpname for the DIAGNOSE instruction, unless the NAME option is also specified.

Options

ENTRY symbol
: is the external symbol of the entry point in the 3704/3705 control program load module. (The standard entry for the Emulation Program is CYASTART.) If the SAVE option of the GEN3705 command is specified, this symbol is set in the output EXEC file according to the Stage 2 input file.

NAME ncpname
: is the ncpname to be used when the DIAGNOSE parameter list is built. The ncpname specified must match an entry in the system name table. These entries are created with the NAMENCP macro when VM/370 is generated.

LIBE libraryname
: is the filename of a load module library file, filetype LOADLIB, which contains the control program image as member 'fname'.

CAMOD {0 | 1}
: must be specified if a Type 4 Channel Adapter is being used. VM/370 supports only one Type 4 Channel Adapter at a time, although two may be present.

CAMOD 0 corresponds to -0 following the subchannel address on the ADDRESS operand of the LINE macro in Stage 1 of the EP system generation. (0 may have been coded or defaulted on the LINE macro; you must specify it on the CAMOD option.)

CAMOD 1 corresponds to -1 following the subchannel address on the ADDRESS operand of the LINE macro in Stage 1 of the EP system generation.

EXECUTION OF THE SAVENCP PROGRAM

The DIAGNOSE X'50' instruction invokes the CP module DMKSNC to:

- Interpret the parameter list (CCPARM) built by SAVENCP.
- Check the parameter specifications against the NAMENCP macro for the 3704/3705 control program.
- Write the page-format image of the control program onto the appropriate CP-owned volume.

The parameter list for the DIAGNOSE instruction must start on a 4096-byte boundary.

When the DIAGNOSE X'50' instruction is executed, the module DMKSNC searches the DMKSNT module for a NAMENCP macro of the same ncpname as the one in the CCPARM parameter list. The values specified in the parameter list are compared to those specified in the NAMENCP macro. If any parameters conflict, an error message is displayed at the terminal. If no error conditions are detected, DMKSNC starts to transfer the control program image from CMS virtual storage to the CP-owned volume specified in the NAMENCP macro. Successful completion of this process completes the generation of a 3704/3705 control program for VM/370 use.

Step 10. Load the 3704/3705 Control Program

The 3704/3705 control program is automatically loaded each time the VM/370 system is loaded, if the CPNAME operand was specified on the RDEVICE macro when VM/370 was generated and if the 3704/3705 is online. If the CPNAME operand was not coded, you must issue the CP NETWORK LOAD command line to load a 3704/3705 control program into the 3704/3705 Communications Controllers' storage.

THE NETWORK LOAD COMMAND LINE

Use the NETWORK LOAD command to initiate the loading of an EP control program into a 3704/3705 Communications Controller. The format of the NETWORK LOAD command line is:

NETwork	LOAD raddr ncpname

where:

LOAD initiates the control program load operation.

raddr is the real address of the 3704/3705 to be loaded.

ncpname is the name, defined by a NAMENCP macro, of the 3704/3705 control program image to be loaded into the 3704/3705 specified by raddr.

EXECUTION OF THE NETWORK LOAD COMMAND

The NETWORK LOAD command accesses the control program image using the information in the system name table (DMKSNT) entry created by the NAMENCP macro. If the 3704/3705 specified in the command is not in an "IPL Required" state at the time the command is issued, the message:

DMKNET461R CTLR raddr IPL NOT REQUIRED; ENTER "YES" TO CONTINUE:

appears at the terminal. If the reply to the message is other than "yes", the command terminates without loading the 3704/3705. Otherwise, the loader bootstrap routines are written to the 3704/3705 and loading starts. VM/370 does not execute the "bring-up" test routines as a part of the load process. If these tests are to be made, they must be run from a virtual machine with the 3704/3705 dedicated.

When the load of the control program image is complete, the command processor verifies that the 3704/3705 configuration described by the control program can be serviced by the VM/370 CP control blocks in storage.

SPECIAL CONSIDERATIONS FOR LOADING THE EP 3704/3705 CONTROL PROGRAM

If a 3704/3705 Emulation Program is automatically reloaded after a 3704/3705 failure, the system may loop after the restart. The message

DMKRNH463I CTLR raddr UNIT CHECK; RESTART IN PROGRESS

and two responses

CTLR raddr DUMP COMPLETE
CTLR raddr ncpname LOAD COMPLETE

indicate that the 3704/3705 has been reloaded. If the system loops after the second response, you must reset all emulator lines from the 3704/3705 control panel.

If the automatic dump feature is not enabled, one of the messages

DMKRNH462I CTLR raddr UNIT CHECK; IPL REQUIRED

-- or --

DMKRNH464I CTLR 'raddr' CC=3; DEPRESS 370X "LOAD" BUTTON indicates a 3704/3705 abnormal termination. The 3704/3705 Emulation Program must be reloaded via the NETWORK LOAD command. If the system loops when an attempt is made to enable the lines, you must reset all emulator lines from the 3704/3705 control panel.

The _IBM 3704 and 3705 Communications Controllers Operator's Guide_ describes the procedure for resetting emulator lines from the 3704/3705 control panel.

Step 11. Logging On through the 3704/3705

Because a 3704/3705 can support emulator-mode lines and can also support a variety of terminals, the procedure for logging on is sometimes complicated. Use the following procedure to log on to VM/370.

TURN THE POWER ON

First, turn the power on for your terminal and wait 15 to 30 seconds.

CHECK FOR AN ONLINE MESSAGE

Second, look for an online message at your terminal.

If one of the following messages appears at your terminal

```
vm/370 online       xxxxxx xxxxxx

            -- or --

xxxxxx xxxxxx       vm/370 online
```

your terminal is a 2741 connected to VM/370 via a 2701/2702/2703 line or via a 3704/3705 line in emulation mode. You can proceed with the normal logon procedure for your type of terminal, as described in the _VM/370 Terminal User's Guide_.

If the message

```
vm/370 online
```

appears at your terminal, your terminal is:

- A 1050, 3101, or CPT-TWX Model 33/35 terminal connected to VM/370 in EP mode.

You can proceed with the normal logon procedure for your terminal type. This procedure is described in the _VM/370 Terminal User's Guide_.

Step 12. Applying PTFs to the 3704/3705 Load Library

If necessary, it is possible to apply Program Temporary Fixes (PTFs) directly to the 3704/3705 load library. The CMS ZAP program applies the PTF. See the _VM/370 Operator's Guide_ for information on using the ZAP service program.

Testing the 3704/3705 Control Program

After you have generated a 3704/3705 control program, loaded it, and logged on, you may want to test the 3704/3705 control program. Several CP commands are provided to control the operation, check the status, and dump the contents of the 3704/3705. The NETWORK command loads and dumps any 3704/3705 control program. The existing CP commands (ENABLE, DISABLE, QUERY, DISPLAY, VARY, and HALT) also provide support for EP 3704/3705 control programs. The NCPDUMP command formats and prints a dump of 3704/3705 storage. Use these commands to test the 3704/3705 control program.

The NETWORK, ENABLE, DISABLE, NCPDUMP, QUERY, DISPLAY, VARY, and HALT commands are described in the _VM/370 Operator's Guide_ and the _VM/370 CP Command Reference for General Users_.

Part 5. Updating VM/370

Part 5 tells you how to apply Program Temporary Fixes (PTFs) and updates to an installed VM/370 system. It contains information about the following:

- Introduction
- A Virtual Machine for Updating VM/370
- Files for System Updates
- System Program Update Tape (PUT)
- Recommended Procedures for Updating VM/370
- Building a New CP Nucleus
- Updating CMS
- Updating RSCS
- Updating IPCS Modules
- Updating Service Programs
- Updating the Loader Program
- EXEC Procedure and Command Format Summaries

Introduction

VM/370 provides you with several procedures and techniques for updating your VM/370 system. Using a virtual machine, you can perform updating and maintenance tasks concurrently with other production work. The framework provided by VM/370 gives you a maximum amount of flexibility in maintaining your system. This framework includes:

- A recommendation for a system support plan, with a userid MAINT provided with access to minidisks containing files necessary for system updating and maintenance.
- A monthly system Program Update Tape (PUT) is automatically distributed to VM/370 users. This tape contains updated TEXT and MODULE files, as well as PTFs (Program Temporary Fixes) that may be applied to your VM/370 system.
- The UPDATE command and two EXEC procedures, VMFASM and VMFMAC, which allow multilevel updating capabilities with concomitant multilevel backup.
- Naming conventions for update files and control files.
- Several EXEC procedures and programs that simplify updating VM/370. These programs are listed in Figure 33.

All of these techniques require the use of CMS; you should have a thorough understanding of the CMS file system and disk search order, the CMS EXEC processor, and the UPDATE command before you attempt to use any of the procedures described here.

The _VM/370 CMS User's Guide_ provides complete tutorial information on CMS; for reference material on CMS commands and EXEC control statements, see _VM/370 CMS Command and Macro Reference_.

Deciding Which Procedure To Use

When you have a maintenance task, you want to accomplish it as quickly as possible without excessive delay or unnecessary steps. There are two types of maintenance, and each has one basic procedure.

Text level maintenance is available with the system PUT distributed by IBM. When you use this type of maintenance, you do not have to worry about which procedures to use. The user memo always tells you what to do. Existing TEXT and TXTAP files for your VM/370 system are replaced, on a one-for-one basis, by new files contained on the system PUT.

The second type of maintenance involves more work on your part. If you have updates that you want to apply to IBM modules (for example, if you have written an accounting routine you want to include in the DMKACO module), use the following procedures:

1. If an update is being made to a macro library, use the VMFMAC EXEC to update the library.

Program	Comments
VMSERV	Updates CP, CMS, RSCS, and IPCS from the system Program Update Tape (PUT). Text decks supplied with the service tape replace existing text decks.
VMFASM	Updates a source file using IBM updates and PTFs and user updates, then assembles the updated source file.
VMFMAC	Updates macro libraries using IBM and user updates.
VMFLOAD	Creates a new CP, CMS, or RSCS nucleus based on a control file and a load list EXEC file.
GENERATE	Performs a variety of maintenance functions, including directory and service program updates. May also be used to invoke the VMFLOAD program to punch a new CP, CMS, or RSCS nucleus.
CMSGEND	Creates a new CMS command module from updated TEXT files.
ASMGEND	Updates the VM/370 system assembler.
VSAMGEN	Updates and rebuilds the CMSVSAM and CMSAMS discontiguous saved segments based on PTFs to VSAM code.
UPDATE	Applies single or multilevel updates to source programs.

Figure 33. Programs for Updating VM/370

2. Use the VMFASM EXEC procedure to reassemble the source module using update files. These may be IBM PTFs or updates or your own. If you are reassembling a module because of a MACLIB change, no update files are necessary.

3. Use the VMFLOAD program to punch a new CP, CMS, or RSCS nucleus, incorporating existing TEXT files and new ones created by the VMFASM EXEC.

4. Depending on whether you are creating a new CP, CMS, or RSCS nucleus, you may next have to perform additional steps, like writing the new nucleus onto disk, and so on.

The various procedures and steps to take are summarized in Figure 34. These procedures are described in detail in the remainder of Part 5. Before you use any of the procedures, you should have established a virtual machine userid for your maintenance tasks. You must also be acquainted with the CMS files that are used for updating and the naming conventions used by IBM. These topics are discussed next.

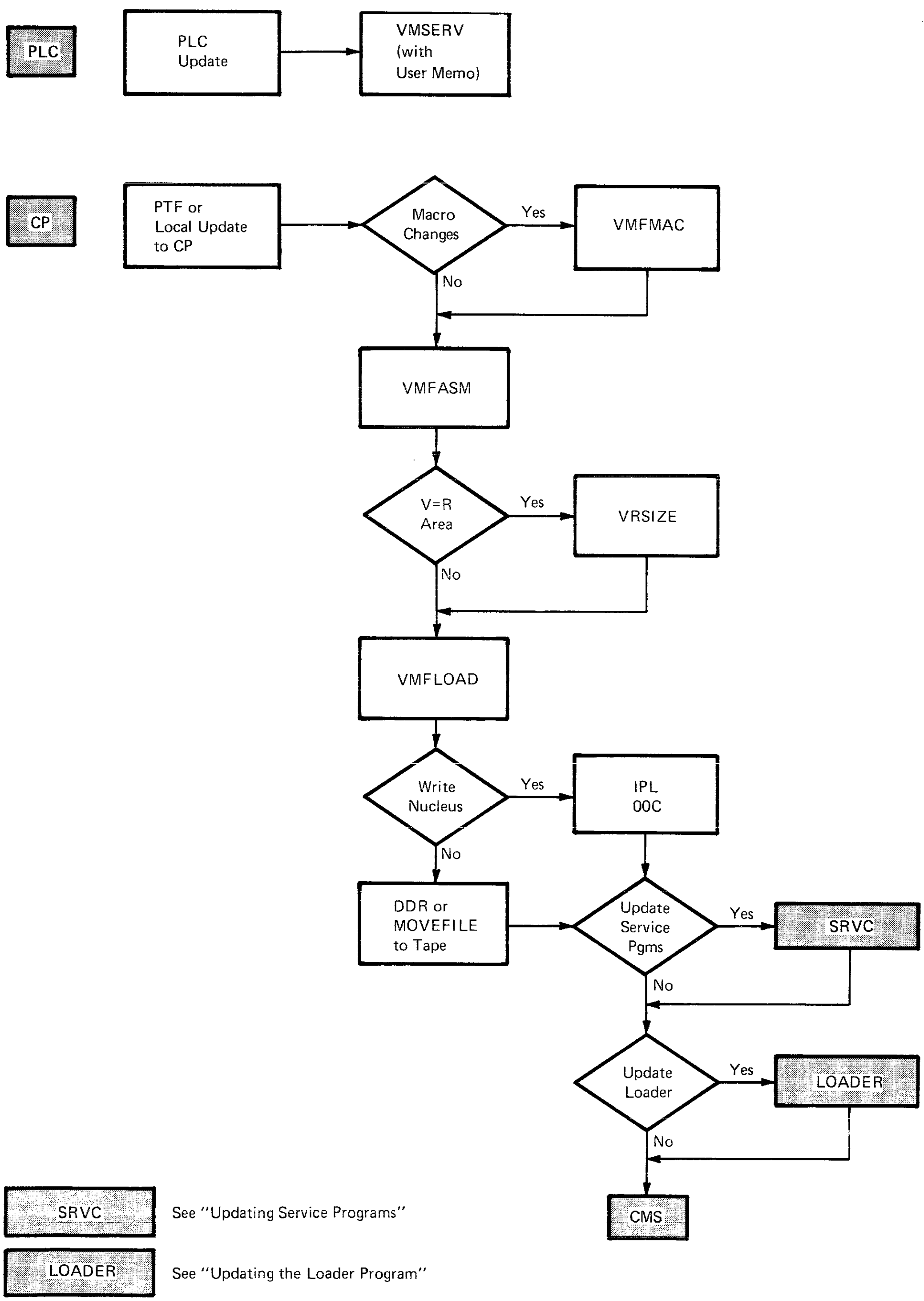


Figure 34. Deciding Which Updating Procedures To Use (Part 1 of 2)

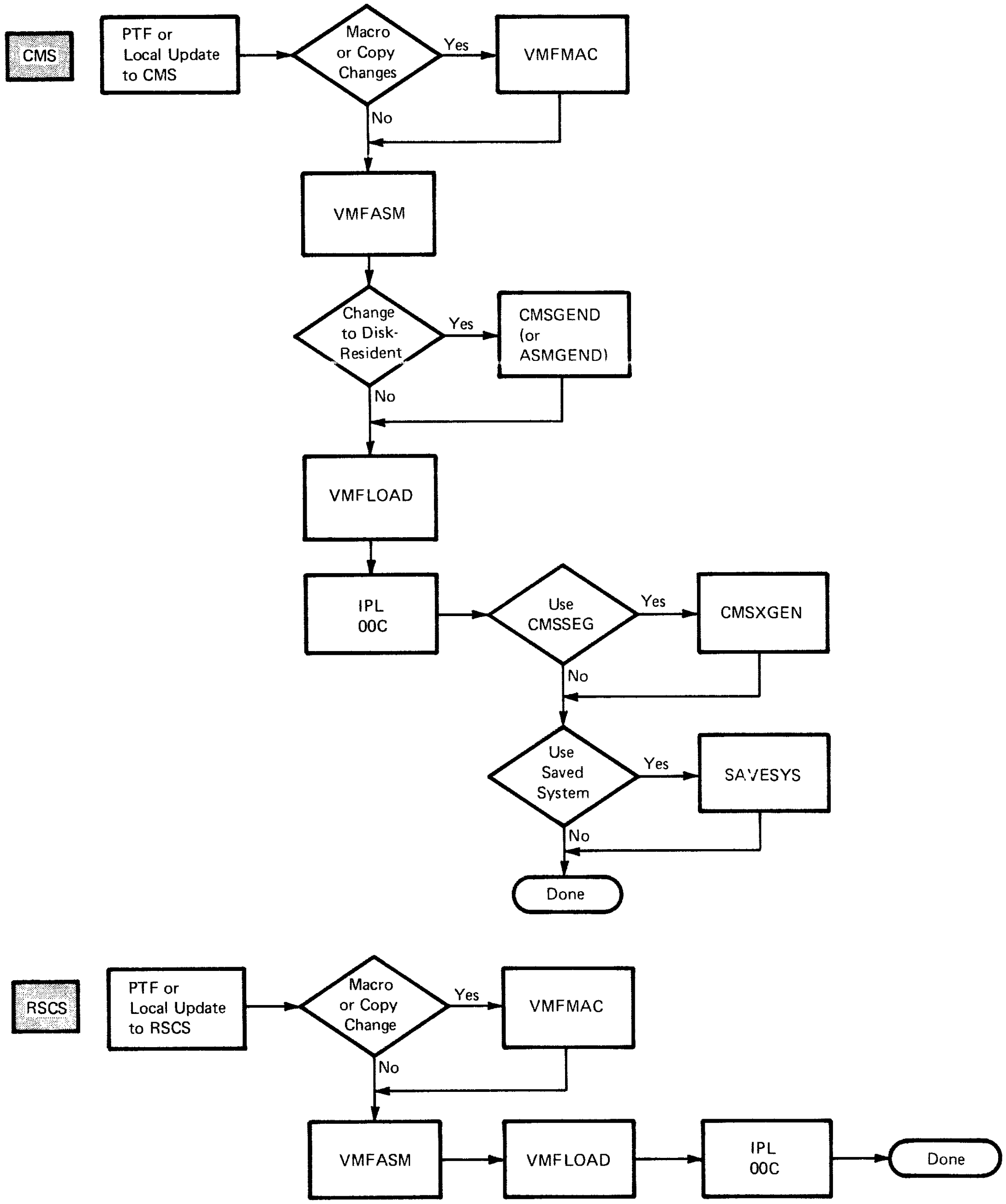


Figure 34. Deciding Which Updating Procedures To Use (Part 2 of 2)

A Virtual Machine for Updating VM/370

The VM/370 directory distributed with each starter system contains an entry for a userid, MAINT. You may want to use this userid for system updating and maintenance. MAINT's virtual machine should have access to all the disks required for system maintenance.

A suggested virtual machine configuration for updating a 2314 system is:

```
USER MAINT CPCMS 720K 16M BCEG
  ACCOUNT (installation defined)
    OPTION ECMODE REALTIMER
    CONSOLE   009   3215
    SPOOL     00C   2540   READER A
    SPOOL     00D   2540   PUNCH  A
    SPOOL     00E   1403   A
    MDISK  190 2314 035 135 CPRnLO¹ MR READ
    MDISK  191 2314 019 010 CPRnLO WR READ
    MDISK  194 2314 170 033 CPRnLO MR READ
    MDISK  199 2314 034 001 CPRnLO WR READ
    MDISK  193 2314 001 050 USERD1 MR READ
    MDISK  294 2314 051 050 USERD1 MR READ
|   MDISK  393 2314 001 125 USERD2 MR READ
    MDISK  394 2314 001 160 USERD3 MR READ
    MDISK  390 2314 101 003 USERD1 MW READ
    MDISK  cuu 2314 000 203 yyyyyy MW
```

where cuu and yyyyyy are the address and label of your system residence volume defined in the DMKSYS module.

A suggested virtual machine configuration for updating a 3330 system is:

```
USER MAINT CPCMS 720K 16M BCEG
  ACCOUNT (installation defined)
    OPTION ECMODE REALTIMER
    CONSOLE   009   3215
    SPOOL     00C   2540   READER A
    SPOOL     00D   2540   PUNCH  A
    SPOOL     00E   1403   A
    MDISK  190 3330 030 085 CPRnLO MR READ
    MDISK  191 3330 016 007 CPRnLO WR READ
    MDISK  194 3330 115 027 CPRnLO MR READ
    MDISK  199 3330 029 001 CPRnLO WR READ
    MDISK  193 3330 001 030 USERD1 MR READ
    MDISK  294 3330 031 030 USERD1 MR READ
|   MDISK  393 3330 061 070 USERD1 MR READ
    MDISK  394 3330 141 090 USERD1 MR READ
    MDISK  390 3330 231 002 USERD1 MW READ
    MDISK  cuu 3330 000 404 yyyyyy MW
```

where cuu and yyyyyy are the address and label of your system residence volume defined in your DMKSYS module.

----

[1] CPRnLO may be CPR4LO, CPR5LO, CPR6LO and so forth, depending on the release level.

A Virtual Machine for Updating VM/370

A suggested virtual machine configuration for updating a 3340 system is:

```
USER MAINT CPCMS 720K 16M BCEG
  ACCOUNT (installation defined)
    OPTION ECMODE REALTIMER
    CONSOLE  009   3215
    SPOOL    00C   2540   READER A
    SPOOL    00D   2540   PUNCH  A
    SPOOL    00E   1403   A
    MDISK 190 3340 048 240 CPRnLO MR READ
    MDISK 191 3340 026 015 CPRnLO WR READ
    MDISK 194 3340 288 060 CPRnLO MR READ
    MDISK 199 3340 046 002 CPRnLO WR READ
    MDISK 193 3340 001 075 USERD1 MR READ
    MDISK 294 3340 076 075 USERD1 MR READ
|   MDISK 393 3340 151 185 USERD1 MR READ
|   MDISK 394 3340 001 250 USERD2 MR READ
    MDISK 390 3340 221 003 USERD2 MW READ
    MDISK cuu 3340 000 348 yyyyyy MW
```

where cuu and yyyyyy are the device address and disk label of your system residence volume defined in the DMKSYS module.

The entries in the preceding VM/370 directory, with the exception of the 193, 294, 393, 394, and 390 virtual disks, are included in the 2314, 3330, and 3340 VM/370 directories supplied with the starter system, and should be included in your VM/370 directory, as they are used by IBM for support.

The contents of the preceding virtual disks are:

Disk	Contents
190	Current CMS system disk
191	Work area
194	CP, RSCS, and IPCS text retention
199	CPGEN's 191 minidisk (work area)
193	CMS PTFs, updates, and updated text decks (object modules)
294	CP, RSCS, and IPCS PTFs, updates, and updated text decks (object modules)
393	CMS source and macros
394	CP, RSCS, and IPCS source, macros, and copy files
390	CMS test nucleus area
cuu	CP system residence device, or a replica of it, for test purposes

These virtual disks are shown in Figure 35.

Source code for the CMS system is included on the CMS2 tape. Source code for CP is included on the CP2 tape. Source code for RSCS and IPCS is included on the RSCS/IPCS tape. These tapes are distributed by the Program Information Department (PID).

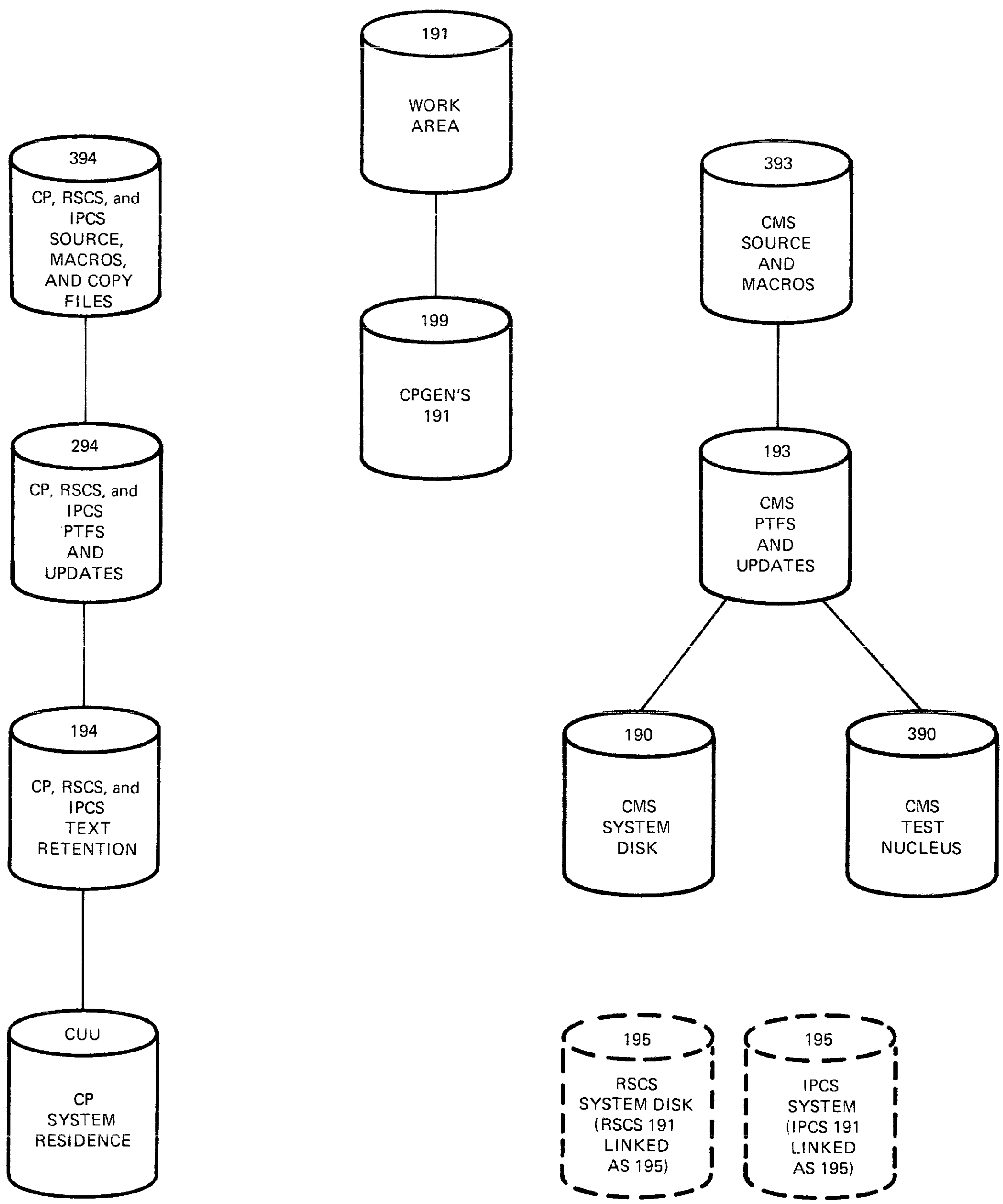


Figure 35. System Support Plan

Accessing Disks

When you are using the VM/370 procedures to apply updates to system modules, update file identifiers may be duplicated on more than one disk; there may be updates that are located in several different places. You should always be sure that you have the correct disks accessed, and that you have accessed them with an appropriate search order.

You may find it convenient to create EXEC procedures that perform the links and accesses necessary to perform a particular update. For example to update a CP module, from files located on MAINT's 191 and 294, your EXEC procedure might look like the following:

```
ACCESS 191 A
ACCESS 294 B/A
ACCESS 394 C/A
```

This search order ensures that if a control file or auxiliary control file with the same filename exists on both the 191 and 294, the one on the 191 is used.

When updates are made to RSCS the 191 disks provided for these virtual machines can be linked to and accessed in read/write status, for example

```
link rscs 191 195 w wpass
access 195 a
```

Files for System Updates

Each of the components of VM/370 has a unique character module identifier, which is used to name the component's modules. These component identifiers are also used to name the files used to update the components. The identifiers are:

Component	Module Identifier
CP	DMK
CMS	DMS
RSCS	DMT
IPCS	DMM

The default CMS filetypes are used to identify the source, object code, module files and libraries associated with each component. These filetypes are:

Filetypes	Type of File
ASSEMBLE	Source File
TEXT	Object deck (relocatable)
TXTAP	Object deck with Attached Processor Support (relocatable)
MODULE	Nonrelocatable object code
MACLIB	Macro or copy library

Two of the update procedures, VMFMAC and VMFASM, use the CMS UPDATE command to update macro libraries and source files. Since the updates that are applied are multilevel updates, there are control files (with a filetype of CNTRL) and auxiliary control files, (with filetypes of AUXxxxxx) as well as the actual update files (consisting of UPDATE control statements and new source records). These files may have the following generic filetypes:

Filetype	File Contents
CNTRL	Control file
AUXxxxxx	Auxiliary control file
UPDTxxxx	Local update (listed in a CNTRL file)
anything	Local update (listed in an AUX file)

IBM uses file identifiers as listed below for distributed updates to VM/370.

DMKRn0[1] CNTRL: is used for CP source, copy, and macro updates. Its contents are:

```
TEXT MACS DMKMAC CMSLIB OSMACRO
TEXT AUXRn0
```

DMKRnA CNTRL: is used for CP source, copy, and macro updates with support for the Attached Processor. Its contents are:

```
TEXT MACS DMKAMAC DMKMAC CMSLIB OSMACRO
AP   UPDTAP
AP   AUXRn0
```

---

[1] n may be 1, 2, 3 and so forth depending on the release level.

DMSRn0 CNTRL: is used for CMS source updates. Its contents are:

```
TEXT MACS CMSLIB OSMACRO
TEXT AUXRn0
```

DMSMn0 CNTRL: is used for CMS copy and macro updates. Its contents are:

```
TEXT MACS
TEXT AUXMn0
```

DMTRn0 CNTRL: is used for RSCS source, copy, and macro updates. Its contents are:

```
TEXT MACS DMTLOC DMTMAC
TEXT AUXRn0
```

DMMRn0 CNTRL: is used for IPCS source, copy, and macro updates. Its contents are:

```
TEXT MACS CMSLIB OSMACRO DMMMAC DMKMAC
TEXT AUXRn0
```

NCPRn0 CNTRL: is used for assembling the NCPDUMP source. Its contents are:

```
TEXT MACS OSMACRO DMKMAC CMSLIB
TEXT AUXRn0
```

All auxiliary control files distributed by IBM have the filetype AUXRN0 (or AUXMn0 for CMS MACLIB changes). When an update is issued for a module, an auxiliary control file is also distributed. For example, if an update is sent for DMKCFM then the file DMKCFM AUXRn0 is also distributed. This file, DMKCFM AUXRn0, lists the updates to be applied to the CP module DMKCFM.

All of the the update files distributed by VM/370 are assigned filetypes as follows:

$\left\{\begin{matrix} Z \\ M \\ R \end{matrix}\right\}$ nnnnnxx

where:

Z indicates a Release 5 update.

M indicates a CMS macro update.

R indicates a Release 6 update.

nnnnn is an APAR or PTF number.

xx is the 2-character component identifier (DK, DS, DT, or DM).

For example, the code and updates to answer APAR VM12765 against the Release 6 level of CP module DMKCFM are contained in the file DMKCFM R12765DK. The file DMKCFM AUXRn0 contains the entry:

```
R12765DK - COMMENT DESCRIBING FIX
```

When you create files for local updates of VM/370 modules, you should create a local control file, consisting of the appropriate VM/370 CNTRL file with an entry for your local MACLIB and AUX file. For example, the file CPLCL CNTRL may contain:

```
TEXT MACS LCLLIB DMKMAC CMSLIB OSMACRO
LCL AUXLCL
TEXT AUXRn0
```

The AUXRn0 control file should be last in the control file, so that the IBM updates are applied first. (Remember that the UPDATE command, when applying multilevel updates, reads from the bottom of the control file.)

Text files must have a filetype of TEXT. For example, after you have updated an object module using VMFASM, the most recent object file has a filetype such as TXTLOCAL. To use that text file here, you must rename it to a filetype of TEXT. If there is currently a text file on the system disk, you may want to rename it too, so that your updated text file (which may reside on another disk) is the one that is loaded.

System Program Update Tape (PUT)

IBM regularly distributes a system Program Update Tape (PUT) containing updates to VM/370. The system PUT updates are cumulative, and contain:

- Updated text decks, for a one-to-one replacement of existing text decks on MAINT's 194 disk.
- Update files with UPDATE control statements, and auxiliary control files to control the application of these updates to the source files on MAINT's 394 disk. The PTF's and AUX files may be loaded on MAINT's 294, if you want to apply the updates in conjunction with local updates.

The second file on the system PUT contains the Memo to Users, which describe in detail all of the updates that are currently available. The Memo to Users also provides step-by-step instructions on how to apply updates. These files can be loaded on your A-disk and printed, using the VMSERV EXEC.

Program level change service updates involve the use of the VMSERV EXEC, which is the same procedure that you used to install VM/370 during system generation. The VMSERV EXEC is always provided in the first tape file of the system PUT, along with the VMFPLC2 module, which is used to read the tape.

If you have no local updates to apply to any VM/370 modules, you should follow the instructions in the Memo to Users; this memo always contains up-to-date documentation on how to use the VMSERV EXEC. The Memo to Users also tells you when you must perform additional steps before invoking VMSERV. For example, if a macro library has been updated and your system definition files (DMKRIO, DMKSYS, and so on) must be reassembled, the user memo tells you to use the VMFASM EXEC to reassemble the source files.

The VMSERV EXEC builds a new CP, CMS, or RSCS nucleus from the replacement text decks on the system PUT. If you have local updates to some system modules, you may not want to use VMSERV.

For example, if you have written a local accounting routine and assembled it into the module DMKACO, and the Memo to Users indicates a PTF is to be applied to DMKACO, you may want to reassemble the source module to create a text deck that contains your modification, as well as the PTF. In this case, you have to load the DMKACO AUXRn0 file and the PTF (DMKACO RnnnnnDK) file from the system PUT. The Memo to Users indicates the location of these files on the tape; remember to use the VMFPLC2 module to load the files, rather than the CMS TAPE command. VMFPLC2 uses a blocking factor of 50:1 thereby better utilizing the system PUT and insuring maintenance will be contained on one tape.

The procedures that you use next are the same procedures you would use to apply a local update without the system PUT: you would use VMFASM to assemble the source files for all modules you wish to update, and VMFLOAD to punch a new CP nucleus. Before you use VMFLOAD, however, you want to make sure that you have loaded, from the system PUT, the updated text files for those modules you are not reassembling.

The procedures for applying updates to VM/370 are described next.

Recommended Procedures for Updating VM/370

The procedures that you can use to apply local updates are similar for CP, CMS, RSCS, and IPCS. The examples in the following pages use CP modules and control files to illustrate the use of:

- The VMFASM EXEC Procedure
- The VMFMAC EXEC Procedure
- The VMFLOAD Program

You should keep in mind that the procedures for updating source files and macro libraries are the same for all VM/370 components, and that the procedure for punching a new CMS or RSCS nucleus is basically the same as the procedure for punching a CP nucleus.

For specific details and special considerations for loading and testing a new CP, CMS, or RSCS nucleus, or for generating new IPCS modules, see:

- "Building a New CP Nucleus"
- "Updating CMS"
- "Updating RSCS"
- "Updating IPCS Modules"

The minidisk areas used in the examples in all of these discussions use the MAINT virtual machine described under "A Virtual Machine For Updating VM/370" and illustrated in Figure 35. Note that the virtual machine configuration consists of the MAINT entry in the IBM-supplied VM/370 directory, with the addition of MDISK statements for virtual disks (193, 294, 393, 394, and 390). Figure 35 shows the virtual disks described by the resultant MAINT entry. This virtual machine configuration should provide you with all the areas you need to update and test VM/370.

VM/370 Integrity

In order to preserve the integrity of VM/370 source and text files, you should keep updates and PTFs on a separate minidisk (not on the same disk as the original source and text files). This minidisk (usually | MAINT's 294) should contain the required IBM PTF updates from the latest system PUT, updates that you make (such as expanding the accounting routines or adding a command to CP), and the resultant text files containing the updates.

You also need access to the current CP text files and macro | libraries. This is MAINT's 194. This is the disk used by VMSERV when | it loads replacement text files from the System PUT.

The assembler language source files are on the 394 minidisk. You should not change these files, unless directed to do so by the Memo to Users. When you use the CMS UPDATE command and the VMFASM and/or VMFMAC EXEC procedures with the suggested virtual machine configuration shown in Figure 35 and the access search order shown in the following examples, modified files are written onto your A-disk. Also, you should not change the IBM-supplied auxiliary files nor the PTF | (XnnnnDMK) files as these are controlled by the PUT procedure.

If you want to update a VM/370 component, you should create your own control file. This file should contain entries for your own updates as well as for the IBM-supplied updates.

Control File Preparation

Control files are used by the CMS UPDATE command. Both the VMFMAC and VMFASM update procedures invoke UPDATE with the CTL option to modify source files. For VMFMAC and VMFASM, the control file must have a filetype of CNTRL. In addition, the VMFLOAD program also uses a control file: this is usually the same control file used by the VMFASM EXEC.

For an understanding of how the update procedures work, you should have a thorough understanding of the elements in a control file. Control files are described extensively in the _VM/370 CMS User's Guide_ and the _VM/370 CMS Command and Macro Reference_. The following discussion summarizes how VMFMAC, VMFASM, and VMFLOAD use the control file.

```
¹*THIS IS A SAMPLE CNTRL FILE FOR LOCAL CP UPDATES

 TEXT MACS² LOCALIB DMKMAC CMSLIB OSMACRO

 UP³ UPDTFIX1⁴

 PTF⁵ FIXTEST

 LCL AUXLCL⁶

 TEXT AUXRn0⁷
```

Notes

[1]This is a comment record.

[2]VMFASM uses the library list from the MACS record to issue a GLOBAL command before assembling the updated source file. The libraries are searched in the order specified. DMKAMAC should precede DMKMAC if AP support is required.

[3]VMFASM and VMFLOAD use the update level identifier to identify the text deck. VMFASM uses the update level identifier of the most recent update that was found and applied to name the text deck produced by the assembly. VMFLOAD uses update level identifiers to locate text decks when punching a new CP, CMS, or RSCS nucleus.

The update level identifier on the MACS record is used by VMFASM to name an assembled update text deck when no update files are found; it is also used by VMFLOAD when it fails to locate a text file based on update level identifiers associated with update files or auxiliary control files.

[4]The characters UPDT identify the filetype of a single update file, UPDTFIX1 in this example. (The characters "UPDT" maybe omitted.)

[5]The characters PTF in the update level identifier field identify this file as a PTF file. FIXTEST is the filetype of the update file.

[6]The characters AUX identify an auxiliary control file that lists additional updates to be applied, local modifications in this example.

[7]AUXRn0 is the VM/370 auxiliary control file, listing updates distributed by IBM. This file is listed at the bottom of the control file so that these updates are applied first.

A control file can have any number of update identification (UPDTxxxx) records, AUX file identification (AUXxxxxx) records, and comments, but can have only one MACS record.

Example of a CP Update

Let's assume that you want to update CP, and then load a new CP nucleus. The updates you are going to make consist of the following:

1. You want to add a command to CP. It has already been assembled into the file DMKCMD TEXT. The CP module DMKCFC must be updated to recognize the new command name, so you have updates to apply to DMKCFC.

2. You have a local update to apply to the CP module DMKSCN.

3. You want to change two members of DMKMAC MACLIB; you have updates to apply to ACCTON COPY (for accounting routines) and to RDEVICE MACRO. Since the ACCTON COPY is modified, you have to reassemble DMKACO; changes to the RDEVICE macro require you to reassemble DMKRIO.

The procedures that you would use to perform these updates are described next. Remember that the same procedures can be used when you apply updates to any of the VM/370 components.

Using VMFASM To Update Source Files

If you are going to update a VM/370 module, you should always use the VMFASM EXEC procedure, since it allows you to incorporate IBM-supplied updates with your own.

The files used in the following example are shown in Figure 36. In addition to the 194, 294, and 191 minidisks, you should also have access to the CP assembler language source files on MAINT 394, and the CMS system disk. The search order is:

```
191 A     R/W
294 B/A   R/O
194 C/A   R/O
394 D/A   R/O
190 S     R/O
```

This search order ensures that when the command

```
vmfasm dmkcfc yourown
```

is issued, the DMKCFC AUXLCL file from the 191 is used, not the copy on the 294 disk. (The copy on the 191 contains an additional entry for the second local update file, DMKCFC LOCAL02).

The VMFASM EXEC prodecure invokes the UPDATE command with the CTL, STK, and PRINT options. In this example, UPDATE uses the file YOUROWN CNTRL to determine the order in which to apply the updates. Since the IBM auxiliary control file is the last item in YOUROWN CNTRL, updates named in the file DMKCFC AUXRn0 are applied first; then the entries named in DMKCFC AUXLCL A are applied. Because no file named DMKCFC UPDTLCL exists, no update is applied for that entry in the control file.

```
File: YOUROWN CNTRL B                  File: DMKRn0 CNTRL B

TEXT MACS DMKMAC CMSLIB OSMACRO        TEXT MACS DMKMAC CMSLIB OSMACRO
LOC2 UPDTLCL                           TEXT AUXRn0
LCL AUXLCL
TEXT AUXRn0[1]

File: DMKCFC AUXLCL A                  File: DMKCFC AUXRn0 B

LOCAL02                                R12576DK
LOCAL01

File: DMKCFC AUXLCL B                  Update Files

LOCAL01                                DMKCFC R12576DK B
                                       DMKCFC LOCAL01 B
                                       DMKCFC LOCAL02 A
                                       DMKSCN UPDTLCL A

Output Files
DMKCFC TXTLCL A contains updates R12576DK, LOCAL01 , and LOCAL02
DMKCFC TXTLCL B contains updates R12576DK and LOCAL01
DMKSCN TXTLOC2 A contains update DMKSCN UPDTLCL


    294 (B-disk)                    191 (A-disk)

 ┌──────────────────┐            ┌──────────────────┐
 │ DMKRn0   CNTRL   │            │ DMKCFC AUXLCL    │
 │ YOUROWN CNTRL    │            │ DMKCFC LOCAL02   │
 │ DMKCFC   AUXRn0  │            │ DMKCFC TXTLCL    │
 │ DMKCFC   R12576DK│            │ DMKSCN UPDTLCL   │
 │ DMKCFC   AUXLCL  │            │ DMKSCN TXTLOC2   │
 │ DMKCFC   TXTLCL  │            │                  │
 │ DMKCFC   LOCAL01 │            │                  │
 └──────────────────┘            └──────────────────┘


    194 (C-disk)                    394 (D-disk)

 ┌──────────────────┐            ┌──────────────────┐
 │  .               │            │ DMKCFC  ASSEMBLE │
 │  .               │            │  .               │
 │ DMKCFC   TEXT    │            │  .               │
 │  .               │            │  .               │
 │ DMKSCN   TEXT    │            │ DMKCFC  ASSEMBLE │
 │  .               │            │                  │
 │ DMKMAC   MACLIB  │            │                  │
 └──────────────────┘            └──────────────────┘
```

Figure 36. Files for VMFASM

---

[1]AUXRn0 may be AUXR40, AUXR50, AUXR60 and so forth, depending on the release level.

When all the updates have been applied, VMFASM calls the assembler to assemble the updated source file, which has a temporary name of $DMKCFC. When the assembly is complete, VMFASM uses the update level identifier of the most recent update that was found and applied by the UPDATE command to rename the text file produced by the assembly: in this example, the output file is named DMKCFC TXTLCL.

The updated source file created by the UPDATE command is erased.

The UPDATES file produced by a multilevel update is concatenated into the output text deck so that when this object code is loaded, information pertaining to its creation is contained in the load map.

Next, issue the VMFASM command to assemble DMKSCN ASSEMBLE:

```
vmfasm dmkscn yourown
```

The UPDATE command searches for files named DMKSCN AUXRn0 and DMKSCN AUXLCL. Neither of these files exists; however, DMKSCN UPDTLCL (the local update you created) does exist. This update is applied, the source file is reassembled, and the output file is named DMKSCN TXTLOC2.

The updated source file created by the UPDATE command is erased. The UPDATES file produced by a multilevel update is concatenated into the output text deck, so that when this object code is loaded, information pertaining to its creation is contained in the load map.

Note: VMFASM creates (or replaces, if it already exists) a temporary workfile with a fileid of 'assemble-filename control-filename A1'. This file is erased when VMFASM is finished with it. If the user already has a file with this name it should be renamed using the CMS RENAME command, to prevent its loss. For example, if you enter

```
vmfasm dmkrcf yourown
```

the work file is named DMKRCF YOUROWN A1.

| Note: If the object modules created have a filetype of "TXTxxxx" and are to be used with one of the GEN EXECs (CMSGEND, DOSGEN, VSAMGEN, etc.), they must be renamed with a filetype of "TEXT".

Using VMFMAC To Update Macro Libraries

The VMFMAC EXEC procedure is similar to the VMFASM EXEC procedure, except that it is specifically designed to update macro libraries. You must provide:

- Update files, with UPDATE control statements to modify the macro library members. You must also have available any IBM PTFs that have been distributed for the macro library.
- A control file that lists update files or auxiliary control files to be updated.
- An EXEC file listing the names of the members to be included in the macro library.

The files to be used for updating RDEVICE and ACCTON COPY are shown in Figure 37. In addition to these disks, you should have access to the source COPY and MACRO files on MAINT's 394 and the CMS system disk. The search order should be:

```
194 A     R/W
191 B/A   R/O
294 C/A   R/O
394 D/A   R/O
190 S     R/O
```

YOUROWN CNTRL

```
TEXT MACS DMKMAC CMSLIB OSMACRO
LOC2  UPDTLCL
LCL AUXLCL
TEXT  AUXRn0
```

DMKMAC EXEC

```
   &1 &2 ABEND     MACRO
   &1 &2 ACCTOFF   COPY
   &1 &2 ACCTON    COPY
         .
         .
         .
   &1 &2 RDEVICE   MACRO
```

RDEVICE AUXLCL

```
FIXPTF
```

ACCTON AUXRn0

```
R12567DK
R12263DK
```

RDEVICE AUXRn0

```
R12024DK
```

Update Files

```
ACCTON    R12576DK
ACCTON    R12263DK
ACCTON    UPDTLCL
RDEVICE   FIXPTF
```

194 (A-disk)

```
 .
 .
 .
DMKMAC MACLIB
DMKMAC COPY
 .
 .
 .
```

191 (B-disk)

```
DMKMAC   EXEC
RDEVICE  AUXLCL
ACCTON   UPDTLCL
```

294 (C-disk)

```
YOUROWN CNTRL
DMKRn0  CNTRL
ACCTON  AUXRn0
ACCTON  R12576DK
ACCTON  R12263DK
RDEVICE AUXRn0
RDEVICE R12024DK
RDEVICE FIXPTF
```

394 (D-disk)

```
 .
ACCTON   COPY
ACCTOFF  COPY
ACCTON   UPDTLCL
```

Figure 37. Files for VMFMAC

You must have the 194 in read/write status because VMFMAC renames the existing MACLIB and writes a new one.

When you issue the command:

```
vmfmac dmkmac yourown
```

the VMFMAC EXEC procedure uses the DMKMAC EXEC to rebuild DMKMAC MACLIB. VMFMAC calls the UPDATE command to update each of the macro and copy files named in the EXEC.

In this example, ACCTON COPY is updated with YOUROWN CNTRL as follows:

1. The IBM updates named in ACCTON AUXRn0, R12263DK and R12576DK, are applied, in that order.
2. Since no ACCTON AUXLCL file exists, the next entry in the control file results in no update.
3. The update file ACCTON UPDTLCL is applied.

For each entry in DMKMAC EXEC, VMFMAC checks to see if there are any updates; if not, then the existing MACRO or COPY file is included in the new MACLIB without any changes.

When the entry for RDEVICE is reached, RDEVICE MACRO is updated with YOUROWN CNTRL as follows:

1. The IBM update named in RDEVICE AUXRn0, R12024DK, is applied.
2. The update named in RDEVICE AUXLCL, RDEVICE FIXPTF, is applied.
3. Since no RDEVICE UPDTLCL file exists, the last entry in the control file results in no update being applied.

After all the entries in the list DMKMAC EXEC are processed, VMFMAC erases the existing DMKMAC MACLIB and creates a new DMKMAC MACLIB with the updated members. An additional file, DMKMAC COPY, is produced; this file contains a record of the updates that were applied. DMKMAC COPY is also added to DMKMAC MACLIB, to provide you with a record of changes.

Now, since macro and copy changes affect CP modules, you must reassemble DMKACO and DMKRIO using the new DMKMAC MACLIB. If you have no local updates for these assembler source files, you can use the DMKRn0 CNTRL file to update them:

```
VMFASM          DMKACO          DMKRn0 (or DMKRnA)
VMFASM          DMKRIO          DMKRn0
```

You must be sure that all the current PTFs and auxiliary control files are available on MAINT's 294.

The text decks produced by these assemblies are not uniquely named, since the update level identifier in DMKRn0 is always TEXT. However, the update log produced by VMFASM does indicate the macro libraries used in the assembly, so you have a record of update activity.

VARIATIONS: If you do not want to use VMFMAC to update all of DMKMAC MACLIB (it is very large, and VMFMAC is not practical if you are updating only one or two members), you may want to consider manually updating the macro and copy files using the UPDATE command and then using the MACLIB REP command to update DMKMAC MACLIB. Or, you may want to use VMFMAC to create a local macro library containing your changes, and use this library, in addition to DMKMAC MACLIB, when you reassemble CP modules.

Consider the files:

LCLMAC EXEC

```
goto label25  RDEVICE MACRO
goto label25  ACCTON COPY
```

YOUROWN CNTRL

```
TEXT MACS LCLMAC DMKMAC CMSLIB OSMACRO
LOC2 UPDTLCL
LCL  AUXLCL
TEXT AUXRn0
```

When you issue the command:

```
vmfmac lclmac yourown
```

the macro library LCLMAC MACLIB is created, containing only the members RDEVICE and ACCTON. When you use YOUROWN CNTRL with the VMFASM EXEC procedure, LCLMAC MACLIB is searched before DMKMAC MACLIB for the assembly, so your macros are found first.

Using VMFLOAD To Punch a New Nucleus

After you have reassembled all the modules that require updating, you may build a new CP nucleus that contains the updated text decks. In our example, you also want to include your new module, DMKCMD, in the CP nucleus.

To punch a new nucleus, you use the VMFLOAD program, which requires:

- A loadlist file, which must have a filetype of EXEC. It contains the filenames of the object modules in the order in which they are to reside in the nucleus.
- A control file, from which VMFLOAD can determine the filetypes of the latest level text decks, so it can punch them.

The files to be used for creating a new CP nucleus are shown in Figure 38. This nucleus incorporates the updates described in the preceding pages. The search order is:

```
191  A    R/W
294  B/A  R/O
194  C/A  R/O
190  S    R/O
```

```
YOUROWN CNTRL                          DMKR30 CNTRL

TEXT MACS DMKMAC CMSLIB OSMACRO        TEXT MACS DMKMAC CMSLIB OSMACRO
LOC2 UPDTLCL                           TEXT AUXRn0
LCL  AUXLCL
TEXT AUXRn0

YOURLOAD EXEC                          CPLOAD EXEC

 &CONTROL OFF                          &CONTROL OFF
&1 &2 &3 DMKLD00E LOADER               &1 &2 &3 DMKLD00E LOADER
&1 &2 &3 DMKPSA                        &1 &2 &3 DMKPSA
&1 &2 &3 DMKMCH                        &1 &2 &3 DMKMCH
      .                                      .
      .                                      .
      .                                      .
&1 &2 &3 DMKCFC                        &1 &2 &3 DMKCFC
&1 &2 &3 DMKACO                        &1 &2 &3 DMKACO
&1 &2 &3 DMKRIO                        &1 &2 &3 DMKRIO
&1 &2 &3 DMKCMD                              .
      .                                      .
      .                                      .
      .                                &1 &2 &3 LDT DMKSAVNC
&1 &2 &3 LDT DMKSAVNC



        194 (C-disk)         294 (B-disk)          191 (A-disk)

 DMKLD00E LOADER       YOUROWN CNTRL          YOURLOAD EXEC
 DMKPSA   TEXT         DMKRn0  CNTRL          DMKCFC   TXTLCL
  .                    DMKCFC  TXTLCL         DMKACO   TEXT
  .                    DMKCMD  TEXT           DMKRIO   TEXT
  .                                           DMKSCN   TXTLOC2
 DMKCFC   TEXT
 DMKACO   TEXT
 DMKRIO   TEXT
 DMKSCN   TEXT
  .
  .
 CPLOAD   EXEC
```

Figure 38. Files for VMFLOAD

Since the VMFLOAD program uses your virtual card reader and virtual punch, you must be sure there are no files in either of these devices before you begin. You can issue the commands:

```
close punch
purge punch all
close reader
purge reader all
```

and you must be sure to spool your virtual punch to your own card reader:

```
spool punch *
```

When you issue the command

```
vmfload yourload yourown
```

VMFLOAD uses YOURLOAD EXEC to determine which files to punch. In our example, YOURLOAD EXEC is identical to the distributed CPLOAD EXEC file, except that you have added an entry for your module DMKCMD.

VMFLOAD uses the loadlist to establish the filenames of modules to be punched, and it punches them in the order they appear in the loadlist. Thus, DMKLD00E LOADER is punched first. If a filename and a filetype are specified in the loadlist, VMFLOAD punches the file.

When a filetype is not specified (as is usually the case), VMFLOAD uses the update level identifier field in the control file to determine the filetype. Since control files are structured so that the most recent update is named at the top of the file, VMFLOAD begins reading at the top of the file.

Since the next entry in the loadlist, DMKPSA, does not provide a filetype, VMFLOAD looks at the control file. In our example, since the update level identifier for the first update record is LOC2, VMFLOAD searches for the file DMKPSA TXTLOC2. Since this file does not exist, VMFLOAD looks at the next lowest identifier: LCL. It searches for DMKPSA TXTLCL. Since this file does not exist, it reads the next lowest identifier, TEXT. DMKPSA TEXT exists on the 194, so it is punched. Then VMFLOAD returns to the loadlist EXEC and repeats the same procedure for the next entry.

You can see that when VMFLOAD reaches the entry for DMKCFC in the loadlist, it locates the file DMKCFC TXTLCL, the DMKCFC module that contains your updates. Notice that although there are copies of DMKCFC TXTLCL on both the A-disk and the B-disk, VMFLOAD punches the one on the A-disk, since it uses the standard CMS order of search.

The loading process continues in this way until the end of the loadlist EXEC file. When all of the modules have been punched, you receive the messages

```
SYSTEM LOAD DECK COMPLETE
PUN FILE 0821 TO MAINT COPY 01 NOHOLD
```

These messages indicate that a copy of the new CP nucleus is in your card reader. This CP nucleus contains all the text decks on the 194 disk, except that the files:

```
DMKCFC TXTLCL
DMKSCN TXTLOC2
```

have been punched instead of their TEXT counterparts; the files

```
DMKACO TEXT A
DMKRIO TEXT A
```

have been punched instead of their counterparts on the C-disk; and your new command module, DMKCMD, is included.

Once the new nucleus has been punched into your card reader, you can load it and test it. Considerations for loading and testing each of the VM/370 components are discussed separately in the following pages.

Building a New CP Nucleus

If you are going to use the MAINT userid to load and test a new CP nucleus, you should be sure that MAINT's virtual machine has:

- A minimum 512K of virtual storage. The loader requires 512K to execute. In general, MAINT's virtual machine should have as much virtual storage as the real machine storage size.

- The ECMODE option specified in the VM/370 directory (or has used the CP SET ECMODE ON command). ECMODE is required for testing the CP system in your virtual machine.

- Write access to the CP system residence volume, or a minidisk that is a replica of the system residence volume. The minidisk must be defined in your virtual machine at the same address as the real address of the system residence volume. The minidisk must have been formatted with the CP Format/Allocate program, such that it resembles the CP system residence; nonexistent cylinders beyond the extent of the minidisk must be allocated as permanent space (PERM).

When you prepare to load a new CP nucleus, you should be sure that you have the disks containing object modules accessed in the proper order to ensure that the correct files are punched. Then you issue the following series of commands:

```
close punch
purge punch all
close reader
purge reader all
spool punch *
vmfload yourload yourown
```

where YOURLOAD is the loadlist EXEC file and YOUROWN is the control file.

When the VMFLOAD program completes, you receive the messages:

```
SYSTEM LOAD DECK COMPLETE
PUN FILE 0821 TO MAINT COPY 01 NOHOLD
```

At this point the standalone loader (DMKLD00E LOADER) is in your card reader, followed by all of the text decks necessary to construct a CP nucleus. There are several ways to handle this reader file.

First, you can use the CMS MOVEFILE command to place the entire file on tape, thus creating a CP nucleus load tape. Later you can IPL the tape drive on the real machine when you want to update the CP system. Remember, however, that the loader requires 512K of storage.

The second way to handle the CP nucleus reader file is to IPL the loader from the tape. If you have access to the real system residence device in your virtual machine, the nucleus is written on the real system residence volume. If you have a minidisk defined at a virtual address corresponding to the real address of the CP system residence disk, the nucleus is written on that disk.

A third way is to IPL the nucleus directly from the card reader; this method is shown in the example that follows. However, it does not provide you a backup copy that you can IPL.

For example, if MAINT's virtual machine has entries for the real system residence volume at address 330, and for a minidisk replica at address 331, you may detach the real system residence volume and define the minidisk at that address:

```
detach 330
define 331 as 330
```

Now you can IPL the CP nucleus, specifying the address of your virtual card reader. When the load operation completes, the message "NUCLEUS LOADED ON SYSRES" is displayed, followed by a message indicating a disabled wait state PSW, the normal termination of the standalone loader program.

```
ipl 00c
NUCLEUS LOADED ON SYSRES
DMKDSP450W CP ENTERED; DISABLED WAIT PSW
CP
```

When you IPL the nucleus, the load map is spooled to your virtual printer. You must issue the CLOSE command to close the spool file. If you want to retain a copy of the load map as a CMS disk file, you first issue the command:

```
spool printer to *
```

so that the load map is routed to your card reader and you can later use the CMS READCARD command to write the load map on disk.

Now, define your console address to be the same as defined in the RIOGEN macro in DMKRIO. Then you can IPL the system residence device, which is the virtual disk with an address of 330.

```
def 009 as cuu
CONS cuu DEFINED
ipl 330
```

The following example shows the IPL of the nucleus. The two error messages (both DMKLNK108E) occur because UDISK1 and UDISK2 are not defined in MAINT's virtual machine configuration.

```
VM/370 VERSION n LEVEL 0

NOW 14:53:07 EST THURSDAY 03/28/74
CHANGE TOD CLOCK (YES|NO) :no
14:53:50 DMKLNK108E CMSSYS 190 NOT LINKED; VOLID UDISK2 NOT MOUNTED
RRRR....RING....GGGG
14:53:50 DMKLNK108E OPERATOR 191 NOT LINKED; VOLID UDISK1 NOT MOUNTED
RRRR....RING....GGGG
14:53:50 START ((COLD|WARM|CKPT|FORCE) (DRAIN|SHUTDOWN)):shutdown
14:53:50 AUTO LOGON *** OPERATOR USERS = 001 BY SYSTEM

DMKCPI960W SYSTEM WARM START DATA SAVED

DMKCPI961W SYSTEM SHUTDOWN COMPLETE

DMKDSP450W CP ENTERED; DISABLED WAIT PSW
CP
```

After you check the new CP, you may redefine your console and IPL the CMS system (CMS accepts only 009 and 01F as valid console addresses). After you IPL CMS, you can use the DDR command to create a backup copy of the CP nucleus (which can then be restored to the real system).

Define your input unit as the address of your system residence device. Your output unit is the tape (181) that is attached to your virtual machine. When you enter the DUMP statement with the NUCLEUS operand, DDR creates a copy of the nucleus that was just loaded.

The sequence of commands and responses is:

```
def cuu as 009
CONS 009 DEFINED
ipl cms
CMS...mm/dd/yy
ddr
ENTER: in 330 3330 sysres
ENTER: out 181 2400
ENTER: dump nuc
DUMPING SYSRES
END OF DUMP
ENTER:
END OF JOB
R; T=0.21/2.63 15:04:04
```

You created a backup copy of the CP nucleus. This copy may later be restored using the standalone version of the DDR program on the real machine.

Updating CMS

The procedures for updating CMS source files and macro libraries are the same as for updating CP. The order of search for CMS updates is:

```
191 A   R/W
193 B/A R/O
190 C/A R/O
393 D/A R/O
```

where 193 contains PTFS, control files, and user updated TEXT decks and 393 contains the CMS source files. 190 contains the current CMS system, including text decks, command modules, and the CMS nucleus.

You might use the following steps when you update CMS:

1. Format the minidisk you are going to use to test the CMS nucleus, if any.
2. Use the VMFLOAD program to punch the updated CMS object modules.
3. Regenerate any disk-resident modules that have been updated.
4. Load the new CMS nucleus.
5. Save the CMSSEG discontiguous shared segment and the new CMS operating system. CMS should be resaved whenever the S-disk is updated. This will insure that the saved CMS system reflects the physical system.

The exact steps that you take depend on whether you are testing the CMS nucleus before you load it onto the system disk, whether you are using shared segments, and so on.

Disks for Updating CMS

If you want to keep CMS source files on disk, the minidisk you use must be at least 145 cylinders for a 2314 (or 2319), 80 cylinders for a 3330 disk, 190 cylinders for a 3340 disk, or 40 cylinders for a 3350 disk. Then, you should have the CMS source tape mounted and attached to the virtual machine, and issue the following commands to load the source programs onto the CMS disk:

```
vmfplc2 fsf
vmfplc2 load (eof 2)
```

If you want to test the new CMS nucleus in a virtual machine before you update the real CMS system, you should have a disk available for a copy of the nucleus. The configuration shown for MAINT in "A Virtual Machine for Updating VM/370" shows a 6-cylinder minidisk at virtual address 390 for testing the CMS nucleus.

You can test updated disk-resident CMS modules on your A-disk before moving them to the CMS system disk (190).

FORMATTING A DISK TO TEST THE CMS NUCLEUS

Before you can use the minidisk you have available for testing CMS, it must be formatted with the CMS FORMAT command. For example, to format the 390 minidisk, you might issue:

```
format 390 g
```

Now, you must reissue the FORMAT command with the RECOMP option, so that the number of cylinders on the disk is recomputed to reserve space for the CMS nucleus at the end of the disk. To do this, format the disk with one or two cylinders fewer than it actually has (one cylinder on a 3330 or 3350, two cylinders on a 2314 or 3340).

For example, if the 390 minidisk is a 3-cylinder 3330, enter

```
format 390 g 2 (recomp
```

The 390 disk is now ready for use as the CMS test nucleus.

You should not have to reformat the disk again; you can use it each time you update CMS.

CONSIDERATIONS FOR CREATING A NEW CMS SYSTEM DISK

If you want to create a new CMS system disk that contains all the CMS text and MODULE files as well as the CMS nucleus, do the following:

- If you are going to save this CMS system, be sure that the operands VSYSADR, SYSCYL, and VSYSRES in the NAMESYS macro corresponding to this system are correct.
- After copying all the existing files with filetypes of TEXT and MODULE onto the new disk, regenerate any modules that use auxiliary directories (such as the ASSEMBLE command). Auxiliary directories are described in the _VM/370 System Programmer's Guide_. You can use the CMSGEND EXEC procedure to regenerate the assembler. Some IBM Program Products may also use auxiliary directories.

Punching the CMS Nucleus

When you prepare to build a new CMS nucleus, be sure that you have access to the text decks on the system disk, as well as any updated decks that you may have created. Since the CMS text decks are on the CMS system disk (usually 190), you should access it so that you have these text decks available for the VMFLOAD program:

```
access 190 a
```

Be sure that your virtual card punch and reader do not have any files in them and that your virtual punch is spooled to your virtual reader:

```
close punch
purge punch all
close reader
purge reader all
spool punch to *
```

Then you can issue the VMFLOAD command specifying the CMS loadlist EXEC filename and the control file filename:

```
VMFLOAD CMSLOAD DMSRn0
```

In this example, the system-supplied CMSLOAD EXEC and DMSRn0 CNTRL files are used to punch a new CMS nucleus.

When you receive the messages

```
SYSTEM LOAD DECK COMPLETE
PUN FILE 0353 TO MAINT COPY 01 NOHOLD
```

a new copy of the CMS nucleus is available in your card reader. Before you go on to load the new nucleus, you may want to regenerate any CMS MODULE files that have been updated. This procedure is described next.

To determine whether an update requires module regeneration see "Appendix C: CP/CMS Regeneration Requirement." If you do not need to regenerate any modules, see "Loading a CMS Nucleus."

Creating CMS Disk-Resident Modules

The CMSGEND EXEC procedure creates CMS disk-resident command modules from CMS text files. CMSGEND is invoked by specifying the filename of the module to be generated. For example, if there is a change to the text file DMSACF, you must generate a new ACCESS MODULE.

```
cmsgend access
```

CMSGEND will rename any existing file from 'ACCESS MODULE A2' to 'ACCESS MODOLD A1'. After an existing file of 'ACCESS MODOLD A1' is erased CMSGEND then loads the text files that comprise the ACCESS command module and generates a new ACCESS module A2.

When you use CMSGEND, you must access the S-disk as your read/write A-disk, and have all pertinent text files available. The text files must have a filetype of TEXT; thus, if you have updated an object module using VMFASM, and the most recent object file has a filetype such as TXTLOCAL, you must rename it to a filetype of TEXT. (Note that if there is currently a text file on the system disk, you may want to rename it also, so that your updated text file, on some other disk, is the one that is loaded.)

CMSGEND displays status messages as it executes. For example:

```
cmsgend access

*** CURRENT STATUS:
FILE ' ACCESS MODULE A2' DOES NOT EXIST
FILE ' ACCESS MODOLD A1' DOES NOT EXIST

*** LOADING:
 INVALID CARD - *       CMSLIB   MACLIB   A2 RnM190 12/04/75  04:20
 INVALID CARD - *       DOSMACRO MACLIB   A2 RnM190 10/16/75  23:19
 INVALID CARD - *       DMSACC   ASSEMBLE A1 RnM303 12/03/75  04:02
 ACCESS   SD 00E000
 INVALID CARD - *       CMSLIB   MACLIB   A2 RnM196 10/16/75  23:19
 READFST  SD 00EBC0
 DMSACM   SD 00EF10
 READMFD     00EF10
```

```
INVALID CARD - *        OSMACRO  MACLIB   S2 RnM290 10/16/75  22:47
INVALID CARD - *        OSMACRO1 MACLIB   S2 RnM290 10/16/75  22:49
DMSALU    SD 00F4A8
RELUFD       00F4A8
SORTFST      00F716
END$RELU     00FF38

*** RESULTS:
' ACCESS MODULE A2' CREATED FROM TEXT DECK ( S ) DMSACC DMSACF
DMSACM DMSALU WITH ATTRIBUTES TRANS SYSTEM NOMAP
```

Since CMSGEND renames the existing module, users who are currently using the CMS system disk are unaffected by the regeneration procedure. This is because the SSTAT (system status table) of the CMS system disk is still pointing to the old (renamed) module. Whenever 190 is subsequently IPLed, the SSTAT points to the updated modules, so that the old module can be erased.

Loading a CMS Nucleus

When you are ready to load the CMS nucleus, you should plan ahead for two situations.

If you are going to test the CMS nucleus on a minidisk other than 190 (we are using 390 in this example), you may want to save the nucleus reader file so that you do not have to repeat the VMFLOAD procedure if the nucleus tests out all right. To do this, issue the command:

```
spool reader hold
```

You may also want to issue the command

```
spool printer to *
```

so that the nucleus load map is routed to your card reader, instead of the virtual printer.

If your CMS system uses the CMSSEG discontiguous saved segment, you should anticipate that it may not be compatible with the new CMS nucleus. Later, you will want to use the CMSXGEN procedure to save the segment, but for testing purposes, you do not need it. Therefore, to prevent CMS from attempting to attach CMSSEG after IPL, you can define your virtual storage to 2M:

```
define storage 2m
```

Now you can issue the IPL command to load the CMS nucleus:

```
ipl 00c clear
```

During the IPL sequence, you must respond to the following messages.

DMSINI606R SYSTEM DISK ADDRESS = cuu

Enter the device address (cuu) of the system disk (S-disk). This is usually 190. On this disk CMS expects to find all CMS system information and programs not contained within the CMS nucleus, such as the disk-resident command modules. If the CMS nucleus is written on this disk, then cuu is also the IPL device address.

If you enter an invalid device address, the message

DMSINI079E INVALID DEVICE ADDRESS - REENTER

is issued. Message DMSINI606R is reissued so that you can enter a valid device address.

If you press the carrier return without entering a device address, X'190' is assumed to be the system disk address.

DMSINI615R Y-DISK ADDRESS = cuu

Enter the device address (cuu) of the system disk extension (Y-disk). On this disk CMS expects to find all CMS system information and programs not contained within the CMS nucleus and not on the S-disk. If the CMS nucleus is written on the Y-disk, then cuu is also the IPL device address.

If you enter an invalid device address, the message:

DMSINI079E INVALID DEVICE ADDRESS - REENTER

is issued. Message DMSINI615R is reissued so that you can enter a valid device address.

If you press the carrier return without entering a device address, X'19E' is assumed to be the address of the system disk extension.

Note: If you do not want to have a Y-disk, do not attach the device that was specified (or defaulted to) as the Y-disk address.

DMSINI607R REWRITE THE NUCLEUS? (YES|NO)

If you enter "yes", a copy of the CMS nucleus is written onto the disk indicated in the response to message DMSINI608R. If you enter "no", the CMS nucleus is not written to disk.

If you enter neither "yes" nor "no," the message

DMSINI081E INVALID REPLY - ANSWER "YES" OR "NO"

is issued. Message DMSINI607R is reissued so that you can enter a valid response.

If you enter "no", the remaining messages in generating a new CMS nucleus are skipped and control is passed to the CMS initialization routine.

DMSINI608R IPL DEVICE ADDRESS = cuu

Enter the address of the device (cuu) on which the CMS nucleus is to be written. If you are using 390 to test the CMS nucleus, you enter: 390. If the system disk and the IPL device are to be the same, you need only press the carrier return.

If you enter an invalid device address, the message

DMSINI079E INVALID DEVICE ADDRESS - REENTER

is issued. Message DMSINI608R is reissued so that you can enter a valid device address.

If the IPL device you designated is not currently defined, is not in read/write status, or is an unsupported device type, the message

    DMSINI082E IPL DEVICE ERROR - REENTER

is issued. Message DMSINI608R is then reissued. At this time, you may enter CP mode by pressing the Attention key (or equivalent), then determine the status of the device you designated by entering the CP command

    QUERY VIRTUAL cuu

and take the corrective action necessary to define the device for your virtual machine or to access it in read/write status. You may reenter CMS by issuing the CP command

    BEGIN

Then you must reenter the device address. Once the device address is accepted, message DMSINI609R is issued.

DMSINI609R NUCLEUS CYL ADDRESS = nnn

Enter the 1- to 3-digit cylinder number (nnn), for the device entered in response to message DMSINI608R, where the CMS nucleus is to be written. The number (nnn) must be between 1 and m-1 (where m equals the number of cylinders on the disk). The number nnn must be entered in decimal. This is the cylinder you reserved when you formatted the disk with the RECOMP option. In our example, since the nucleus is written on the last cylinder of MAINT's 390, you enter: 2

If you do not enter a valid decimal cylinder number, the message

    DMSINI080E INVALID CYLINDER NUMBER - REENTER

is issued. Message DMSINI609R is reissued and you may enter a valid cylinder number.

If the cylinder specified is not greater than the number of cylinders already in use on the device (as indicated in the master file directory, then the message

    DMSINI083E NUCLEUS WILL OVERLAY FILES - RECOMPUTE

is issued. You may respond with a larger cylinder number, or IPL CMS and format the specified IPL device with the RECOMP option.

DMSINI610R ALSO IPL CYLINDER 0? (YES|NO)

The initial IPL text is always written on the same cylinder as the CMS nucleus (the cylinder designated in response to message DMSINI609R). The initial IPL text is a bootstrap program which reads the nucleus from the designated cylinder. If it is not also written on cylinder 0, then you must enter the cylinder number when subsequent IPL commands are issued for the system being generated. See the IPL command description in the _VM/370 CP Command Reference for General Users_. Your response has the following meaning:

- yes — Initial IPL text is written on cylinder 0 as well as on the cylinder designated in response to message DMSINI609R.
- no — Initial IPL text is written only on the cylinder designated in response to message DMSINI609R.

If you do not enter "yes" or "no," the message

```
DMSINI081E INVALID REPLY - ANSWER "YES" OR "NO"
```

is issued. Message DMSINI610R is reissued so that you can enter a valid response.

If your response is valid, message DMSINI611R is issued.

DMSINI611R VERSION IDENTIFICATION =

Enter up to 32 bytes of information, including blanks, to specifically identify the version and level of CMS; this information is printed each time you IPL the CMS system now being generated. The default identification (specified by a carrier return) is:

```
CMS VERSION n.n - mm/dd/yy
```

where n.n is the version and level of CMS, and mm/dd/yy is the month, day and year the CMS nucleus was created.

DMSINI612R INSTALLATION HEADING =

Enter up to 64 bytes of information, including blanks, to serve as an installation standard heading at the beginning of each output file. The default heading (specified by a carrier return) is:

```
CONVERSATIONAL MONITOR SYSTEM
```

The nucleus is then written on the specified disk cylinder and the version identification is displayed, indicating that the CMS system is loaded successfully and is ready to accept CMS commands.

You can use this copy of CMS to test updates and changes, including changes to CMS modules that you may have made with the CMSGEND EXEC.

Before you test the CMS system, you can create a disk file from the CMS nucleus and the nucleus load map:

```
spool rdr nohold
close prt
close rdr
PRT FILE 0342 TO MAINT COPY 01 NOHOLD
```

Now you can read a copy of the CMS nucleus onto disk:

```
read cmsnuc nucleus a1
```

and read a copy of the CMS load map:

```
read cmsnuc loadmap a1
RECORD LENGTH IS '132' BYTES.
```

You now have two CMS files on your 191 disk: CMSNUC NUCLEUS, which contains the CMS nucleus created above, and CMSNUC LOADMAP, the load map for this nucleus.

After you test the new CMS nucleus on 390, and you are satisfied that it is all right, you can use the disk file to create the new nucleus on the system disk (190).

To regenerate a nucleus which exists as a disk file (CMSNUC NUCLEUS, for example), issue the following commands:

```
spool pun to *
punch cmsnuc nucleus a1 (noheader)
ipl 00c
```

You may then answer the IPL messages previously described. This time, you specify the IPL address as 190 instead of 390, and enter the correct cylinder for your system disk. Now you can go on to save the CMSSEG and the CMS saved system, if you wish.

Note: If a named saved system has been built from this CMS system disk, it must be resaved because the SSTAT is recreated only when the disk is loaded (for example 190). Appendix C details what must be regenerated for changes in any CMS text file.

Saving CMSSEG and the CMS System

If your system has entries in the system name table (DMKSNTBL) for a CMSSEG discontiguous segment and a named CMS, you should now save these system names. To do this, first be sure that you have defined virtual machine storage to a value above the location of CMSSEG and the loader tables, for example 2M.

Then, you can IPL CMS and issue the CMSXGEN command to save the CMSSEG segment:

```
ipl 190 parm seg=null
  (null line)
Y(19E) R/O
R;
access 190 b/a
R;
cmsxgen 100000
```

where 100000 is the hexadecimal address where the segment is loaded. This number must correspond to the starting page number specified in the NAMESYS macro for the CMSSEG saved system name. CMSXGEND generates a LOAD MAP on 00E which will appear in your reader and is required by IPCS. When the CMSXGEN procedure is completed, you should IPL the CMS system disk again, and issue the CP SAVESYS command immediately, before pressing a carriage return to complete the IPL:

```
ipl 190
savesys cms
```

In this example, CMS is the name of the saved CMS system. If you have specified another name for the saved CMS system, you should specify that name when you issue the SAVESYS command. SAVESYS is a CP privilege class E command; it allows you to write on the CP system residence volume.

Now, the saved portion of CMS may be shared among many users, who can load CMS by referring to its saved name, for example

```
ipl cms
```

When you IPL a saved CMS system, CMS operates as if an IPL of a specific device had occurred, with the single exception that the directory for the system disk is part of the nucleus.

Update Procedures for CMS/VSAM and Access Method Services

You are responsible for ordering and applying all updates to DOS/VS that affect the VSAM and access method services routines and the DOS logical transients ($$BOMSG1, $$BOMSG2, $$BOMSG7, and $$BENDQ) that CMS distributes. You can order these updates in card form or on tape.

If you order the updates on cards, you must remove all of the DOS/VS job control statements from each PTF (program temporary fix) deck and place a CP ID card at the beginning of the deck. The CP ID card must contain the userid of the CMS virtual machine that is being used to update VSAM and access method services.

For example, if you want to apply an update to a module using the MAINT virtual machine, your ID card is:

    ID MAINT

If you installed VSAM and access method services under CMS as an OS user, VSAMGEN created CMS text files for all the required DOS/VS relocatable library modules. Now you need not have a DOS/VS relocatable library for updating. CMS creates text files from the updates and replaces the old text files on your A-disk with the new text files.

Text files must have a filetype of TEXT. For example, after you have updated an object module using VMFASM, the most recent object file has a filetype such as TXTLOCAL. To use that text file here, you must rename it to a filetype of TEXT. If there is currently a text file on the system disk, you may want to rename it too, so that your updated text file (which may reside on another disk) is the one that is loaded.

VSAMGEN UPDATE CONSIDERATIONS

Applying DOS/VS PTFs to either the CMSVSAM or the CMSAMS discontiguous saved segments may result in the generated segment exceeding the space defined for it in the system name table (see the NAMESYS macro of the DMKSNT module). You may want to anticipate this problem by defining in the system name table an additional shared and nonshared segment for each of the discontiguous saved segments (CMSVSAM and CMSAMS). This is one way of providing for additional growth.

Alternatively, upon completion of the VSAMGEN update procedure, you can check whether the updated segments have exceeded their definitions and correct that situation as follows:

1. Determine the new size of the changed VSAM and/or access method services shared and nonshared segments by subtracting the phase LOCORE address from the HICORE address indicated on the linkage editor map. The phase names are:
   - DMSVVS - VSAM shared
   - DMSVVN - VSAM nonshared
   - DMSVAS - access method services shared
   - DMSVAN - access method services nonshared
   - DMSV33 - VSAM shared (DOS/VS Release 33 or 34)
2. Compare the new sizes of these segments with the sizes of the corresponding shared or nonshared segments as defined in your DMKSNT NAMESYS macro.

3. If the new size exceeds your defined size, recode the NAMESYS macros to include an additional segment. Refer to the phase names listed in Step 1 to determine whether the segment is shared or nonshared. To add one segment:

   - Increase the SYSPGCT operand by 16
   - Increase the SYSPGNM operand by 16
   - Increase the SYSHRSG operand by 1, if the segment is shared
   - Increase the SYSSIZE operand by 64K
   - Change the SYSSTRT operand of this or other segments, if the increase in this segment causes any segments to overlap

4. Reassemble the DMKSNT module, build a new CP nucleus, and then reexecute the VSAMGEN procedure.

If a PTF contains a new VSAM or access method services module, it is not included in CMSVSAM or CMSAMS during VSAMGEN unless you have current level of installation files.

If you want to use a new release level of DOS/VS, you must regenerate the CMSVSAM and CMSAMS segments using the starter system supplied by DOS/VS. VM/370 also provides new installation EXECs and DOSLNK files, which you must use to install the new release properly. The installation procedure is described in Part 3.

Note: It is not necessary to regenerate existing CMSVSAM and CMSAMS segments using the Release 34 starter system. If you do so, however, you must be sure that the space allocation for the CMSVSAM segment in the system name table (DMKSNT) file is increased to five shared segments and one nonshared segment, and the space allocation for the CMSAMS segment is increased to 6 shared segments and 2 nonshared segments. Note that the segment addresses above these segments must be increased accordingly.

USING VSAMGEN TO UPDATE CMS VSAM AND ACCESS METHOD SERVICES

Before you invoke VSAMGEN, do the following:

- Access a CMS read/write disk as your A-disk. This disk must be the same disk that you used as your A-disk when you installed VSAM and access method services.
- If you are a DOS user, link to and access the DOS/VS system disk.
- If you are applying PTFs from tape, a tape drive must be attached at virtual address 181 and the PTF tape must be mounted and positioned at the first tape file you want processed. VSAMGEN assumes that the PTF tape is unlabeled and contains 3440-byte blocks of records.

Use the VSAMGEN EXEC procedure to update VSAM and Access Method Services support in CMS. Invoke VSAMGEN as follows:

```
vsamgen
```

VSAMGEN prompts you to enter what type of user you are and whether you plan to install or update VSAM and access method services. You receive the following messages:

```
DMSVGN360R ENTER EITHER 'INSTALL' OR 'UPDATE':
DMSVGN361R ENTER EITHER 'DOS' OR 'OS':
```

At this time, you should respond: UPDATE and either DOS or OS. If you are both, respond "DOS."

VSAMGEN requires a read/write A-disk and checks that one is available. If an A-disk is not available, VSAMGEN issues one of the following messages and terminates:

```
DMSVGN069E DISK 'A' NOT ACCESSED
DMSVGN361E DISK 'A' IS NOT A CMS DISK
```

You are prompted to enter the release level of the DOS/VS starter system you are using:

```
DMSVGN369R ENTER RELEASE NUMBER OF THE DOS/VS STARTER SYSTEM:
```

You should enter 31, 32, or 33 or 34. If you enter anything else, you receive the message

```
DMSVGN369E INVALID - RELEASE 31 OR LATER REQUIRED
```

and the VSAMGEN EXEC procedure terminates.

If you are a DOS user, you are asked to identify the DOS/VS relocatable library by filemode.

```
DMSVGN362R ENTER MODE OF DOS SYSTEM RELOCATABLE LIBRARY DISK:
```

Next, VSAMGEN checks that the files it requires are on an accessed disk and invokes the CMS/DOS environment. You are now prompted to indicate whether you want to update VSAM or access method services or both.

```
DMSVGN364R ENTER 'CMSVSAM' OR 'CMSAMS' OR 'BOTH' FOR GENERATION
           OF NEW SYSTEM(S).
```

Now, you must indicate whether the updates you are applying are in card format or on tape. Respond to the message:

```
DMSVGN380R ENTER 'TAPE' OR 'CARDS' FOR PTF APPLICATION:
```

The following two sections describe the procedure for applying card and tape PTFs to CMS VSAM and access method services.

Applying Updates from Cards

If you reply CARDS, you must already have placed each PTF deck, with a CP ID card as the first card in the PTF deck, in a real card reader and must have read the decks into the virtual card reader. VSAMGEN indicates it is reading the PTF decks by issuing the message:

```
DMSVGN366I STARTING TO READ PTF DECKS FROM READER...
```

You must now indicate what module you want updated by responding to the message:

```
DMSVGN365R ENTER MODULE NAME (8 CHARS OR LESS) OR 'END':
```

The order of the names you give in response to this message must be the same as the order in which you placed the decks in the reader.

Each time you respond to this message, VSAMGEN checks that a module with that name already exists on the A-disk, erases any old text file with that name, renames the current text file to fn TEXTOLD, and reads the new text file in and writes it on the CMS A-disk. When the new text file replaces the old, you receive the message:

```
DMSVGN367I 'fn TEXT' WRITTEN ON DISK 'A'.
```

You will receive message DMSVGN365R again; either enter the name of the next text file you want to update, or enter END.

When all the new text files are written to the CMS A-disk, you receive the status message:

```
DMSVGN368I nn NEW PTF DECKS WILL BE APPLIED
```

From this point on, the procedure for updating VSAM and access method services is identical to the installation procedure for fetching, link-editing, loading and saving them. You receive the following messages:

```
DMSVGN362I LINK-EDITING {CMSVSAM}...
                        {CMSAMS }

DMSVGN363I {CMSVSAM} DOSLIB CREATED ON DISK 'A'
           {CMSAMS }

DMSVGN363R ENTER LOCATION WHERE {CMSVSAM} WILL BE LOADED AND SAVED:
                                {CMSAMS }

DMSVGN366R ENTER NAME OF SYSTEM TO BE SAVED:
```

VSAMGEN fetches the modules, loads them at the designated address, assigns storage protection keys, and saves the segments. You receive the completion message:

```
DMSVGN365I SYSTEM segmentname SAVED.
```

Finally, you must indicate whether or not you wish to erase the DOSLIB created during link-edit. You receive the following message:

```
DMSVGN368R ERASE {CMSVSAM} DOSLIB? ... ENTER YES OR NO:
                 {CMSAMS }
```

Applying Updates from Tape

If you replied TAPE to the DMSVGN380R message, you must now indicate whether you want to apply all the PTFs or just a selected number of PTFs. Reply to the message:

```
ENTER 'SELECT' OR 'ALL' FOR TAPE PTF APPLICATION:
```

You must also indicate how many PTF tape files are now to be processed; respond to the message:

```
DMSVGN382R ENTER NUMBER OF TAPE FILES TO BE PROCESSED:
```

If you entered ALL, the installation procedure applies all the text files that affect the CMS VSAM and access method services support. If you entered SELECT, the installation procedure sends you a message each time it finds a text file that is used by CMS to support VSAM and access method services. You must reply to these messages:

```
DMSVPD383R APPLY 'filename'? ... ENTER 'NO' OR EOB:
```

Before VSAMGEN writes a new text to the A-disk, it checks that a module with that name already exists on the A-disk, erases any old text file with that name, renames the current text file to fn TEXTOLD, and writes the new text file on the CMS A-disk. When the new text file replaces the old, you receive the message:

```
DMSVPD367I 'fn TEXT' WRITTEN ON DISK 'A'.
```

When all the new text files are written to the CMS A-disk, you receive the status message:

```
DMSVPD368I nn NEW PTF DECKS WILL BE APPLIED
```

From this point on, the procedure for updating VSAM and access method services is identical to the installation procedure for fetching, link-editing, loading, and saving them. You receive the following messages:

```
DMSVGN362I LINK-EDITING {CMSVSAM}...
                        {CMSAMS }

DMSVGN363I {CMSVSAM} DOSLIB CREATED ON DISK 'A'
           {CMSAMS }

DMSVGN363R ENTER LOCATION WHERE {CMSVSAM} WILL BE LOADED AND SAVED:
                                {CMSAMS }

DMSVGN366R ENTER NAME OF SYSTEM TO BE SAVED:
```

VSAMGEN fetches the modules, loads them at the designated address, assigns storage protection keys, and saves the segments. You receive the completion message:

```
DMSVGN365I SYSTEM segmentname SAVED.
```

Finally, you must indicate whether or not you wish to erase the DOSLIB created during link-edit. You receive the following message:

```
DMSVGN368R ERASE {CMSVSAM} DOSLIB? ... ENTER YES OR NO:
                 {CMSAMS }
```

Updating Considerations for CMS/DOS

CMS/DOS has no effect on the update procedures for DOS/VS, DOS/VS COBOL, or DOS PL/I. You should follow the normal update procedure for applying IBM-supplied coding changes to them.

Updating RSCS

The same procedure used to update CP and CMS can be used to update RSCS. However, unlike CP and CMS, RSCS can test the system that is built; it does not need to test a duplicate copy of the system that is built. Again, the MAINT virtual machine can be used to do the updating. You should link to the RSCS virtual machine's 191 minidisk as your 195 and access it as your A-disk.

The order of search for updating is:

```
195  A    R/W
194  B/A  R/O
190  S    R/O
```

To build a new RSCS nucleus, you must create a new RSCS system disk and generate a new RSCS nucleus.

Creating an RSCS System Disk

Use the following procedure to create a new RSCS system disk:

1. Log on as MAINT.
2. IPL 190.
3. Link to the minidisk that you want to contain the new RSCS nucleus as 195 in write status and access it as your A-disk.
4. Issue the CMS FORMAT command to format that minidisk.
5. Issue the CMS FORMAT command with the RECOMP option to format the same minidisk with one or two cylinders less than the total number of cylinders on the disk (one less on a 3330 or 3350, two less on a 2314 or 3340). The last cylinders are used for the RSCS nucleus.
6. If you wish to change the RSCS configuration, re-create the AXSLINKS, LAXLINES, and TAGQUEUE COPY files and create a new DMTLOC macro library. See "Part 3. Generating VM/370 (CP, CMS, RSCS and IPCS)."
7. Generate a new RSCS nucleus using the commands described in the following section.

Generating the RSCS Nucleus

1. Load the RSCS files onto the CP system disk (the 194 minidisk belonging to MAINT) if they are not already there.
2. Access the MAINT 194 disk as an extension of the RSCS system disk.

   access 194 b/a

Note: If you want to apply your own updates, they should be on the 294 disk. Then disk access should be:

```
access 294 b/a
access 194 c/a
```

3. Assemble the RSCS configuration table module, DMTSYS, using the VMFASM EXEC procedure. The control file DMTRn0 CNTRL identifies DMTLOC as the macro library.

```
VMFASM DMTSYS DMTRn0¹
```

   If an error occurs due to an incorrectly coded macro, correct the macro and restart by generating a new DMTLOC macro library.

   Note: If you do not have enough disk space to assemble, acquire additional T-disk space.

4. Close and clear the punch and reader. Spool the punch to your virtual card reader. Use the VMFLOAD EXEC procedure to punch the RSCS nucleus. The loadlist EXEC, DMTLOAD, contains the filenames of all the RSCS TEXT modules.

```
close pun
close rdr
purge rdr all
purge pun all
spool pun to *
vmfload dmtload DMTRn0
SYSTEM LOAD DECK COMPLETE
PUN FILE spoolid TO userid
```

5. Copy the RSCS supervisor text decks, DMTAXS and DMTLAX, from the MAINT 194 disk to your RSCS system disk.

```
copyfile dmtaxs text b1 = =  a1 (replace olddate
copyfile dmtlax text b1 = = a1 (replace olddate
```

6. Copy the required line driver TEXT decks, DMTNPT and/or DMTSML, from the MAINT 194 disk to the RSCS system disk. The MAINT 194 disk can now be released and detached.

```
copyfile dmtnpt text b1 = = a1 (replace olddate
copyfile dmtsml text b1 = = a1 (replace olddate
release 194
detach 194
DEV 194 DETACHED
```

7. IPL your card reader which contains the RSCS nucleus A series of prompting messages requests information relative to the physical location and disk address of the RSCS nucleus. Answer them as shown.

```
ipl c
DMTINI406R SYSTEM DISK ADDRESS = 195
DMTINI407R REWRITE THE NUCLEUS ? yes
DMTINI409R NUCLEUS CYL ADDRESS = 003 (for 2314 or 3340, or 004
                                      for 3330 or 3350)
DMTINI410R ALSO IPL CYLINDER 0 ? yes
```

----

[1] DMTRn0 may be DMTR40, DMTR50, DMTR60 and so forth, depending on the release level.

The message

```
DMTAXS103E FILE 'spoolid' REJECTED -- INVALID DESTINATION
           ADDRESS
```

is issued. It indicates that the RSCS nucleus file was purged from the card reader.

8. The message shown in line 1 of the following example indicates completion of the writing of the RSCS nucleus on the specified disk. IPL the specified disk and start your RSCS operations as described in the _VM/370 Remote Spooling Communications Subsystem (RSCS) User's Guide_.

```
DMTREX000I RSCS (VER n, LEV n, mm/dd/yy) READY
           !
           CP
           logoff
           logon rscs
           ipl 191
DMTREX000I RSCS (VER n, LEV n, mm/dd/yy) READY
start newyork
         .
         .
         .
         .
         installation RSCS operation
         .
         .
```

Note: If you are logged on as MAINT, you IPL 195, but if you are logged on as RSCS, you IPL 191 to IPL the RSCS system disk.

Updating IPCS Modules

You can use the same procedures used to update VM/370 CP, CMS, or RSCS modules to update IPCS modules. Since there is no nucleus associated with the IPCS component, the procedure is simplified.

If you have the IPCS source files on MAINT's 394, and the PTFs and update files on 294, then, to reassemble an IPCS module, you might have the search order as

```
191  A    R/W
294  B/A  R/O
394  C/A  R/O
390  S    R/O
```

you could then use the VMFASM EXEC to assemble the module, for example:

```
vmfasm dmmpro DMMRn0
```

To generate the new IPCS commands, you should use the LOAD and GENMOD commands to generate the new command modules from updated text files on MAINT's 191, and nonupdated text files from the 194. Once the new IPCS module has been generated, copy it to the CMS S-disk (MAINT's 190). For example, you issue the commands:

```
access 195 a
access 191 b/a
access 194 c/a
```

The command names, and the CMS commands you need to issue to generate them, are shown in Figure 39.

IPCS Command Name	CMS Commands to Generate
PROB	load dmmpro genmod prob
DUMPSCAN	load dmmdsc genmod dumpscan
VMFDUMP	load dmmedm genmod vmfdump (to dmmext genmod vmfdump2 (from dmmext
STAT	load dmmsta genmod stat
SUMMARY	load dmmsum genmod summary

Figure 39. IPCS Command Names

Updating Service Programs

Service programs are CP modules that are not a part of the CP nucleus. They may execute either standalone from a card reader (the real system card reader or your virtual card reader) or in some cases, as a CMS command. The service programs are:

- DASD Dump/Restore (module DMKDDR)
- Directory program (module DMKDIR)
- Format/Allocate program (module DMKFMT)
- IBCDASDI, the virtual disk initialization program (module IBCDASDI)
- NCPDUMP, the 3704/3705 dump program (module DMKRND)

If you apply a PTF to a service program, you may use the GENERATE EXEC to create a new IPLable copy of the service program that can be loaded via IPL or the CMSGEND EXEC to create a new CMS command module, or both.

For example, the Directory program exists as the CP module DMKDIR. To apply PTFs to the source file, you would use the VMFASM EXEC procedure, as follows:

```
vmfasm dmkdir DMKRn0
```

where DMKRn0 is the filename of the control file (the filetype is CNTRL).

The Directory program can be used in three ways: (1) as a standalone program that you IPL from the real system card reader (2) as a standalone program that you can IPL from virtual card reader, and (3) as a CMS command, DIRECT.

| To create a new standalone copy for loading by IPL, you can use the GENERATE EXEC:

```
generate ipldeck
```

you are prompted to enter the name of the program with the message

```
ENTER THOSE DECKS TO BE GENERATED ( DDR | DIR | FMT | ALL )
```

you enter:

```
dir
```

Then the GENERATE EXEC prompts you to enter the target disk address (where the deck will reside):

```
ENTER TARGET DISK ADDRESS.
```

If you want the program on the system disk, you respond

```
190
```

You must have this disk accessed as your read/write A-disk. When the GENERATE EXEC is finished, it issues the message

```
'IPL DIR A1' CREATED
```

Next, to generate the CMS DIRECT command, use the CMSGEND EXEC procedure:

```
cmsgend direct
```

If you want to punch a real card deck, to keep available for standalone operations in the machine room, you can punch the file IPL DIR A1 (with the NOHEADER option), or use the GENERATE EXEC with the SRVCPGM operand:

```
generate srvcpgm
```

While you issue the above command, all of the standalone service programs are punched onto cards.

Figure 40 lists the services programs and indicates the programs and procedures you can use for each.

Program / Update	DASD Dump/ Restore	Directory	Format/ Allocate	IBCDASDI	NCPDUMP
CP module name (use VMFASM to update)	DMKDDR	DMKDIR	DMKFMT	IBCDASDI	DMKRND
CMS command name (use CMSGEND to Generate	DDR	DIRECT	--	--	NCPDUMP
CMS disk file (use GENERATE IPLDECK to create)	IPL DDR	IPL DIR	IPL FMT	--	--
Standalone card deck (use GENERATE SRVCPGM to punch)	IPL DDR	IPL DIR	IPL FMT	IPL IBCDASDI	--

Figure 40. Updating Service Programs

Updating the Loader Program

The loader (DMKLD00E) is a service program that loads a CP, CMS, or RSCS nucleus, and produces a load map. The loader loads the object modules (text files) supplied with it, resolves CCW addresses, and resolves address constants.

If an overlay error occurs while the loader is executing, define a larger virtual machine and reload the system.

The loader is distributed with the following default I/O addresses:

Console=009
Printer=00E

If there is no printer at address 00E, the load map is printed at the first printer that causes an interrupt (not-ready to ready sequence).

| Note: The loader does not support display mode consoles. If an IPL is attempted, wait state code 'FFF' is entered if the printer address is not 00E. To circumvent this occurrence, reconfigure the console to printer-keyboard mode or use the following procedure to correct the printer address.

You can override these addresses by placing a control card between the last card of the loader and the first card of the text decks. The format of the control card is:

Column	Contents
1	12-2-9 multipunch (X'02')
2-4	DEV
5	blank
6-13	PRNT=cuu (cuu is the printer address)
14	blank or comma
15-22	TYPW=cuu(cuu is the console address)
23-72	blank

The other loader control statements are the same as the loader control statements described with the CMS LOAD command in the VM/370 CMS Command and Macro Reference.

The loader is self-relocating, that is, it is initially loaded at address 2000 (hexadecimal); it then relocates itself at the top of storage. (For example, if the size of the loader is 10K, and the real storage size of the CPU is 512K, the loader occupies the area of storage between 502K and 512K.) As the loader needs free storage to perform its operations, it extends downward through storage.

The object modules being loaded must not overlay either the loader or any address between 0 and 100 (hexadecimal). The object modules are loaded into storage in a positive direction (that is, upward through storage). Before the loader actually loads an object module, it checks that the module does not overlay the loader's free storage. If an object module would overlay the loader, the loader terminates. You must close the printer to get the load map printed. The last line of the load map indicates the overlay area, if there was one.

If the loader terminates the operation, a wait condition is indicated in the instruction counter. If the instruction counter contains X'999999', indicating an SVC wait state, the interruption code (the third and fourth bytes of the supervisor old PSW) indicate the error condition. For a detailed explanation of the error conditions and interruption codes, see *VM/370 System Messages*.

The Load Map

The load map (the output of the loader) indicates:

- The size of each object module and the address where it is loaded. For example:

```
DMKMCH AT 00E68   MODULE SIZE IS 000C00
```

- The end of the resident nucleus with the message:

```
***                                ***
        END OF VM/370 RESIDENT NUCLEUS
***                                ***
```

  The CP modules that precede this message in the load map are not pageable; the CP modules that follow this message are pageable.

- When a Set Page Boundary (SPB) card has been inserted. If an object module cannot fit on the same page as the object module(s) loaded before it, the loader inserts an SPB card to force the modules to be loaded at a page boundary. This procedure ensures that object modules do not cross page boundaries.

- Two external names may be listed as undefined on the load map. If the virtual=real option is not specified, the external name DMKSLC is listed as undefined. If a 3704/3705 control program entry is not defined in the system name table (via the NAMENCP macro), the DMKRNTBL external name is undefined.

Generating a New Loader

The loader service program, in its executable form, has a filetype of LOADER. Whenever you assemble a new copy of DMKLD00E, you must convert the resulting text file to a loader file. If there is a virtual punch at address 00D and a virtual reader at address 00C, the procedure for generating a new loader is:

Step 1. Assemble the New Loader

Update and assemble DMKLD00E. The output from this assembly is DMKLD00E TEXT.

Step 2. Punch a Copy of the Old Loader

Spool the punch continuously and punch a copy of the old loader.

```
spool 00d * cont
punch dmkld00e loader (noh
```

Step 3. Punch a Copy of the New Loader Text File

Punch a copy of the newly assembled loader, then close the punch. When the punch is closed the two files (DMKLD00E LOADER and DMKLD00E TEXT) are sent to your reader. The commands to punch the new loader text file and close the punch file are:

```
punch dmkld00e text (noh
spool 00d * nocont close
```

Step 4. Load the New Loader

IPL your virtual reader to read the old version of the loader (DMKLD00E LOADER) into your virtual machine. Then the old loader reads the new loader text file into your virtual machine and creates the new loader file.

```
ipl 00c clear
```

When the IPL is complete, the message

```
DMKDSP450W CP ENTERED; DISABLED WAIT PSW
```

is issued. The instruction address in the disabled wait PSW is X'404040'.

Step 5. Punch a Copy of the New Loader (Executable Form)

Close the punch to punch a copy of the new loader, which was created in Step 4. Also close the reader and printer.

```
close 00d
close 00c
close 00e
```

Step 6. Name the New Loader DMKLD00E LOADER

IPL CMS, access a read/write disk as your A-disk, and read the file you punched in Step 5. Name this file DMKLD00E LOADER; this replaces the original DMKLD00E LOADER file with the new one.

```
ipl cms
access 191 a
read dmkld00e loader
```

Note: Save a copy of the original DMKLD00E LOADER file before you replace it with the updated loader.

EXEC Procedures and Command Format Summaries

The command formats, options, and operands for each of the updating EXEC and command procedures are described next, in alphabetical order.

ASMGEND

Use the ASMGEND EXEC procedure to build the system assembler and to create the associated auxiliary directory. ASMGEND loads the text decks for the assembler in the correct overlay structure and produces a load map. The format of the ASMGEND command is:

ASMGEND	

Responses

The ASMGEND EXEC procedure displays the following status and error messages:

ENTER TARGET DISK MODE FOR ASSEMBLE MODULES
DEFAULTS TO S-DISK IF NONE ENTERED

You enter the mode letter of the disk containing the modules referred to from the auxiliary directory. If you enter a mode letter, ASMGEND uses that mode letter as the "targetmode" operand of the GENDIRT command when it creates the auxiliary directory. If you do not specify a mode letter, S is used.

ASMGEND XF GEND COMPLETE

This message indicates that the system assembler and its associated auxiliary directory are generated successfully.

ASMGEND XF GEND FAILED

This message indicates that the system assembler text files were not loaded successfully.

| Usage Note:

| ASMGEND creates an assemble module on the A-disk.

CMSGEND

Use the CMSGEND EXEC procedure to generate a new CMS module from a text file and place the new CMS module on the A-disk.

The format of the CMSGEND EXEC command is:

```
┌──────────────────────────────────────────────────────────────────┐
│           │      ┌          ┐                                     │
│ CMSGEND   │ fn   │ CTLCMS   │                                     │
│           │      │ CTLALL   │                                     │
│           │      │ NOCLEAR  │                                     │
│           │      │ MAP      │                                     │
│           │      │ NOINV    │                                     │
│           │      └          ┘                                     │
└──────────────────────────────────────────────────────────────────┘
```

where:

fn is the filename of the CMS module that is to be generated by the CMSGEND EXEC. Only one filename may be specified in the CMSGEND command line.

The filenames that may be specified in the CMSGEND command are any disk-resident CMS commands and service programs.

Note: You can also use the CMSGEND EXEC to regenerate the ASSEMBLE command when you move the CMS system disk. When you specify ASSEMBLE, CMSGEND prompts you to enter a disk mode letter so it can refresh the assembler's auxiliary directory. Use the ASMGEND EXEC procedure if you are updating the assembler.

CTLCMS displays each CMS command as it is executed in the CMSGEND EXEC procedure. This is equivalent to the EXEC statement &CONTROL CMS.

CTLALL displays every executable statement as it is executed in the CMSGEND EXEC procedure. This is equivalent to the EXEC statement &CONTROL ALL.

NOCLEAR specifies that the CLEAR option is not to be issued when CMSGEND invokes the LOAD command.

MAP specifies that the NOMAP option is not to be issued when CMSGEND invokes the GENMOD command.

NOINV issues the NOINV option when CMSGEND invokes the LOAD command; this suppresses the displaying of invalid cards at the terminal. If the text deck was created with the VMFASM EXEC, it may contain update listing information; these records are displayed during the loading process unless you specify NOINV.

How CMSGEND Works

CMSGEND keeps a list of the CMS disk-resident modules, the filenames of the text files used to create them, and any special attributes required to generate them. For example, the RELEASE command must be generated with the ORIGIN TRANS and the SYSTEM options. It is composed of the text files DMSARE and DMSALU. To generate a new RELEASE module, you issue:

```
cmsgend release
```

you may receive messages such as the following:

```
*** CURRENT STATUS:
FILE ' RELEASE MODULE A2' EXISTS
FILE ' RELEASE MODOLD A1' DOES NOT EXIST

*** LOADING:
 INVALID CARD - X9999DMS - (PTF description)
        .
        .
        .
 DMSARE    SD 00E000
 DMSALU    SD 00E3B8
 RELUFD       00E3B8
 SORTEST      00E626
 END$RELU     00EE48

*** RESULTS:
FILE ' RELEASE MODULE A2' RENAMED TO ' RELEASE MODOLD A1'
' RELEASE MODULE A2' CREATED FROM TEXT DECK ( S ) DMSARE DMSALU
WITH ATTRIBUTES TRANS SYSTEM NOMAP
```

Responses

The CMSGEND EXEC procedure displays status and error messages.

```
*** CURRENT STATUS:
┌                          ┐
|'fn MODULE A2' EXISTS      |
|'fn MODULE A2' DOES NOT EXIST|
└                          ┘

┌                          ┐
|'fn MODOLD A1' EXISTS      |
|'fn MODOLD A1' DOES NOT EXIST|
└                          ┘
```

This message indicates whether a generated module already exists.

```
*** LOADING:
```

This message indicates that CMSGEND is loading the text decks.

```
*** (UNDEF. NAMES NORMAL FOR EDINIT)
*** NOW WE HAVE A SECOND PASS FOR EDINIT MODULE.
```

These messages indicate that the EDIT command requires two passes to resolve undefined names.

```
*** RESULTS:
['fn MODOLD A1' WAS ERASED]
['fn MODULE A2' RENAMED TO 'fn MODOLD A1']
 'fn MODULE A2' CREATED FROM TEXT DECK(S) ...
  WITH ATTRIBUTES ...
```

These messages indicate which existing modules were erased and renamed, which text files were used to create the new module, and the attributes used to create the module.

```
ENTER GENDIRT TARGET DISK MODE LETTER
( NULL LINE DEFAULTS TO 'S' DISK )
```

This message is issued when you specify ASSEMBLE. You should enter the mode letter of the disk that contains the system assembler. This letter is used as the target disk mode address for the GENDIRT command.

Error Messages

ERROR OCCURRED. CMSGEND STOPS.

This message indicates that an error occurred and that CMSGEND is terminated.

INVALID ARGUMENT fn

This message indicates an invalid filename was specified on the command line.

Note: Text files must have a filetype of TEXT. For example, after you have updated an object module using VMFASM, the most recent object file has a filetype such as TXTLOCAL. To use that text file here, you must rename it to a filetype of TEXT. If there is currently a text file on the system disk, you may want to rename it too, so that your updated text file (which may reside on another disk) is the one that is loaded.

GENERATE

GENERATE is a multipurpose EXEC used to generate VM/370. You may also use it to perform updating and maintenance of

- CP, CMS, and RSCS
- VM/370 service programs

You may also use it to regenerate your VM/370 operand after updating

- The directory
- The real I/O configuration (DMKRIO)
- The system control file (DMKSYS)
- The forms control buffer load (DMKFCB)
- The system name table (DMKSNT)

Instructions for coding the control statements and macros that define your VM/370 directory, and DMKRIO, DMKSYS, and DMKSNT files are in "Part 2. Defining Your VM/370 System." Instructions for coding a new DMKFCB module are in the VM/370 System Programmer's Guide.

The GENERATE EXEC procedure assumes the following:

```
Virtual RSCS/IPCS tape address = 181
Virtual CP nucleus tape address = 182
Virtual address of CMS build area = 190
Virtual address of CP and RSCS build area = 194
Virtual card reader = 00C
```

The format of the GENERATE EXEC command is:

```
GENERATE | ⎧ VM370
         | ⎪ DIRECT [ONLY]
         | ⎪ DMKRIO [ONLY]
         | ⎪ DMKSYS [ONLY]
         | ⎪ DMKFCB [ONLY]
         | ⎪ DMKSNT [ONLY]
         | ⎨ IPLDECK
         | ⎪ SRVCPGM
         | ⎪ ┌    ┐
         | ⎪ |CP  | NUCLEUS [NOLOAD]
         | ⎪ |CMS |
         | ⎪ └    ┘
         | ⎩ RSCS [BUILD]
```

where:

VM370 builds a new VM/370 directory, assembles DMKRIO and DMKSYS (also assembles DMKFCB and DMKSNT, if they are supplied), writes the CP nucleus to tape and optionally loads it.

You must have the appropriate files in your virtual card reader at address 00C. Place the following in your card reader, in the order shown:

```
:READ fn DIRECT
(Directory program control statements)
:READ DMKRIO ASSEMBLE
```

(real I/O configuration macros)
:READ DMKSYS ASSEMBLE
(CP system control macros)

where fn is the filename of your VM/370 directory. GENERATE prompts you for the filename of your directory. Optionally, you can place new versions of DMKFCB and/or DMKSNT in your card reader following the DMKSYS macros:

:READ DMKFCB ASSEMBLE
(forms control buffer load macros)
:READ DMKSNT ASSEMBLE
(system name table macros)

GENERATE reads the files in the reader. It invokes the Directory program to build the VM/370 directory and invokes the VMFASM EXEC procedure to assemble all the ASSEMBLE files in the reader. VMFASM is invoked with the current IBM-supplied control file to ensure that the proper macro libraries are available when the modules are assembled and to assign the correct filetype. Then, GENERATE invokes the VMFLOAD EXEC procedure to place all the CP object modules on tape in the correct order. If an error is detected during any of these processing steps, GENERATE terminates at the end of that step.

For a CP nucleus without a virtual=real area, GENERATE loads the tape, thus loading the newly generated CP nucleus. For a CP nucleus with a virtual=real area, GENERATE writes the nucleus to tape and exits. You are instructed to shutdown the system. Then you can IPL the tape on a real machine or on a virtual machine that has enough virtual storage. The "Specifying a Virtual=Real Machine" section of Part 1 tells you how much virtual storage you need.

DIRECT [ONLY]
builds a new VM/370 directory.

If you do not specify ONLY, GENERATE executes exactly as if you specified GENERATE VM370.

If you specify ONLY, only the VM/370 directory is built. You must place the Directory program control statements in your virtual card reader at address 00C, as follows:

:READ fn DIRECT
(Directory program control statements)

where fn is the filename of your VM/370 directory. GENERATE prompts you for the filename of your directory file.

DMKRIO [ONLY]
assembles the real I/O configuration file (DMKRIO) by invoking VMFASM.

If you do not specify ONLY, the GENERATE EXEC procedure executes just as if you specified GENERATE VM370 (except that it does not build a new VM/370 directory). Consequently, you should follow the directions for issuing GENERATE VM370, except do not place the Directory program control statement (nor the corresponding :READ statement) in your card reader.

If you do specify ONLY, only the DMKRIO module is assembled.

You must place the real I/O configuration macros in your virtual card reader at 00C, as follows:

:READ DMKRIO ASSEMBLE
(real I/O configuration macros)

DMKSYS [ONLY]
assembles the CP system control file (DMKSYS) by invoking VMFASM.

If you do not specify ONLY, the GENERATE EXEC procedure executes just as if you specified GENERATE VM370 (except that it does not build a new VM/370 directory and does not assemble a new DMKRIO). Consequently, you should follow the directions for issuing GENERATE VM370, except do not place the Directory program control statements or real I/O configuration macros (nor their corresponding :READ statements) in your card reader.

If you do specify ONLY, only the DMKSYS module is assembled. You must place the CP system control macros in your virtual card reader at 00C, as follows:

:READ DMKSYS ASSEMBLE
(CP system control macros)

DMKSNT [ONLY]
assembles the system name table (DMKSNT) by invoking VMFASM.

If you do not specify ONLY, the GENERATE EXEC procedure goes on to invoke the VMFLOAD EXEC procedure to place all the CP object modules on tape. If no error occurs, and if the CP nucleus does not have a virtual=real area, GENERATE then loads the tape, thus loading the CP nucleus (with the new version of DMKSNT).

If you do specify ONLY, the DMKSNT module is assembled, but the CP nucleus is not written to tape and is not loaded.

To assemble DMKSNT, you must place the system name table macros in your virtual card reader, at address 00C, as follows:

:READ DMKSNT ASSEMBLE
(system name table macros)

DMKFCB [ONLY]
assembles the forms control buffer load (DMKFCB) by invoking VMFASM.

If you do not specify ONLY, the GENERATE EXEC procedure executes just as if you specified GENERATE VM370 (except it does not build a new VM/370 directory and does not assemble new DMKRIO or DMKSYS modules. Consequently, you should follow the directions for issuing GENERATE VM370; except do not place the Directory control statements, real I/O configuration macros, or CP system control macros (nor their corresponding :READ statements) in your card reader.

If you want to add or replace the IBM supplied FCB image, you must place your FCB macro in the IBM supplied DMKFCB assemble file; after label DMKFCBLD.

If you specify ONLY, only the DMKFCB module is assembled. You must place the forms control buffer load macros in your virtual card reader at 00C, as follows:

```
:READ DMKFCB ASSEMBLE
(forms control buffer load macros)
```

IPLDECK creates the standalone service programs on disk from their associated object modules (text decks). These files must be on the CMS system disk (S-disk). You are prompted to enter the names of the service programs you wish to generate. You can respond ALL, DDR, DIR, or FMT. If you respond ALL, the DASD Dump Restore, Directory, and Format/Allocate standalone programs are built on disk.

SRVCPGM punches all the standalone service programs (DMKDIR, DMKDDR, DMKFMT, and IBCDASDI).

```
┌   ┐
|CP | NUCLEUS [NOLOAD]
|CMS|
└   ┘
```

generates the CP or CMS nucleus. If you specify CP NUCLEUS, the CP nucleus is loaded onto tape.

For a CP nucleus without a virtual=real area, GENERATE loads the tape, thus loading the newly generated CP nucleus. For a CP nucleus with a virtual=real area, GENERATE writes the nucleus to tape and exits. You are instructed to shutdown the system. Then you can IPL the tape on a real machine or on a virtual machine that has enough virtual storage. The "Specifying a Virtual=Real Machine" section of Part 1 tells you how much virtual storage you need.

Attached Processor support will be included in the nucleus, if desired. If you specify CP NUCLEUS NOLOAD, the tape is created with the new nucleus but the disk is not loaded.

If you specify CMS NUCLEUS, a card-image deck is created and placed in the virtual card reader. GENERATE issues a prompting message to see if you want a card-image copy of the CMS nucleus put on disk. If you respond "yes," GENERATE writes a copy of the CMS nucleus on disk, and loads the nucleus. The card-image copy of the CMS nucleus is a file (CMSNUC NUCLEUS) that can later be loaded to create a CMS nucleus. If you specify CMS NUCLEUS NOLOAD, the new nucleus remains in the virtual card reader.

RSCS [BUILD]

Prepares the RSCS build area and, optionally, builds the RSCS nucleus, as described in the section "Generating and Installing RSCS."

Responses and Error Messages

The GENERATE EXEC procedure issues many descriptive responses, most of which are shown in the system generation procedures in Part 3. Generating VM/370 (CP, CMS, RSCS, and IPCS).

VMFASM

Use the VMFASM EXEC procedure to update a specified source file according to entries in a control file, and to assemble the updated source file. VMFASM invokes the CMS UPDATE command. The format of the VMFASM command is:

```
┌──────────────────────────────────────────────────────────────┐
| VMFASM | fn ctlfile [(options...[)]]                         |
|        |                                                     |
|        | Options:                                            |
|        | ┌      ┐ ┌       ┐ ┌       ┐                        |
|        | |DISK  | |TERM   | |LIST   |                        |
|        | |PRINT | |NOTERM | |NOLIST |                        |
|        | └      ┘ └       ┘ └       ┘                        |
|        | ┌       ┐ ┌       ┐                                 |
|        | |DECK   | |RENT   | [EXP] [XREF]                    |
|        | |NODECK | |NORENT |                                 |
|        | └       ┘ └       ┘                                 |
└──────────────────────────────────────────────────────────────┘
```

where:

fn — is the filename of the source file to be updated. It must have a filetype of ASSEMBLE.

ctlfile — is the filename of the control file. The control file must have a filetype of CNTRL.

Options: VMFASM only accepts the nondefaulted options. All other assembler options entered are ignored and the defaults are used.

DISK — places the listing file on a virtual disk.

PRINT — writes the listing file to the printer.

TERM — writes the diagnostic information on the SYSTERM data set. The diagnostic information consists of the diagnosed statement followed by the error message issued.

NOTERM — suppresses the TERM option.

LIST — produces an assembler listing.

NOLIST — does not produce an assembler listing.

DECK — writes an object module on the device specified on the FILEDEF statement for PUNCH.

NODECK — suppresses the DECK option.

RENT — checks the program for a possible violation of program reenterability. Code that makes the program nonreenterable is identified by an error message.

NORENT — suppresses the RENT option.

EXP — expands printing of certain macros which check for the SUP parameter issued via the SYSPARM option of the ASSEMBLE command. The default is SUP.

XREF causes the XREF(FULL) option to be invoked when VMFASM invokes the assembler. The default for VMFASM is XREF(SHORT).

How VMFASM Works

The steps taken by the VMFASM EXEC are summarized below.

1. The VMFASM EXEC calls the UPDATE command with the CTL, STK, and PRINT options.

   UPDATE uses the control file (ctlfile CNTRL) to update the assembler language source file. The new file is named $fn ASSEMBLE.

   UPDATE stacks information from the control file in the console stack, and prints the update log file.

2. Using the library list from the MACS record in the control file, VMFASM issues a GLOBAL MACLIB command.

3. The updated source file, $fn ASSEMBLE, is assembled using the options indicated on the VMFASM command line.

4. The output text deck from the assembly, $fn TEXT, is concatenated with the UPDATES file so that the text deck contains a history of update activity.

5. Using the update level identifier from the control file (the identifier of the most recent update that was found and applied is stacked by the UPDATE command), VMFASM determines how to rename $fn TEXT.

   If the update level identifier is TEXT, the text deck is renamed fn TEXT.

   If the update level identifier is anything other than TEXT, the text deck is renamed fn TXTxxxxx, where xxxxx is the one- to five-character update level identifier and fn TEXT is erased from the A-disk.

6. Temporary files ($fn ASSEMBLE, fn UPDATES, and fn ctlfile) are erased.

The input and output files used by VMFASM are summarized below.

DISK INPUT FILES:

fn ASSEMBLE	Assembler Language source
ctlfile CNTRL	control file

MACLIBS, auxiliary control files (AUXxxxxx), and miscellaneous update files.

DISK OUTPUT FILES:

fn {TEXT / TXTxxxxx}	object deck, named according to the update level identifier in the control file

This file also contains data from the UPDATES file, together with date and time information.

PRINTER OUTPUT FILES

$fn LISTING   Assembler listing (if PRINT option is in effect)

This file also contains data from the update log file (fn UPDLOG), describing the updates applied to the source file.

Note: If the object modules created have a filetype of "TXTxxxx" and are to be used with one of the GEN EXECs (CMSGEND, DOSGEN, VSAMGEN, etc.), they must be renamed with a filetype of "TEXT".

Responses

The UPDATE command issues the message DMSUPD178I to indicate each of the update files being applied.

ASMBLING fn
indicates that the assembly is going to begin. If you specified any assembler option on the VMFASM command line, the options used are also displayed.

fn {TEXT | TXTxxxxx} CREATED

indicates the filename and filetype of the text deck.

Error Messages

*** fn ASSEMBLE NOT FOUND ***
The source file could not be located.

*** ctlfile CNTRL NOT FOUND ***
The control file could not be located.

*** ERROR UPDATING fn ***
An error occurred during UPDATE command processing.

*** ERROR ASMBLING fn ***
An assembler error occurred.

*** NO TEXT FOR fn ***
No TEXT file was produced, because of assembler errors.

VMFLOAD

Use the VMFLOAD EXEC procedure to generate a new CP, CMS, or RSCS nucleus. The VMFLOAD program uses two files, a loadlist EXEC file and a control file, to produce a punch file that has several object modules. VMFLOAD requires a virtual machine with 320K.

The format of the VMFLOAD command is:

VMFLOAD	loadlist ctlfile

where:

loadlist is the filename of an EXEC file that contains the names of object modules in the order in which they are to reside in the complete load file for the nucleus. For example:

```
&CONTROL OFF
&1 &2 fn [ft]
&1 &2 fn [ft]
  .
  .
  .
```

where fn and optionally ft, are the filename and filetype of an object module to be punched. The object modules are punched in the order specified, beginning at the top of the loadlist EXEC. If a filetype is specified, VMFLOAD searches for that specific file, and, if it finds it, punches it without a header card.

If the filetype is not specified in the loadlist, VMFLOAD uses the control file to determine which object module is the highest level object module available. VMFLOAD searches the control file from top to bottom. When it finds the appropriate object module, VMFLOAD punches it.

ctlfile is the filename of the control file. This is usually the same control file used to apply updates to modules via the VMFASM or UPDATE commands. This file identifies the highest level object module available, if the filetype is not specified in the loadlist.

How VMFLOAD Works

1. Before the the files specified in the loadlist are punched, VMFLOAD issues a SPOOL PCH CONT command to ensure that the punched files appear as one deck. You may wish to specify SPOOL PCH TO userid before you invoke VMFLOAD to transfer the punched output as a file to your own (or another) virtual machine as a reader file. If you want to perform any additional controls, you should write an EXEC procedure to perform the control and invoke VMFLOAD from that EXEC procedure.

2. For each entry in the loadlist that does not specify a filetype VMFLOAD searches the control file to determine the filetype of the object module.

The filetypes are based on the update level identifiers in the control file. These are the identifiers used by the VMFASM to assign filetypes to object decks. Remember that updates applied to source files are applied from the bottom of the file towards the top. Therefore, VMFLOAD searches the control file from the top towards the bottom to locate the most recent update level.

For example, if a control file contains the records and control file, you would issue:

```
TEXT MACS DMKMAC
LOCAL FIX1
SPEC  AUX1111
PTF   R12765DK
IBM1  AUXRn0
```

Then for each entry in the loadlist, the VMFLOAD search order is:

- fn TXTLOCAL
- fn TXTSPEC
- fn TXTIBM1

As soon as VMFLOAD locates a file, it punches it, then continues processing the next entry in the loadlist. If none of the above filetypes exist for the loadlist entry, VMFLOAD searches for filename TEXT. If there is no TEXT file, VMFLOAD displays a message and continues processing with the next entry in the loadlist.

Note: VMFLOAD ignores records that have an update level identifier of PTF, and so searches for the next lowest level identifier when determining the filetypes of object modules to punch.

3. When all the object modules are punched, VMFLOAD issues the commands

```
SPOOL PUNCH NOCONT
CLOSE PUNCH
```

If you issued the command

```
spool punch to *
```

prior to invoking VMFLOAD, the completed load deck is placed in your virtual card reader.

DISK INPUT FILES:

loadlist EXEC	contains the filenames, and optionally filetypes, of the object modules to be punched.
DMKLDOOE LOADER	the loader, which should be the 1st entry in the loadlist EXEC

object modules with filetypes of TEXT or TXTxxxxx, where xxxxx is the update level identifier in a control file, used by VMFASM to name the object module.

PUNCH OUTPUT FILES

load deck	punched to your virtual machine

Notes

1. The distributed system uses the following loadlists:

Loadlist	Usage
APLOAD	CP nucleus without V=R for the attached processor
APVRLOAD	CP nucleus with V=R for the attached processor
CPLOAD EXEC	CP nucleus without V=R for uniprocessor
VRLOAD EXEC	CP nucleus with V=R for uniprocessor
CMSLOAD EXEC	CMS nucleus
DMTLOAD EXEC	RSCS nucleus

For example, to punch a new CP nucleus with the distributed loadlist

```
vmfload cpload DMKRn0
```

The GENERATE EXEC and the VMSERV EXEC uses VMFLOAD to generate a new CP nucleus.

2. After you have punched a new nucleus with VMFLOAD, you can either move the nucleus to tape, using the MOVEFILE command, or, if the nucleus is in your virtual card reader, you can IPL it:

```
ipl 00c
```

When you IPL the virtual card reader, the loader is read first, and it loads the rest of the object modules. If the loader is successful, the nucleus is written on disk, and the load map is spooled to the virtual printer. If you want to preserve a disk copy of the load map, you should spool your printer to your virtual card reader, then read the file onto disk.

Responses

SYSTEM LOAD DECK COMPLETE

This message is displayed when all the files in the loadlist have been punched.

Error Messages

INSUFFICIENT OR INVALID ARGUMENTS
The command line was incorrectly entered.

NO CONTROL FILE
The control file could not be located.

ERROR IN CONTROL FILE
The control file contains an invalid record.

NO LOAD LIST
The loadlist could not be located.

ERROR IN LOAD LIST
The loadlist contains an invalid record.

fn ft NOT FOUND
No text file was found.

ERROR ON PUNCH
An error occurred punching a file.

CP LOADLIST REQUIREMENTS

The CPLOAD loadlist EXEC contains a list of CP modules that is used by the VMFLOAD procedures to punch the text decks for the CP system. All modules following DMKCPE in the list are pageable CP modules. Each 4K page in this area may contain one or more modules. Pageable modules must not span the 4K page boundaries. The module grouping is governed by SPB (Set Page Boundary) cards. An SPB card is a loader control card that forces the loader to start this module at the next higher 4K boundary. If more than one module is to be contained in a 4K page, only the first is preceded by an SPB card.

The loader inserts SPB cards automatically where they are needed; you need not insert SPB cards.

The position of two modules in the loadlist is critical. All modules following DMKCPE must be reenterable and must not contain any address constants referring to anything in the pageable CP area. DMKCKP must be the last module in the loadlist.

VMFMAC

Use the VMFMAC EXEC procedure to update macro libraries. It invokes the CMS UPDATE command to update specified copy or macro files, according to entries in a control file, and then builds a new macro library from the resulting new versions of those files.

The format of the VMFMAC command is:

VMFMAC	libname ctlfile

where:

libname is the filename of the macro library to be updated, and of the EXEC file that contains the names of the library members. The entries in libname EXEC must be in the following format:

```
&1 &2 fn1
&1 &2 fn2
 .
 .
 .
```

where fn1, fn2, and so on, are filenames of macro or copy files to be updated and included in the macro library, which must have a filetype of MACLIB.

ctlfile is the filename of a control file to be used to apply the updates. The filetype must be CNTRL. The filenames used by VM/370 are DMKRn0, DMSRn0, DMSMn0, DMTRn0, and DMMRn0.

How VMFMAC Works

The steps taken by VMFMAC are summarized below.

1. VMFMAC locates libname EXEC and the control file. It also erases any existing files named NEWMAC MACLIB and NEWMAC COPY. Then VMFMAC begins reading the macro or copy filenames from the EXEC, beginning at the bottom.

2. For each entry in the libname EXEC, VMFMAC:

   -- Invokes the UPDATE command with the CTL option to apply the updates specified in the control file.

   -- Adds the updated macro or copy file ($filename MACRO or $filename COPY) to the macro library NEWMAC MACLIB.

   -- Adds the UPDATES file created by the UPDATE command to the file NEWMAC COPY.

   -- Erases $filename MACRO or $filename COPY, and filename UPDATES.

3. If there are no update files for a macro or copy file specified in libname EXEC, the macro or copy file is added to NEWMAC MACLIB in its current form. NEWMAC COPY, containing a history of updates applied by VMFMAC, is added to NEWMAC MACLIB.

4. If no errors occur during the procedure, then when all the macros have been added to NEWMAC MACLIB, NEWMAC MACLIB is renamed libname MACLIB. libname MACLIB, if it exists, is erased.

   If errors occur during the VMFMAC EXEC procedure (for example, if a MACRO or a COPY file is not found) libname MACLIB is not erased, and the updated macro library retains the name NEWMAC MACLIB.

DISK INPUT FILES:

libname EXEC	contains a list of macro a copy file to be updated and/or included in libname MACLIB.
ctlfile CNTRL	is the control file used by the UPDATE command.

MACRO and COPY files to be updated and/or included in the macro library, plus miscellaneous auxiliary control files and update files.

DISK OUTPUT FILES

libname MACLIB	is the updated macro library.
libname COPY	contains the UPDATES files produced by UPDATE command processing.

PRINTER OUTPUT FILES:

The printer is spooled with the CONT option, so that when VMFMAC completes, the printer file contains:

- A copy of the control files
- For each updated macro or copy file, the update log file produced by the UPDATE command.
- A copy of each macro or copy file is the macro library
- The libname COPY file, which contains the accumulated UPDATES files created by the UPDATE command.

Notes

1. When files with MACRO filetypes are added to a MACLIB, the membername is taken from macro prototype statement. When files with COPY filetypes are added to a MACLIB, the membername is taken from the filename of the COPY file, (which will be $filename if updates were found, otherwise filename) unless you include a *COPY statement as the first record in the file, in the format:

   *COPY membername

   Then, the MACLIB directory uses membername to name the copy file.

2. If errors occur during VMFMAC processing, consult the libname COPY file printed by VMFMAC. If you can correct the errors involving one or two macro or copy files, you can add these members to NEWMAC MACLIB using the MACLIB command, then rename NEWMAC MACLIB to libname MACLIB after erasing the current libname MACLIB.

<u>Responses</u>

The UPDATE command issues the message DMSUPD178I to inform you of the updates being applied to each macro or copy file. If no updates are found, message DMSUPD181E is issued.

```
fname {COPY } ADDED.
      {MACRO}
```

indicates that the specified macro or copy file has been added to the macro library.

libname COPY ADDED.

indicates that libname COPY, containing the update history, of the MACLIB, has been added.

<u>Error Messages</u>

*** TYPE 'VMFMAC LIBNAME CTL' ***

This message indicates that the command line did not have two operands.

*** libname EXEC NOT FOUND ***

VMFMAC could not locate the EXEC file associated with the macro library.

*** ctlfile CNTRL NOT FOUND ***

VMFMAC could not locate the control file.

*** fn COPY or MACRO NOT FOUND ***

A library member named in libname EXEC could not be located.

```
*** ERRORS UPDATING fn {COPY } ***
                       {MACRO}
          {COPY }
    fname {MACRO} NOT INCLUDED IN MACLIB
```

This message indicates an UPDATE command error occurred for the member, and the file was not written into the MACLIB.

DUE TO PREVIOUS ERRORS, THE RESULT OF THIS MACLIB BUILD
IS CALLED 'NEWMAC MACLIB', libname MACLIB HAS
NOT BEEN REPLACED

One or more errors were encountered, and you must correct them and create the MACLIB yourself.

VMSERV

VMSERV is an EXEC procedure that is included on the System Program Update Tape (PUT) to assist you in the application of IBM service updates to the VM/370 system. VMSERV directs the installation of maintenance and service updates to the components and program products that form your system. It is recommended that VMSERV be allowed to apply service sequentially. The individual service EXECs contained on the PUT are designed to apply service in the prescribed order.

The format of the VMSERV EXEC command is:

```
┌────────┬──────────────────────────────────────────────────────────────────┐
│        │                                                                  │
│        │ ┌                        ┐ ┌        ┐ ┌      ┐ ┌        ┐          │
│ VMSERV │ │RESTART [PROGID]        │ │SIPOE   │ │NOIPL │ │NOMEMO  │ [COMP]   │
│        │ │BUILD   [PROGID]        │ │NORESP  │ │EXIT  │ │SIPOMEM │          │
│        │ └                        ┘ └        ┘ └      ┘ └        ┘          │
│        │                                                                  │
└────────┴──────────────────────────────────────────────────────────────────┘
```

where:

RESTART	returns control to VMSERV after an IPL of a nucleus or any other event that causes VMSERV not to regain control.
BUILD	creates a new nucleus. This option assumes you have loaded all service from the system PUT.
PROGID	specifies the program number of the product you are servicing.
SIPOE	suppresses prompting messages related to specific service applications and staging area disk addresses.
NORESP	suppresses prompting messages for staging area disk addresses. You will receive prompts to offer service to each component and any applicable program products.
NOIPL	loads the service from the system PUT but does not create and (IPL) a nucleus for any component that is serviced. Because VMSERV does not normally lose control with this option, all of the service on the PUT is loaded at one time. You may then BUILD the desired nucleus after applying corrective PTF's or any user modifications.
EXIT	installs service sequentially up to the point where a nucleus would be created and loaded via IPL. VMSERV exits at this point.
NOMEMO	suppresses printing of the Memo-to-Users.
SIPOMEM	prints the SIPO/E user memos.
COMP	specifies the component that is intended to receive service. If BUILD is also specified, COMP is the nucleus of the component that you are building. The default is CP. You can also specify CMS, IPCS, or RSCS.

Usage Notes

1. PROGID is a positional parameter which is valid only if you specify RESTART or BUILD.

2. The parameters that are separated vertically are mutually exclusive. They cannot be specified at the same time. For example, specifying BUILD and NOIPL is not allowed.

3. If you are using SIPO (System Installation Productivity Option), refer to the VM System IPO Extended Planning Guide, Order No. GC20-1874.

HOW VMSERV WORKS

The PUT is composed of two or more virtual tapes stacked sequentially. Each virtual tape contains the service for one product. The first virtual tape on the PUT contains the service for the VM/370 System Control Program (SCP). The VMSERV EXEC is included in file one along with the SCP service installation EXEC (5749010) and the program level file.

Assuming you need a new map of the PUT, certain housekeeping functions are performed. VMSERV erases work EXECS PTFLVL and PTFMEMO, and issues a CP REWIND and VMFPLC2 FSF in an attempt to insure this is the correct tape. VMSERV then initializes the EXECs necessary to print the Memo-to-Users and user memos. The tape is rewound and VMFPLC2 looks for space on an A-disk on which a TAPE MAP can be written. A TAPE MAP is created for files one and two. Any selective-load EXECs that may have remained from the last PUT are erased and the new service EXECs are created. A line of data is printed to the console to indicate where you are in the procedure and the remaining virtual tapes are skipped. VMSERV offers to print the PUT DOCUMENT and the Memo-to-Users, reminding you to review the memos prior to installing service. You are also reminded of the NOIPL option of VMSERV, and since you probably have not reviewed the memos at this point, VMSERV exits. Specify RESTART to apply service to this product. VMSERV invokes the service (progid) EXEC with the applicable parameters that you entered when you invoked VMSERV.

After the service for all the desired products is installed, VMSERV types the SERVICE DISKMAP to the console to show the service status of all the products on the tape.

INITIALIZATION MESSAGES

Several messages may be issued to you during the application of service. The messages both normal and error that VMSERV issues are documented here.

VMSERV MUST BE RUNNING FROM YOUR 'C' DISK. PLEASE REVIEW THE INSTALLATION INSTRUCTIONS CONTAINED IN THE PLC'S COVER LETTER.

Return code: 4

Issued if the VMSERV EXEC is not running from the 'C' disk.

OPTION CONFLICT EXISTS. PLEASE RE-ENTER

Return code: 4

Issued for conflicting parameters issued on the command line: Such as 'BUILD' and 'NOIPL'. You should review the PUT DOCUMENT and re-enter the command properly.

UNKNOWN OR INCOMPLETE OPTION SPECIFIED. PLEASE RE-ENTER

Return code: 4

You have invoked VMSERV with misspelled or incorrect parameters. You must re-enter the command correctly.

THE SECOND PARAMETER ENTERED MUST BE THE PROGRAM NUMBER AT WHICH YOU WISH SERVICE INSTALLATION TO 'RESTART'. ENTER: VMSERV RESTART prog

Return code: 4

You have requested a 'RESTART' but have not indicated which service EXEC you wish to restart.

ENTER THE PROGRAM NUMBER YOU WISH TO BUILD OR 'EXIT'.

Return code: none - no exit

You have requested 'BUILD' but specified no progid to invoke. You should respond with the program number you wish to 'BUILD', or you may enter 'EXIT' to terminate VMSERV.

progid EXEC NOT FOUND. RE-ENTER IF TYPING ERROR. OR ENTER 'EXIT' IF YOU NEED TO RE-START.

Return code: none - no exit

You have incorrectly specified the progid to 'BUILD'. You may re-enter the progid or enter 'EXIT' to terminate VMSERV.

VMSERV WILL NOW MAP THE PUT. PROCESS WILL TAKE A FEW MINUTES

Return code: none - informational

Followed by the typing of the first line of the 'program level' files as they are read from the PUT. This file contains the official name of that particular (program) product.

YOU SHOULD REVIEW THE MEMOS-TO-USERS PRIOR TO INSTALLING SERVICE WOULD YOU LIKE THE 'MEMO(S)-TO-USERS' PRINTED? (-YES- | NO)

Return code: none - procedural question

Issued if the PUT was just mapped. It may be the initial invocation of VMSERV, or a 'RESTART' where no 'SERVICE DISKMAP' file was found on the 'C' disk.

INSURE (VIA THE MEMO-TO-USERS) THAT NO CHANGES HAVE OCCURRED TO THE MACROS AFFECTING DMKRIO, DMKSNT, and DMKSYS. IF THESE, OR ANY MODULE(S) NEED TO BE RE-ASSEMBLED FOR MACRO OR OTHER CHANGES, OR IF YOU WISH TO RECEIVE CONTROL AFTER THE MACLIBS AND OTHER SERVICE IS LOADED, USE THE 'NOIPL' OPTION WHEN YOU RE-INVOKE VMSERV TO INITIATE THE APPLICATION OF SERVICE.

THE MEMO-TO-USERS SHOULD BE REVIEWED PRIOR TO INSTALLING SERVICE WHEN YOU HAVE REVIEWED THE MEMOS AND ARE READY TO APPLY SERVICE, ENTER... VMSERV NOMEMO (USE THE 'NOIPL' OPTION IF NECESSARY)

Return code: 0 - normal termination

Issued after the printing of the MEMOs (if requested) to remind you about the 'NOIPL' option and to let you know that VMSERV is about to terminate and how you may restart.

SERVICE APPLICATION MESSAGES

YOU WILL BE REQUIRED TO REPLY TO QUESTIONS REGARDING THE APPLICATION OF SERVICE. THE ACCEPTABLE RESPONSES WILL BE SHOWN IN PARENTHESES. A RESPONSE SHOWN WITHIN DASHES: -yes-, IS THE DEFAULT, AND MAY BE SELECTED WITH A NULL RESPONSE.

Return code: none - informational message

MOUNT THE VM/370 SYSTEM PUT ON 181 AND RE-ISSUE VMSERV

Return code: 4

A 'CP REW 181' has resulted in an error. Perhaps the tape is is not attached.

THE TAPE ON YOUR VIRTUAL 181 IS NOT THE CORRECT TAPE. MOUNT THE VM/370 SYSTEM PUT ON 181 AND RE-ISSUE VMSERV

Return code: 4

The initial 'VMFPLC2' command has resulted in a tape error. Probably the wrong tape was mounted.

THIS SERVICE PROCEDURE NEEDS AN 'A' DISK TEMPORARILY ON WHICH TO WRITE 2 SMALL FILES. ENTER THE 'CUU' OF A DISK THAT MAY BE ACCESSED BY THE PROCEDURE. (-194- | CUU)

Return code: none - procedural question

Enter an address of a minidisk to which you now have write access. If there is a problem, the message will be reissued.

***** *COREQ* *** *COREQ* *** *COREQ* *** *COREQ* *****

Return code: none - informational

Issued if the service about to be installed has indicated that a corequisite condition should be indicated to you. Normally, the Memo-to-Users will detail exactly what corequisite conditions exist.

DO YOU WISH TO CONTINUE APPLYING SERVICE? (-YES- | NO)

Return code: none - procedural question

Issued at the end of the application of service for the product requested by 'RESTART'. If you respond 'YES' (the default), the 'RESTART' indicator is turned off, and the application of service continues as if 'RESTART' was never specified.

Return code: 0 - Normal termination

A 'NO' response will cause VMSERV to terminate.

progid HAD TROUBLE. YOU MUST RESTART VMSERV ... YOU ARE NOT FINISHED WITH THE INSTALLATION OF SERVICE.

Return code: 16 - error, tape position unknown

Issued when a service EXEC (progid) returns with a return code of 16 and 'EXIT' was not specified. This indicated the position of the tape is unknown and VMSERV must be restarted. Other errors may have been indicated to you such as a tape error or disk full condition.

RETURN CODES FROM SERVICE EXECS

0 Service complete. All load and 'BUILD' functions accomplished

4 Service loaded. System PUT is no longer required for any activity. a 'BUILD' may be required, depending on the product's packaging.

8 Service aborted. Some (or no) service was loaded from the tape. A 'RESTART' is required to apply service. The position of the tape is known and correct.

12 Unknown error. Unexpected error occurred. A 'RESTART' will have to be done. Service status and tape position are unknown.

16 SIPOE EXIT. The service EXEC proceeded to the point that a nucleus could have been created and then terminated. VMSERV will then EXIT (with a return code of 0).

XX Unknown. This is any unexpected return code from a service exec. Since the return code would have been '12' if the tape position was unknown, service installation will proceed.

Appendixes

A. Program Products, Installed User Programs, Field Developed Programs, and Emulators

B. Configuration Aid

C. CP/CMS Nucleus/Module/Segment Regeneration Requirements

D. Compatible Devices

E. Compatibility of VM/370 with CP-67/CMS

F. VM/370 Restrictions

G. A Sample EXEC Procedure for Copying DOS/VS Macros into a CMS MACLIB

Appendix A. Program Products, Installed User Programs, Field Developed Programs, and Emulators

VM/370 Assembler

VM/370 Assembler is distributed as a part of the VM/370 system and is required for installation and further support of the system. All necessary installation and support macros are provided in CMS libraries.

The Conversational Monitor System (CMS), the Remote Spooling Communications Subsystem (RSCS), the Control Program (CP), and the Interactive Problem Control System (IPCS) are components of VM/370 and are distributed with it. Certain other facilities mentioned in this publication are not part of VM/370, but can be separately ordered from IBM. These include: IBM System/360 and System/370 operating systems, IBM language processors and other program products, IBM Installed User Programs, and IBM Field Developed Programs. For more information, contact your IBM representative.

Program Products

The following is a list of IBM program products and their respective program numbers that VM/370 users have found useful. For installation and storage information, consult the publications that support these products.

DOS/VS Advanced Functions (Linkage Enhancements)	5746-XE2
DOS PL/I Optimizing Compiler	5736-PL1
DOS PL/I Resident Library	5736-LM4
DOS PL/I Transient Library	5736-LM5
DOS PL/I Optimizing Compiler and Libraries	5736-PL3
DOS/VS COBOL Compiler and Library	5746-CB1
DOS/VS COBOL Object Library	5746-LM4
OS Code and Go FORTRAN	5734-FO1
OS FORTRAN IV (G1)	5734-FO2
OS FORTRAN IV Library (Mod I)	5734-LM1
OS FORTRAN IV (H) Extended	5734-FO3
OS FORTRAN Library (Mod II)	5734-LM3
FORTRAN Interactive Debug	5734-FO5
OS/VS COBOL Compiler and Library	5740-CB1
OS/VS COBOL Library Only	5740-LM1

OS Full American National Standard COBOL Version 4 Compiler and Library	5734-CB2
OS Full American National Standard COBOL Version 4 Library	5734-LM2
OS COBOL Interactive Debug	5734-CB4
OS PL/I Optimizing Compiler	5734-PL1
OS PL/I Resident Library	5734-LM4
OS PL/I Transient Library	5734-LM5
OS PL/I Optimizing Compiler and Libraries	5734-PL3
OS PL/I Checkout Compiler	5734-PL2
VM Installation Productivity Option	5750-AA3
VS BASIC Processor	5748-XX1
VS APL	5748-AP1
VM/Pass-Through Facility	5748-RC1
Interactive Instructional System	5748-XX6
VM/370 Basic System Extensions	5748-XX8
VM/370 System Extensions	5748-XE1
VM/370 Remote Spooling Communications Subsystem Networking	5748-XP1
Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS)	5746-XX1
VM/Interactive Problem Control System Extension (VM/IPCS Extension)	5748-SA1
Document Composition Facility (Script/VS)	5748-XX9
VM/System Product (VM/SP)	5664-167

Installed User Programs

The following is a list of the Installed User Programs (IUPs) and their respective program numbers that VM/370 users have found useful. For installation and storage information, consult the publications that support these programs.

SCRIPT/370 Version 3	5796-PHL
McGill University System for Interactive Computing	5796-AJC
VS/REPACK	5794-PDZ
VM/370 System for Online Tape and Disk Libraries	5794-AGN
VS/370 Graphics Monitor	5794-PDT
VM/SGP Statistics Generating Package	5794-PDD
Assembler H/CMS Interface	5796-PEJ
APL Function Editor	5796-PGY
Display Editing System for CMS	5796-PJP
CMS EXEC Language Extensions	5796-PJA
FORTRAN Interactive Subroutine Library	5796-PHT
APL GPSS	5796-PJG
APL Data Interface	5796-PKA
APL Statistical Library	5796-PHW
APL Econometric Planning Language	5796-PDW
Improved Economic Decision-Making	5796-ANJ
Departmental Reporting System	5796-PEH
General Cross-Assembler Generator	5796-PKD

Field Developed Programs

The following is a list of IBM Field Developed Programs (FDPs) and their respective program numbers that VM/370 users have found useful. For installation and storage information, consult the publications that support these programs.

VM/370 Performance Monitor Analysis Program	5798-CPX
Financial Planning System	5798-CQT
APL/CMS Interactive Financial Analysis	5798-CFX

Integrated Emulators

Emulator-dependent programs (except for DOS emulation under OS or OS/VS) that execute on a particular System/370 equipped with the appropriate compatibility features can execute on that System/370 in DOS or OS virtual machines under VM/370.

Figure 41 shows, by System/370 model number, which integrated emulators can execute under VM/370 and the compatibility feature numbers (#xxxx) that are required.

No changes are required to the emulators, to DOS or OS, or to VM/370 to allow emulator-dependent programs to execute in virtual machines.

On the System/370 Model 158 only, the virtual machine assist feature cannot operate concurrently with the 7070/7074 compatibility feature (#7117).

In an Attached Processor (AP) system, a virtual machine can use the SET AFFINITY command to make use of an emulator installed on only one of the processors. The Directory option for Affinity may be used instead, with similar results.

System/370 Model	S/360 Model 20	1401 1440 1460	1401 1440 1460 1410 7010	7070 7074	7080	709 7090 7094 7094II
135,135-3,138	#7520	#4457				
145,145-3,148		#4457	#4458			
155 II,158			#3950	#7117		
165 II				#7117	#7118	#7119
168				#7127	#7128	#7129
4331		#3950				

Figure 41. Integrated Emulators that Execute under VM/370

Appendix B. Configuration Aid

Appendix B shows the devices and control units that can be specified in a VM/370 system generation; they are grouped by use. It lists the control units, notes the maximum number that can be specified in the FEATURE= operand of the RCTLUNIT macro, and tells whether or not the control units can operate on a shared subchannel.

It shows the devices that can be attached to each control unit, and lists the operands that can be specified for each device in the RDEVICE macro.

The control units and devices are placed in subgroups according to the ways they can be configured. For example, the chart of tape devices indicates that a 2401, 2402, or 2420 can be attached to a 2803 or 2804 control unit.

Type of Device	RCTLUNIT CUTYPE=	RCTLUNIT Maximum FEATURE=	Shared Sub-channel	RDEVICE DEVTYPE=	RDEVICE Other Operands
System Consoles	1052	--	--	1052	--
	3210	--	--	3210	--
	3215	--	--	3215	--
	2150	--	--	2150	--
	3066	--	--	3066	--
	3138	--	--	3138	--
	3148	--	--	3148	--
	3158	--	--	3158	--
	3036	--	--	3036	--
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

Type of Device	RCTLUNIT CUTYPE=	RCTLUNIT Maximum FEATURE=	Shared Sub-channel	RDEVICE DEVTYPE=	RDEVICE Other Operands
Trans-mission Control Units (cont.)	3704 3705	16-DEVICE 256-DEVICE	--	3704 3705	ADAPTER=BSCA, IBM1, TELE2, TYPE1, TYPE2, TYPE3, or TYPE4 MODEL=A1 through H8 SETADDR=0, 1, 2, or 3 CPTYPE=EP CPNAME=ncpname BASEADD=cuu
	ICA	16-DEVICE	--	ICA	ADAPTER=BSCA, IBM1, TELE2, or SDLC
	2955	--	--	2955	--
Display Devices (Local Attach.)	2848	32-DEVICE	yes	2260 1052	--
	2845	--	yes	2265	--
	2250	--	--	2250	--
	3272[1]	32-DEVICE	yes	3277 3284 3286 3288	FEATURE=OPRDR
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

[1] Specify a 3274 Control Unit Model 1B as a 3272. The following are the valid DEVTYPE operands for a 3274:

Device Type	DEVTYPE
3277	3277
3278	3277
3284	3284
3286	3286
3287	3284 or 3286
3289	3288
3289	3288

If a 3287 is attached to a 3272, the 3287 is specified as a 3288.

Type of Device	RCTLUNIT CUTYPE=	RCTLUNIT Maximum FEATURE=	Shared Sub-channel	RDEVICE DEVTYPE=	RDEVICE Other Operands
Remote 3270 Display Devices (cont)	ICA	--	--	ICA	ADDRESS=cuu (line address) ADAPTER=BSCA CLUSTER=label
	3704 3705	--	--	3704 3705	ADDRESS=cuu (line address) ADAPTER=BSCA CPTYPE=EP BASEADD=cuu CLUSTER=label
Direct Access Storage Devices	2841	--	yes	2311 2321 2303	
	2314 2319 IFA	-- 16-DEVICE	yes	2314 2319	
	3830[2] 3830[2] 3345 ISC	32-DEVICE 64-DEVICE 16-DEVICE 64-DEVICE	--- --- ---	3330 3330 3333	MODEL=1, 2, or 11 FEATURE=SYSVIRT, FEATURE=VIRTUAL MODEL=1 or 11
	3830[2] 3345 ISC IFA[1] 3830[2] ISC IFA[1]	64-DEVICE 16-DEVICE 64-DEVICE 16-DEVICE 64-DEVICE 64-DEVICE 16-DEVICE	 --- ---	3340 3350	
	2820	--	yes	2301	
	2835	16-DEVICE		2305	MODEL=1 or 2
Tape Devices	2803 2804	16-DEVICE 16-DEVICE	yes	2401 2402 2420	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

[1]If using IFA with 3344 or 3350 devices with more than 16 logical units, specify CUTYPE=3830 with either FEATURE=32-DEVICE or 64-DEVICE

[2]Specify a 3880 Model 1 as a 3830. The 3880 Model 1 attaches to the following processors: System 370 Models 145, 145-3, 148, 155-II, 158, 158-3, 165-II, 168, 168-3, 3031, 3032, 3033, 4341.

Type of Device	RCTLUNIT CUTYPE=	RCTLUNIT Maximum FEATURE=	Shared Sub-channel	RDEVICE DEVTYPE=	RDEVICE Other Operands
Tape Devices (cont)	3411	--	yes	3410 3411	MODEL=1, 2, or 3 FEATURE=7-TRACK, DUALDENS
	3803	16-DEVICE[1]	yes	3420	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...])
	1442 1443	--	--	1442P 1443	 CLASS=(class[,class...])
	3811	--	--	3211	CLASS=(class[,class...])
	2826	--	--	1018	
	2520	--	--	2520P	CLASS=(class,[class...])
	3203	--	--	3203	MODEL=4 or 5 FEATURE=UNVCHSET
	3505	--	--	3525	CLASS=(class,[class...])
	3800	--	--	3800	FEATURE=4WCGMS, IMAGE=imagelib, CHARS=ffff,FCB=lpi, DPMSIZE=n
Unit Record Input Devices	2821	--	--	2540R	
	2520	--	--	2520R	
	3505	--	--	3505	
	1442	--	--	1442R	
Special Devices	2495 CTCA	-- --	-- yes	2495 CTCA	
	7443	--	--	7443	

[1] FEATURE=16-DEVICE should be specified for 3808 when the communicator feature is used, allowing access to a second tape control unit and eight more tape drives.

Appendix C. CP/CMS Nucleus/Module/Segment Regeneration Requirements

Whenever a PTF is applied to CMS source code, the CMS nucleus, one of the segments, and/or some CMS modules, must be regenerated. The following table shows which must be regenerated in each case. (If a source name does not appear in the table, either the file is contained within the CMS nucleus, or it is loaded by another file, for example, DMSBTB loads DMSBTP.)

Change in Source	Requires Regeneration of Module/Nucleus/Segment	EXEC Procedure To Use
DMSACC	ACCESS, Nucleus	CMSGEND
DMSACF	ACCESS, Nucleus	CMSGEND
DMSACM	ACCESS, Nucleus	CMSGEND
DMSALU	ACCESS, FORMAT, RELEASE, Nucleus	CMSGEND
DMSAMS	AMSERV	CMSGEND
DMSARE	RELEASE	CMSGEND
DMSASD	ASSEMBLE	CMSGEND
DMSASM	ASSEMBLE	CMSGEND
DMSASN	ASSGN	CMSGEND
DMSBAB	Segment	DOSGEN
DMSBOP	Segment	DOSGEN
DMSBTB	CMSBATCH	CMSGEND
DMSCLS	Segment	DOSGEN
DMSCMP	COMPARE	CMSGEND
DMSCPY	COPYFILE	CMSGEND
DMSDLB	DLBL	CMSGEND
DMSDLK	DOSLKED	CMSGEND
DMSDOS	Segment	DOSGEN
DMSDSK	DISK	CMSGEND
DMSDSL	DOSLIB	CMSGEND
DMSDSV	DSERV	CMSGEND
DMSEDC	EDIT (see Note),Segment	CMSGEND, CMSXGEN
DMSEDF	EDIT (see Note),Segment	CMSGEND, CMSXGEN
DMSEDI	EDIT (see Note),Segment	CMSGEND, CMSXGEN
DMSEDX	EDIT (see Note),Segment	CMSGEND, CMSXGEN
DMSEXT	DMSEXT, Segment	CMSGEND
DMSFCH	Segment	DOSGEN
DMSFLD	FILEDEF	CMSGEND
DMSFOR	FORMAT	CMSGEND
DMSGIO	EDIT (see Note), Segment	CMSGEND, CMSXGEN
DMSGLB	GLOBAL	CMSGEND
DMSGND	GENDIRT	CMSGEND
DMSHDI	HNDINT	CMSGEND
DMSHDS	HNDSVC	CMSGEND
DMSIFC	CPEREP, Nucleus	CMSGEND
DMSLBM	MACLIB	CMSGEND
DMSLBT	TXTLIB	CMSGEND
DMSLDS	LISTDS	CMSGEND
DMSLGT	Segment	CMSXGEN
DMSLIB	Segment	CMSXGEN
DMSLLU	LISTIO	CMSGEND
DMSLSB	Segment	CMSXGEN
DMSLST	LISTFILE	CMSGEND
DMSLSY	Segment	CMSXGEN
DMSM33	Segment	VSAMGEN(AMS)
DMSMDP	MODMAP	CMSGEND

Change in Source	Requires Regeneration of Module/Nucleus/Segment	EXEC Procedure To Use
DMSMVE	MOVEFILE	CMSGEND
DMSOLD	Segment	CMSXGEN
DMSOP1	Segment	DOSGEN
DMSOPT	OPTION	CMSGEND
DMSOR1	Segment	DOSGEN
DMSOR2	Segment	DOSGEN
DMSOR3	Segment	DOSGEN
DMSOVR	SVCTRACE	CMSGEND
DMSOVS	DMSOVS	CMSGEND
DMSPDP	Segment	DOSGEN
DMSPRT	PRINT	CMSGEND
DMSPRV	PSERV	CMSGEND
DMSPUN	PUNCH	CMSGEND
DMSQRY	QUERY	CMSGEND
DMSRDC	READCARD	CMSGEND
DMSRNE	RENUM	CMSGEND
DMSRNM	RENAME	CMSGEND
DMSRRV	RSERV	CMSGEND
DMSS33	Segment	VSAMGEN (AMS)
DMSSAB	Segment	CMSXGEN
DMSSBD	Segment	CMSXGEN
DMSSBS	Segment	CMSXGEN
DMSSCR	EDIT (see Note), Segment	CMSGEND, CMSXGEN
DMSSCT	Segment	CMSXGEN
DMSSEB	Segment	CMSXGEN
DMSSEG	Segment	CMSXGEN
DMSSET	SET	CMSGEND
DMSSLN	Segment	CMSXGEN
DMSSMN	Segment	CMSXGEN
DMSSOP	Segment	CMSXGEN
DMSSQS	Segment	CMSXGEN
DMSSRT	SORT	CMSGEND
DMSSRV	SSERV	CMSGEND
DMSSSK	SETKEY	CMSGEND
DMSSVN	Segment	CMSXGEN
DMSSVT	Segment	CMSXGEN
DMSSYN	SYNONYM	CMSGEND
DMSTMA	TAPEMAC	CMSGEND
DMSTPD	TAPPDS	CMSGEND
DMSTPE	TAPE	CMSGEND
DMSTYP	TYPE	CMSGEND
DMSUPD	UPDATE	CMSGEND
DMSV33	Segment	VSAMGEN (VSAM)
DMSVAN	Segment	VSAMGEN (AMS)
DMSVAS	Segment	VSAMGEN (AMS)
DMSVIP	Segment	VSAMGEN (VSAM)
DMSVPD	DMSVPD	CMSGEND
DMSVVN	Segment	VSAMGEN (VSAM)
DMSVVS	Segment	VSAMGEN (VSAM)
DMSXCP	Segment	DOSGEN
DMSZAP	ZAP	CMSGEND
DMSZIT	EDIT (see Note)	CMSGEND

Note: When the CMSGEND EXEC procedure is invoked for EDIT, it creates the EDIT module, and then automatically reinvokes itself to create the EDMAIN module.

If you must regenerate the CMS nucleus, see "Updating CMS" in Part 5. "Updating VM/370." If you must regenerate the DOS segment, see "Loading and Saving the CMS/DOS Segment called CMS/DOS" in Part 3, "Generating VM/370"; all the other EXEC procedures for generating segments are described in Part 5 "Updating VM/370".

CMS should be resaved whenever the CMS nucleus, CMSSEG, or system S-disk directory is updated. This will insure that the saved CMS system adequately reflects the physical system.

If a CMS module must be regenerated, see "Creating CMS Disk-Resident Modules" in "Part 5. Updating VM/370." The CMSGEND EXEC is described in "EXEC Procedure and Command Format Summaries", also in Part 5.

If you apply a PTF to certain CP source programs, the corresponding CMS modules must be regenerated also to run properly. The source name, module name, and procedures used for regenerating the module are shown in the following table.

Change in Source	Regenerate Module Name Required	EXEC Procedures to Use
DMKDDR	DDR	GENERATE, CMSGEND
DMKDIR	DIR	GENERATE, CMSGEND
DMKFMT	No module name—does not execute under CMS	GENERATE
DMKRND	NCPDUMP	CMSGEND
DMSARD	ASM3705	INSTEP
DMSARX	ASM3705	INSTEP
DMSARN	ASM3705	INSTEP
DMSGRN	GEN3705	CMSGEND
DMSNCP	SAVENCP	CMSGEND

THE DOSGEN

The INSTEP EXEC procedure is described in Part 4. "Generating the 3704/3705 Control Program"; all the other EXEC procedures are described in Part 5. "Updating VM/370."

Appendix D: Compatible Devices

The devices listed below are functionally equivalent to the 2770 Communication System operating on a VM/370 Remote Spooling Communication Subsystem. Details on the feature requirements for VM/370 RSCS use and operational control of such devices are not contained in VM/370 publications but in the programming and operating publications that support these devices.

IBM 6640 Document Printer-Communicating

- Programming Guide for Communicating with the IBM 6640 Document Printer, Form No. G544-1001
- IBM 6640 Document Printer - Communicating User's Guide, Form No. S544-0507
- IBM 6640 Document Printer - Communicating Operating Instructions, Form No. S544-0506

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

- Programming Guide for Communicating with the IBM Office System 6 Information Processors, Form No. G544-1003
- IBM 6/450, 6/440, and 6/430 Information Processors - Communicating User's Guide, Form No. S544-0521
- IBM 6/450, 6/440, and 6/430 Information Processors - Communicating Operating Instructions, Form No. S544-0522

IBM Mag Card II Typewriter - Communicating and IBM 6240 Mag Card Typewriter - Communicating

- Programming Guide for Communicating with the IBM Mag Card II Typewriter and the IBM 6240 Mag Card Typewriter, Form No. G544-1005
- IBM Mag Card II Typewriter - Communicating and IBM 6240 Mag Card Typewriter - Communicating Reference Guide, Form No. S544-0549
- IBM Mag Card II Typewriter - Communicating and IBM 6240 Mag Card Typewriter - Communicating Operating Instructions, Form No. S544-1005

Appendix E: Compatibility of VM/370 with CP-67/CMS

VM/370 and its components, the control program (CP) and the Conversational Monitor System (CMS), are based on the CP-67/CMS system, and are designed especially for the IBM System/370. The Dynamic Translation Facility on the System/370 provides the same facilities as did Dynamic Address Translation (DAT) on the System/360 Model 67, but differs in hardware design details and software implementation. Consequently, the CP-67/CMS system does not run on a System/370, and the VM/370 system does not run on the System/360 Model 67. The internal control block structure differs between the two systems, and user modifications to the CP-67/CMS system may no longer be necessary, desirable, or usable in the new system without some redesign effort.

Certain commands familiar to CP-67 users are not supported in VM/370. The functions available under CP-67 do, however, have equivalents in VM/370 where appropriate. The following is a list of all CP-67 console functions, with operand variations, showing the incompatibilities with VM/370. Functions listed under CP-67 Version 3.1 and not under VM/370 indicate that the syntax and function are identical. There are, of course, additional functions available with the VM/370 commands. The CP and CMS commands are described in the _VM/370 CP Command Reference for General Users_ and the _VM/370 CMS Command and Macro Reference_.

In Figure 42, the letter in the "Status" column has the following meanings:

A indicates CP-67 syntax accepted in VM/370

R indicates CP-67 syntax not accepted but the function is supported by a different VM/370 command

N indicates CP-67 syntax not accepted and the function is not supported in VM/370.

Whenever an R appears in the "Status" column, the VM/370 command that provides the same function as a former CP-67 command is shown in the "VM/370 Equivalent" column.

Status	CP-67 Command	VM/370 Equivalent
R	ACNT	ACNT ALL
A	ATTACH cuu TO {SYSTEM} AS {volid} {userid} {vaddr}	
A	BEGIN	
A	BEGIN hexloc	
A	DCP hexloc1[-hexloc2]	
A	DCP Lhexloc1[-hexloc2]	
A	DCP Thexloc1[-hexloc2]	
A	DETACH vaddr	
R	DETACH raddr	DETACH raddr FROM {SYSTEM} {userid}
A	DIAL system	
N	DIRECT LOCK	
N	DIRECT UNLOCK	
A	DISABLE line	
A	DISABLE ALL	
A	DISCONN	
R	DISCONN xxx	DISCONN HOLD
A	DISPLAY hexloc1 [-hexloc2]	
R	DISPLAY L[-hexloc2]	DISPLAY L0[-hexloc2 -END]
A	DISPLAY Lhexloc1[-hexloc2]	
R	DISPLAY T[-hexloc2]	DISPLAY T0[-hexloc2 -END]
A	DISPLAY Thexloc1[-hexloc2]	
R	DISPLAY K[-hexloc2]	DISPLAY K0[-hexloc2 -END]
A	DISPLAY Khexloc1[-hexloc2]	
A	DISPLAY G[-reg2]	
A	DISPLAY Greg[-reg2]	
A	DISPLAY Y[-reg2]	
A	DISPLAY Yreg[-reg2]	
A	DISPLAY X[-reg2]	
A	DISPLAY Xreg[-reg2]	
A	DISPLAY PSW	
A	DMCP hexloc1[-hexloc2]	
R	DMCP L[-hexloc2]	DMCP L0[-hexloc2 -END]
A	DMCP Lhexloc1[-hexloc2]	

Figure 42. VM/370 Compatibility with CP-67 (Part 1 of 4)

Status	CP-67 Command	VM/370 Equivalent
R	DMCP T[-hexloc2]	DMCP TO{-hexloc2 | -END}
A	DMCP Thexloc1[-hexloc2]	
A	DRAIN	
N	DUMP	
A	ENABLE line	
A	ENABLE ALL	
A	EXTERNAL	
A	IPL CMS	
A	IPL devadd	
R	IPLSAVE CCW	IPL vaddr NOCLEAR
R	KILL userid	FORCE userid
N	KILL	
A	LINK userid xxx yyy {W | R}[PASS=pwd]	
R	LINK userid xxx yyy {W | R} (NOPASS)	LINK userid xxx yyy {W | R}
A	LOCK userid fpage lpage	
A	LOGIN userid	
R	LOGIN userid xxx	LOGON userid MASK
A	LOGOUT	
R	LOGOUT xxx	LOGOFF HOLD
A	MSG userid line	
R	MSG CP line	MSG OPERATOR line
A	MSG ALL line	
R	PSWRESTART	SYSTEM RESTART
A	PURGE READER	
A	PURGE PRINTER	
A	PURGE PUNCH	
R	QUERY DEVICE ALL	QUERY ALL
R	QUERY DEVICE xxx	QUERY xxx
A	QUERY DUMP	
A	QUERY FILES	

Figure 42. VM/370 Compatibility with CP-67 (Part 2 of 4)

Status	CP-67 Command	VM/370 Equivalent
A	QUERY LOGMSG	
N	QUERY MAX	
A	QUERY NAMES	
n	QUERY PORTS	
R	QUERY PORTS ALL	QUERY LINES
N	QUERY PORTS FREE	
R	QUERY PORTS xxx	QUERY xxx
A	QUERY PRIORITY userid	
R	QUERY Q2	QUERY PAGING
A	QUERY TIME	
A	QUERY userid	
A	QUERY USERS	
A	QUERY VIRTUAL xxx	
A	QUERY VIRTUAL CORE	QUERY VIRTUAL STORAGE
A	QUERY VIRTUAL ALL	
A	READY xxx	
A	REPEAT xxx y	
R	RESET	SYSTEM RESET
R	SET ADSTOP xxxxxx	ADSTOP xxxxxx
R	SET ADSTOP OFF	ADSTOP OFF
R	SET APLBALL {ON, OFF}	TERMINAL APL {ON, OFF}
R	SET ATTN {ON, OFF}	TERMINAL ATTN {ON, OFF}
R	SET CARDSAVE {ON, OFF}	SPOOL READER {HOLD, NOHOLD}
A	SET DUMP xxx	
A	SET {LINEDIT, RUN} {ON, OFF}	
A	SET LOGMSG	
A	SET LOGMSG NULL	
A	SET LOGMSG n	
N	SET MAX	
A	SET MSG {ON, OFF}	
R	SET Q2 nn	SET PAGING nn
R	SET TRACE devtype	TRACE type dev
R	SET TRACE OFF	TRACE type OFF
R	SET TRACE END	TRACE END
A	SET WNG {ON, OFF}	
A	SHUTDOWN	
A	SLEEP	

Figure 42. VM/370 Compatibility with CP-67 (Part 3 of 4)

Status	CP-67 Command	VM/370 Equivalent
A	SPACE xxx	
R	SPOOL xxx ON yyy	SPOOL yyy CLASS x
A	SPOOL xxx CONT	
R	SPOOL xxx OFF	SPOOL xxx NOCONT
A	START xxx	
A	STCP hexloc	
A	STCP Shexloc	
A	STORE Lhexloc hexinfo...	
A	STORE Shexloc hexstring	
A	STORE Greg hexinfo...	
A	STORE Yreg hexinfo...	
A	STORE Xreg hexinfo...	
A	STORE PSW [hexinfo1] hexinfo2	
R	TERM xxx	FLUSH xxx
A	UNLOCK userid fpage lpage	
A	WNG userid text	
A	WNG ALL text	
R	XFER xxx {TO userid / OFF}	SPOOL xxx {TO userid / OFF}

Figure 42. VM/370 Compatibility with CP-67 (Part 4 of 4)

Incompatibility Statement to CP-67/CMS Users

Although the CMS in VM/370 is built upon CMS Version 3.1 in CP-67/CMS, there are five types of modifications that were made to 3.1 that affect the relationship between versions:

1. Unchanged: Some commands and system functions remain unchanged; therefore, complete compatibility exists.

2. Additional Functional Capability: Functional and syntactical enhancements are effected; but, in some cases, old keywords and functions are supported.

3. Command Name Alterations: Commands have name changes, but a SYNONYM file may be included during nucleus system generation.

4. Keyword Changes: Some keywords within a command are modified, deleted or added.

5. Major Modifications: Improvements to commands and system functions caused complete incompatibilities in the following areas:

   - Because the CMS nucleus is significantly larger, all MODULES must be recreated from their object (TEXT) files using the GENMOD command.

   - Because you may now have up to 10 disks, the logical directory identifications (filemode letters) are changed to reflect a more natural, easy-to-remember, search order: P, T, A, B, S, C becomes A, B, C, D, E, F, G, S, Y, Z -- with the system disk being the S-disk, and the primary disk becoming the A-disk.

   - The following global changes of filetypes must be made:

     SYSIN to ASSEMBLE
     ASP360 to MACRO

   - For language processors, the DECK and NODECK options have a new meaning, they route the object (TEXT) file to the spooled card punch; the LOAD and NOLOAD options now invoke the function formerly performed by DECK and NODECK, and the writing of the TEXT file onto a CMS disk.

   - No 2311 disk support is provided for CP or CMS files.

   - In Version 1.0 the tape designations are as follows:

     TAP1 is for 181
     TAP2 is for 182

     The default for tape commands is TAP1.

- CMS does not function on a real CPU without the control program.
- TXTLIB files must be recreated.
- Because many fields are changed in the CMS nucleus or rearranged, many of your programs that refer to these fields have to be reassembled with the new CMS macro libraries.
- Modules that refer to fields containing sizes, limits, and quantities within the CMS nucleus may have to be reassembled and then regenerated.
- SYSLIB MACLIB is renamed to CMSLIB MACLIB.
- All EXEC files should be checked for command name and operand changes, filemode usage, and so on, and changed to conform to VM/370 CMS. Major changes are:

```
&TYPEOUT to &CONTROL
&PRINT   to &TYPE
&INDEX0  to &RETCODE
```

- The options specified for a LOAD command do not remain in effect for subsequent INCLUDE commands; options are reset to default settings unless the SAME option is specified.
- Filenames and filetypes must be composed entirely of alphameric characters.
- If the CMS Version 1.0 system does not recognize a command name, the command line is automatically passed to CP. If the CMS SET and QUERY commands do not recognize an operand or option, the command line is passed to CP. This is not true for EXEC files; the CP command must be explicitly stated. The feature may be negated by entering the command:

```
SET IMPCP OFF
```

VM/370 CMS Support of CP-67/CMS Commands

<u>ALTER</u>

Command name changed to RENAME.
Components of the new file identifier may not be specified as asterisk (*). An equal sign (=) performs the same function.
NOUP option keyword changed to NOUPDIRT; NOUP is the abbreviation.
Default options added: NOTYPE, UPDIRT.

<u>ASSEMBLE</u>

Only one file may be assembled per ASSEMBLE command.

<u>Options Changed</u>:
NODECK is the default
<u>DIAG</u>|NODIAG changed to <u>TERM</u>|NOTERM
LTAPn not supported
LDISK option name changed to DISK

<u>Options Added</u>:
<u>LOAD</u>|NOLOAD, <u>ALGN</u>|NOALGN
<u>OS</u>|DOS, TEST|<u>NOTEST</u>, LINECT nn|<u>55</u>,
<u>NUM</u>|NONUM, STMT|<u>NOSTMT</u>

BLIP — Functionally supported by SET BLIP.

BRUIN — Not implemented.

CEDIT — Functionally supported by EDIT.

CHARDEF — Functionally supported by CP TERMINAL CHARDEF command.

CLOSIO — Functionally supported by the CP commands, SPOOL and CLOSE.

CLROVER — Functionally supported by SVCTRACE OFF command.

CNVT26 — Functionally supported by COPYFILE command with EBCDIC option.

COMBINE — Functionally supported by COPYFILE.

COMPARE — Filemode required.
NOSEQ option functionally supported by COL option.

CPFUNCTN — Command name changed to CP.
NOMSG option not supported.

CVTFV — Functionally supported by COPYFILE.

DEBUG — No change.

DEBUG Subcommands

BREAK	No change.
CAW	No change.
CSW	No change.
DEF	Subcommand name changed to DEFINE; DEF minimum truncation; no change in format.
DUMP	No change.
GO	No change.
GPR	No change.
IPL	Not supported from DEBUG.
KX	Supported by CMS command HX.
ORIGIN	No change.
PSW	No change.
RESTART	Not supported.
RETURN	No change.
SET	No change.
STORE	No change.
TIN	Not supported.
X	If symbol specified, length of field defaults to 4.

DISK — No change.

DUMPD — Functionally supported by DDR command.

DUMPF — Functionally supported by TYPE command with HEX option.

DUMPREST — Functionally supported by DDR command.

ECHO — Functionally supported by CP command ECHO.

EDIT — Filename must be specified.
Filemode may be specified.
LRECL may be specified for a new file.

EDIT Subcommands

BACKSPACE	Functionally supported by CMS command, SET INPUT.
BLANK	Functionally supported by OVERLAY.
BOTTOM	No change.
BRIEF	Function accomplished by VERIFY OFF request.
CHANGE	No change.
DELETE	/string/ is not valid as an operand. (DELETE * may be used to delete to end of file.)
FILE	Filetype and filemode may be specified.
FIND	No change.
INPUT	No change.
INSERT	Functionally supported by INPUT request.
LOCATE	No change.
NEXT	No change.
OVERLAY	No change.
PRINT	Request name changed to TYPE. (TYPE * may be specified to indicate that typing is to continue until EOF.) Instead of L in second field, * is specified.
QUIT	No change.
REPEAT	Valid only for subsequent OVERLAY subcommand. (REPEAT * may be specified to indicate that the OVERLAY subcommand is to be repeated for the remainder of the file.)

Edit Subcommand (cont.)

RETYPE	Request name changed to REPLACE.
SAVE	Filetype and filemode may be specified.
SERIAL	First operand changed to {OFF|'seq'(same as ID)|ON|ALL}.
TABDEF	Functionally supported by CMS command SET INPUT.
TABSET	No change.
TOP	No change.
UP	No change.
VERIFY	Operand format changed; function added.
X	No change.
Y	No change.
ZONE	No change.

ERASE Default option added; old format accepted.
* * * Not supported.

EXEC No change.

EXEC Control Words

&ERROR	Action does not default to &CONTINUE.
&IF	No change.
&EXIT	No change.
&QUIT	Functionally supported by &EXIT 0.
&SKIP	No change.
&GOTO	EXIT not supported as operand. Line-number now a valid operand.
&LOOP	No change.
&CONTINUE	No change.
&TYPEOUT	Functionally supported by &CONTROL control word; ON, NOEXEC, RESUME, KILL not valid. CMS operand added.
&TIME	Operands changed to [ON | OFF] [RESET | TYPE].
&SPACE	No change.
&PRINT	Functionally supported by &TYPE.

&UPRINT	Functionally supported by &BEGTYPE.
&PUNCH	No change.
&UPUNCH	Functionally supported by &BEGPUNCH.
&COMMENT	No change. Function duplicated by * card.
&ARGS	No change.
&READ	VARS operand added.
&STACK	No change.
&BEGSTACK	ALL operand added.
&ENDSTACK	Functionally supported by &END.
&SET	Not implemented.

<u>FILEDEF</u> Device names changed:

CON	– TERMINAL
DSK	– DISK
DSK–nn	– not supported
DUMMY	– no change
PRT	– PRINTER
PUN	– PUNCH
RDR	– READER
TAPEn	– unchanged
BAT	– not supported

<u>FINIS</u> Not supported from terminal.

<u>FORMAT</u> All functions supported; all operands changed.

<u>FORTRAN</u> This command is now supported by the commands FORTG1, FORTHX, GOFORT, and CONVERT which invoke IBM Program Products.

<u>GENDIRT</u> Target mode parameter is added.

<u>GENMOD</u> Incompatible; positional parameters and options changed; equivalent function performed.

<u>GLOBAL</u> PRINT function removed; others compatible.

<u>IPL</u> Not explicitly supported in CMS; however, the command may be issued and is passed to the control program for processing. Either a device address or system name must be specified. Cyl–no may be specified.

<u>Options Added</u>: CLEAR|<u>NOCLEAR</u>.

| <u>KE</u> No functional equivalent.

<u>KO</u> Command name changed to HO.

<u>KT</u> Command name changed to HT.

<u>KX</u> Command name changed to HX.

<u>LINEND</u> Functionally supported by the CP command TERMINAL LINEND.

LISTF — Command name changed to LISTFILE; LISTF accepted.

Option Changes:

- SORT— Option not implemented.
- ITEM— Option not implemented; ALLOC option produces both logical records and blocks.
- NAME— Option name changed to FNAME.
- TYPE— Option name changed to FTYPE.
- MODE— Option name changed to FMODE.
- REC— Supported by ALLOC option.
- DATE— Produces mm/dd/yy hh:mm.
- YEAR— Not implemented; functionally supported by DATE option.
- TIME— Not implemented; functionally supported by DATE option.
- LABEL and FORMAT options added.
- APPEND option added.
- HEADER|NOHEADER option added.

LOAD — Option Changes:

- SLCxxxxxx— option changed to ORIGIN xxxxxx.
- SLC12000— default changed to first available location.
- SINV— option name changed to NOINV.
- PINV— option name changed to INV.
- SREP— option name changed to NOREP.
- PREP— option name changed to REP.
- SLIBE— option name changed to NOLIBE.
- SAUTO— option name changed to NOAUTO.
- XEQ— option name changed to START.
- NOXEQ— option not supported.

Options Added: RESET.

TXTLIB files may no longer be specified in a LOAD command, but must have been previously specified by a GLOBAL command.

LOADMOD — Filetype must be specified if filemode is given.

LOGIN — Functionally supported by LOGON and ACCESS commands. Comma between mode and extdisk replaced with slash (/); extdisk optional.

Options:

- NOPROF—option supported.
- NOTYPE—option not supported.
- NO-UFD—option name changed to ERASE.

LOGOUT — Functionally supported by CP command LOGOFF.

MAPPRT — Functionally supported by CMS commands TYPE or PRINT.

MODMAP — No change.

OFFLINE READ — Functionally supported by READCARD command.

OFFLINE PRINT — Functionally supported by PRINT command.

OFFLINE PRINTCC Functionally supported by PRINT command.

OFFLINE PRINTUPC Functionally supported by PRINT command.

OFFLINE PUNCH Functionally supported by PUNCH command.

OFFLINE PUNCHCC Functionally supported by PUNCH command.

OFFLINE PUNCHDT Functionally supported by PUNCH command.

OSTAPE Command name Changed to TAPPDS.

PLI This command is now supported by the command PLIOPT which invokes an IBM Program Product.

PRINTF Command name changed to TYPE.
Filemode may be specified.
n3 functionally supported by COL option.

Options added: HEX, MEMBER.

REUSE Command name changed to INCLUDE.
Option differences are the same as for LOAD.
TXTLIB files may no longer be specified in the command; but must have been previously specified by a GLOBAL command.

RT No change.

SCRIPT Files may be processed by SCRIPT/370, an IBM user-installed program.

SETERR Functionally supported by SVCTRACE ON.

SETOVER Functionally supported by SVCTRACE ON.

SNOBOL Not implemented.

SORT Filemode must be specified for both input and output files.

SPLIT Functionally supported by COPYFILE command.

START (NO) operand not valid.

STAT Functionally supported by QUERY command.

STATE No change.

SYN — Command name changed to SYNONYM; SYN minimum truncation. Filetype must be SYNONYM.

Options:
SYNONYM command with P and PUSER options is functionally supported by QUERY SYNONYM.

TAPE — TAPn may also be specified by a virtual address. "n" options after SCAN, SKIP, SLOAD, replaced by 'EOF n'.

Options Added: 7TRACK|9TRACK, DEN, TRTCH.

Functions Added: MODESET, BSF, BSR, ERG, FSF, FSR, RUN, REW.

TAPE DUMP — Options added: WTM|NOWTM, NOPRINT|PRINT|TERM|DISK.

TAPE LOAD — Functionally supported by TAPE LOAD EOFn. File identifiers may be specified.

Options Added: NOPRINT|PRINT|TERM|DISK, EOFn|EOT|EOF1.

TAPE SCAN — Filename and filetype may be specified. Functionally supported by TAPE SCAN (EOF n).

Options added: NOPRINT|PRINT|TERM|DISK, EOFn|EOT|EOF1.

TAPE SKIP — Comments same as for TAPE SCAN.

TAPE SLOAD — Functionally supported by TAPE LOAD fn ft. Filemode may be specified.

TAPEIO — Functionally supported by TAPE.

TAPPDS — Default filename is TAPPDS for NOPDS option. Default filetype is CMSUT1.

Option Changes:
NPDS - option name changed to NOPDS.
NCOLI - option name changed to NOCOLI.
TAPx - default is TAP1.
NEND - option name changed to NOEND.
NMAXTEN - option name changed to NOMAXTEN.

TAPRINT — Functionally supported by MOVEFILE command.

TPCOPY — Functionally supported by MOVEFILE command.

TXTLIB — PRINT and LIST functions supported by MAP function.

UPDATE — Option Changes:
P - option name changed to REP.
Default options added NOREP, NOSEQ8, NOINC.

USE — Functionally supported by INCLUDE command with the SAME option. See discussion of REUSE compatibility.

VSET — Command name changed to SET.

Functions:

BLIP	ON may be specified to return to default
CHARDEF	Functionally supported by CP command TERMINAL|{CHARDEL|LINEDEL|ESCAPE}
IMPEX	No change.
LDRTBLS	No change.
LINEND	Functionally supported by CP command TERMINAL LINEND.
RDYMSG	ON|OFF changed to LMSG|SMSG.
REDTYPE	No change.
RELPAGE	No change.

Functions added: INPUT, OUTPUT, ABBREV, IMPCP.

WRTAPE — Functionally supported by TAPE command or MOVEFILE command.

$ — Command name changed to RUN.
Filetype and filemode may be specified.

Files may also have filetypes in addition to EXEC, MODULE, and TEXT of those used by the language processors for input.

CP-67/CMS Macros and Functions and Corresponding CMS Macros

The list below shows VM/370 CMS macros that correspond to CP-67/CMS macros and functions. The CP-67/CMS functions have no structural equivalent in VM/370 CMS, but in most cases the function is available via a VM/370 CMS macro. If you need information on how to build a parameter list that can be scanned properly by VM/370 CMS, refer to the VM/370 System Programmer's Guide or VM/370 CMS User's Guide.

CP-67/CMS Macros	VM/370 Equivalent	CP-67 Functions	VM/370 Macros Equivalent
CKEOF	Not Available	DEBDUMP	Not Available
CMSREG	REGEQU	ERASE	FSERASE[1]
CMSYSREF	Not Available	FILEDEF	[1]
ERASE	FSERASE	FINIS	FSCLOSE
FCB	No Change	HNDINT	HNDINT[1]
FINIS	FSCLOSE	HNDSVC	HNDSVC[1]
FSTB	No Change	POINT	FSCB, FSREAD, FSWRITE, FSOPEN
RDBUF	FSREAD		
SETUP	FSOPEN	PRINTR	PRINTL
STATE	FSSTATE	RDBUF	FSREAD
TYPE	WRTERM	STATE	FSSTATE
TYPIN	RDTERM	STRINIT	No Change
WRBUF	FSWRITE	TAPEIO	RDTAPE, WRTAPE, TAPECTL[1]
ATTN	No Change	TRAP	HNDEXT
CARDIO	PUNCH,RDCARD	TYPE	WRTERM[1]
CLOSIO	[1]	TYPLIN	WRTERM
CONWAIT	WAITT	WAIT	WAITD
CPFUNCTN	[1]	WAITRD	RDTERM
		WRBUF	FSWRITE

[1] See the VM/370 System Programmer's Guide for information on how to call a command from a program.

Appendix F. VM/370 Restrictions

A virtual machine created by VM/370 is capable of running an IBM System/360 or System/370 operating system as long as certain VM/370 restrictions are not violated. Virtual machine restrictions and certain execution characteristics are stated in this appendix.

Dynamically Modified Channel Programs

In general, virtual machines may not execute channel programs that are dynamically modified (that is, channel programs that are changed between the time the START I/O (SIO) is issued and the time the input/output ends, either by the channel program itself or by the processor).

Exceptions (that is, dynamically modified channel programs given special consideration by CP) 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/370 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/370 directory entry. There is no such restriction for DOS ISAM, or for ISAM if it is running in an OS/VS virtual=virtual partition. If ISAM is to run in an OS/VS virtual=real partition, you must specify the ISAM option in the VM/370 directory entry for the OS/VS virtual machine.

Virtual machines using OS/VS TCAM (Level 5, generated or invoked with the VM/370 option) issue a DIAGNOSE instruction when the channel program is modified. This instruction causes CP to reflect the change in the virtual CCW string to the real CCW string being executed by the channel. CP is then able to execute the dynamically modified channel program properly.

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/370 modifies the home address data in user storage at the completion of the channel program because the addresses must be converted for minidisks; therefore, the data buffer area may not be dynamically modified during the input/output operation.

2. On a minidisk, if a CCW string uses multitrack search on input/output operations, subsequent operations to that disk must have preceding seeks or continue to use multitrack operations. There is no restriction for dedicated disks.

3. OS/PCP, MFT, and MVT ISAM or OS/VS ISAM running virtual=real may be used with a minidisk only if the minidisk is located at the beginning of the physical disk (that is, at cylinder 0). There is no such restriction for DOS ISAM or OS/VS ISAM running virtual=virtual.

   Note: Because the VS1 system does no paging, any ISAM programs run under VS1 are treated by VM/370 as though they are running in an ADDRSPC=REAL partition.

4. VM/370 does not return an end-of-cylinder condition to a virtual machine that has a virtual 2311 mapped to the top half (that is, tracks 0 through 9) of 2314 or 2319 cylinders.

5. If the user's channel program for a minidisk does not perform a seek operation, then to prevent accidental accessing, VM/370 inserts a positioning seek operation into the user's channel program. Thus, certain channel programs may generate a condition code (CC) of 0 on a 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, or 3350 device may give results on dedicated drives which differ from results on minidisks having non-zero relocation factors if the channel program includes multiple-track operations and depends on a search ID high or a search ID equal or high to terminate the program. This is because the record 0 count fields on the 3330, 3340, and 3350 must contain the real cylinder number of the track on which they reside. Therefore, a search ID high, for example, based on a low virtual cylinder number may terminate prematurely if a real record 0 is encountered.

   Notes:

   1. Minidisks with non-zero relocation factors on 3330, 3340, and 3350 devices are not usable under OS and OS/VS systems when the minidisk contains a VTOC of more than one track. The locate catalog management function employs a search ID equal or high CCW to find the end of the VTOC. Since VM/370 does not permit the guest to write R0, the VTOC search ends prematurely.

7. On a 3330, 3340, or 3350, an OS/VS, or OS minidisk must start at real cylinder 0 unless the VTOC is limited to one track.

8. The IBCDASDI program cannot assign alternate tracks for a 3330, 3340, or 3350 minidisk.

9. If the DASD channel programs directed to 3330/3340/3350 devices include a write record R(0), results differ depending on whether the 3330/3340/3350 is dedicated (this includes a minidisk defined

as the entire device) or nondedicated. For a dedicated 3330/3340/3350, a write R(0) is allowed, but the user must be aware that the track descriptor record may not be valid from one 3330/3340/3350 to another. For a nondedicated 3330/3340/3350, a write record R(0) is replaced by a read record R(0) and the skip flag is set on. This could result in a command reject condition due to an invalid command sequence.

10. When performing DASD I/O, if the record field of a search ID argument is zero when a virtual Start I/O is issued, but the search ID argument is dynamically read by the channel program before the search ID CCW is executed, then the real search ID uses the relocated search argument instead of the argument that was read dynamically. To avoid this problem, the record field of a search ID argument should not be set to binary zero if the search argument is to be dynamically read or if a search ID on record 0 is not intended.

Timing Dependencies

Timing dependencies in input/output devices or programming do not function consistently under VM/370:

1. The following telecommunication access methods (or the designated option) violate the restriction on timing dependency by using program-controlled interrupt techniques and/or the restriction on dynamically modified channel programs:

   - OS Basic Telecommunications Access Method (BTAM) with the dynamic buffering option.
   - OS Queued Telecommunications Access Method (QTAM).
   - DOS Queued Telecommunications Access Method (QTAM).
   - OS Telecommunications Access Method (TCAM).
   - OS/VS Telecommunications Access Method (TCAM) Level 4 or earlier, and Level 5 if TCAM is not generated or invoked with the VM/370 option.

   These access methods may run in a virtual=real machine with CCW translation suppressed by the SET NOTRANS ON command. Even if SET NOTRANS ON is issued, CCW translation will take place if one of the following conditions is in effect:

   - The channel program is directed at 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 zero or beyond the end of the virtual=real area.

(OS BTAM can be generated without dynamic buffering, in which case no virtual machine execution violations occur. However, the BTAM reset poll macro will not execute under VM/370 if issued from third level storage. For example, a reset poll macro has a NOP effect if executed from a virtual=virtual storage under VS1 which is running under VM/370.)

2. Programming that makes use of the PCI channel interrupt for channel program modification or processor signalling must be written so that processing can continue normally if the PCI is not recognized until I/O completion or if the modifications performed are not executed by the channel.

3. Devices that expect a response to an interrupt within a fixed period of time may not function correctly because of execution delays caused by normal VM/370 system processing. An example of such a device is the IBM 1419 Magnetic Character Reader.

4. The operation of a virtual block multiplexer channel is timing dependent. For this reason, the channel appears available to the virtual machine operating system, and channel available interrupts are not observed. However, operations on virtual block-multiplexing devices should use the available features like Rotational Position Sensing to enhance utilization of the real channels.

Processor Model-Dependent Functions

On the System/370 Model 158 only, the virtual machine assist feature cannot operate concurrently with the 7070/7074 compatibility feature (#7117).

Programs written for processor model-dependent functions may not execute properly in the virtual machine under VM/370. The following points should be noted:

1. Programs written to examine the machine logout area do not have meaningful data since VM/370 does not reflect the machine logout data to a virtual machine.

2. Programs written to obtain processor identification (via the Store CPUID instruction, STIDP) receive the real machine value. When the STIDP instruction is issued by a virtual machine, the version code contains the value 255 in hexadecimal ("FF") to represent a virtual machine.

3. No simulation of other processor models is attempted by VM/370.

4. Since 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/370 will be running on.

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. 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 comply with 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.

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 comply with 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 an arbitrary channel, one of several 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.

Since 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 consistent with the data returned by STIDC (provided that the virtual-to-real channel mapping complies with the requirement stated previously).

A deviation 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/370 are as follows:

1. If the virtual=real option is selected for a virtual machine, input/output operations specifying data transfer into or out of the virtual machine's page zero, or into or out of storage locations whose addresses are greater than the storage allocated by the virtual=real option, must not occur. The storage-protect-key mechanism of the IBM System/370 processor and channels operates in

these situations but is unable to provide predictable protection to other virtual machines. In addition, violation of this restriction may compromise the integrity of the system. The results are unpredictable.

2. A two-channel switch can be used between the IBM System/370 running a virtual machine under VM/370 and another processor.

3. The DIAGNOSE instruction cannot be issued by the virtual machine for its normal function. VM/370 uses this instruction to allow the virtual machine to communicate system services requests. The Diagnose interface requires the operand storage addresses passed to it to be real to the virtual machine issuing the DIAGNOSE instruction. For more information about the DIAGNOSE instruction in a virtual machine, see the VM/370 System Programmer's Guide.

4. A control unit normally never appears busy to a virtual machine. An exception exists when a forward space file or backward space file command is executed for a tape drive. Subsequent I/O operations to the same virtual control unit result in a control unit busy condition until the forward space file or backward space file command completes. If the real tape control unit is shared by more than one virtual machine, a control unit busy condition is reflected only to the virtual machine executing the forward space file or backward space file command. When a virtual machine attempts an I/O operation to a device for which its real control unit is busy, the virtual machine is placed in I/O wait (nondispatchable) until the real control unit is available. If the virtual machine executed a SIOF instruction (rather than SIO) and was enabled for block-multiplexing, it is not placed in I/O wait for the above condition.

5. The CP IPL command cannot simulate self-modifying IPL sequences off dedicated unit record devices or certain self-modifying IPL sequences off tape devices.

6. The VM/370 spooling facilities do not support punch-feed-read, stacker selection, or column binary operations. Detection of carriage control channels is supported for a virtual 3211 only.

7. VM/370 does not support count control on the virtual 1052 operator's console.

8. Programs that use the integrated emulators function only if the real computing system has the appropriate compatibility feature. VM/370 does not attempt simulation. The DOS emulator running under OS or OS/VS is not supported under VM/370.

9. The READ DIRECT and WRITE DIRECT instructions are not supported for a virtual machine.

10. The System/370 SET CLOCK instruction cannot be simulated and, hence, is ignored if issued by a virtual machine. The System/370 STORE CLOCK instruction is a nonprivileged instruction and cannot be trapped by VM/370; it provides the true TOD clock value from the real 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 via a virtual 2701, 2702, or 2703 line.

12. The pseudo-timer (usually device address OFF, device type TIMER) does not return an interrupt from a Start I/O; therefore, do not use EXCP to read this device.

13. A virtual machine device IPL with the NOCLEAR option overlays one page of virtual machine storage. The IPL simulator uses one page of the virtual machine to initiate the IPL function. The starting address of the overlaid page is either the result of the following formula:

$$\frac{\text{virtual machine size}}{2} = \text{starting address of the overlayed page}$$

or the hexadecimal value 20000, whichever is smaller.

14. To maintain system integrity, data transfer sequences to and from a virtual system console are limited to a maximum of 2032 bytes. Channel programs containing data transfer sequences that violate this restriction are terminated with an interrupt whose CSW status indicates incorrect length and a channel program check.

<u>Notes:</u>

1. 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. Data chain seek CCWs with counts of less than four are inconsistent with data security of VM/370 and therefore will give an inconsitent error.

2. Data chained seek CCWs with counts of less than four are inconsistent with the data security of VM/370 and therefore will give an inconsistent error when attempting to use.

15. When an I/O error occurs on a device, the System/370 hardware maintains a contingent connection for that device until a SENSE channel command is executed and sense data is recorded. That is, no other I/O activity can occur on the device during this time. Under VM/370, the contingent connection is maintained until the SENSE command is executed, but I/O activity from other virtual machines can begin on the device while the sense data is being reflected to the virtual machine. Therefore, the user should be aware that on a shared disk, the access mechanism may have moved during this time.

16. The mode setting for 7-track tape devices is maintained by the control unit. Therefore, when a virtual machine issues the SET MODE channel command to a 7-track tape device, it changes the mode setting of all 7-track tape devices attached to that control unit.

    This has no effect on virtual machines (such as OS or DOS) that issue SET MODE each time a CCW string is to be executed. However, it can cause a problem if a virtual machine fails to issue a SET MODE with each CCW string executed. Another virtual machine may change the mode setting for another device on the same control unit, thereby changing the mode setting of all 7-track tape devices attached to that control unit.

17. OS/VS2 is supported in uniprocessor mode only.

18. A shared system or one that uses discontiquous saved segments cannot be loaded (via IPL) into a virtual machine running in the virtual=real area.

19. The DUMMY feature for VSAM data sets is not supported and should not be used at program execution time. Specifying this option on the DLBL command will cause an execution-time OPEN error

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 retrieval is not supported for OS/MFT or VS1.

MSS Restrictions

1. There are two OS/VS system data sets associated with 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 on 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 compatible; the definition of 3330V addresses in the MCS 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/370 System Programmer's Guide.

4. Each active volume in the MSS must have a unique 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/370 residence, paging, spooling, or temporary disk space.

6. You must not change the volume 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.

CMS Restrictions

The following restrictions apply to CMS, the conversational subsystem of VM/370:

1. CMS executes only on a virtual IBM System/370 provided by VM/370.

2. The maximum sizes (in cylinders) of CMS minidisks are as follows:

Disk	Maximum Cylinders	CMS/VSAM
2314/2319	203	200
3330 Model 1	246	404
3330 Model 11	492	808
3340 Model 35	349	349
3340 Model 70/3344	682	698
3350 Series	115	555

3. CMS employs the spooling facilities of VM/370 to perform unit record I/O. However, a program running under CMS can issue its own SIOs to attached dedicated unit record devices.

4. Only those OS and DOS facilities that are simulated by CMS can be used to execute OS and DOS programs produced by language processors under CMS.

5. Many types of object programs produced by CMS (and OS) languages can be executed under CMS using CMS's simulation of OS supervisory functions. Although supported in OS and DOS virtual machines under VM/370, the writing and updating of non-VSAM OS data sets and DOS files are not supported under CMS.

6. CMS can read sequential and partitioned OS data sets and sequential DOS files, by simulating certain OS macros.

   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.
   - BDAM and ISAM data sets are not read.
   - 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.
   - 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 bypassed.

   The following restrictions apply when CMS reads DOS files that reside on DOS disks:

   - Only DOS 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 DOS sequential files.
   - The following types of DOS files cannot be read:

     --DOS DAM and ISAM files.

     --Files with the input security indicator on.

     --DOS 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. There is no end-of-volume switching.

- User labels in user-labeled files are bypassed.

- Since DOS files do not contain BLKSIZE, RECFM, or LRECL parameters, these parameters must be specified via FILEDEF or DCB parameters; otherwise, defaults of BLOCKSIZE=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, since the CMS/DOS implementation of the DUMMY statement corresponds to the DOS/VS implementation. Specifying the DUMMY option with the DLBL command will cause an execution-time error.

7. Assembler program usage of VSAM and the ISAM Interface Program (IIP) is not supported.

Miscellaneous Restrictions

1. If you intend to run VM/370 Release 1 and pre-PLC 9 Release 2 systems alternately, apply Release 1 PLC 14 or higher (APAR V1179) to your Release 1 system, to provide compatibility and to prevent loss of spool files in case of a warm start. Changes to the spool file format in PLC 9 of Release 2 require a cold start when switching between pre-Release 2 PLC 9 and post-Release 2 PLC 9 systems.

2. The number of pages used for input/output must not exceed the total number of user pages available in real storage. Violation of this restriction causes the real computing system to be put into an enabled wait state.

3. If you intend to define more than 64 virtual devices for a single virtual machine, be aware that any single request for free storage in excess of 512 doublewords (a full page) can cause an error message to be issued if storage cannot be obtained. Tables for virtual devices for a virtual machine must reside in contiguous storage. Therefore, two contiguous pages of free storage must be available in order to log on a virtual machine with more than 64 virtual devices, (three contiguous pages for a virtual machine with more than 128 virtual devices, etc.). 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 64 devices should be the first to log on after IPL. The larger the real machine size, the lesser the possibility of this occurring.

4. For remote 3270s, VM/370 supports a maximum of 16 binary synchronous lines, minus the number of 3704/3705 Communications Controllers in NCP mode minus one (if there are any 3704/3705 Communications Controllers in emulation mode).

5. If an I/O device (such as a disk or tape drive) drops ready status 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 outstanding I/O is finished. The system operator should determine which I/O device is involved and make that device ready once more.

6. Any modifications to local OPTIONS COPYFILE, unless otherwise specified in existing documentation, is not supported.

7. If an installation is using an IBM 3031, 3032, or 3033 processor, it must dedicate the service record file (SRF) device to VM/370. Thus, the channel on which the SRF is located cannot be dedicated to any virtual machine.

8. When using the SPOOL, DEDICATE, and SPECIAL directory control statements to define virtual devices, specify virtual addresses that do not conflict or content with the virtual control unit interface. This conflict or contention occurs because devices can require special I/O interface protocol from control units such as shared and nonshared subchannel operations. Putting devices that require different real control units on the same virtual control unit can result in a hung or busy condition. To avoid this problem, users must define (and separate) devices within their own control unit range. For example, if the directory entry specifies:

   ```
   SPOOL 102 3211
   SPECIAL 103 3270
   ```

   The control unit 0 on channel 1 controls both a nonshared device (the 3211 printer) and a shared device (the 3270 display unit). Processing of channel programs involving these two devices can result in a hung or busy condition.

9. The number of virtual devices for a virtual machine cannot exceed the value determined by (7FFF/VDEVSIZE), where VDEVSIZE is the size of the VDEVBLOK. If a greater number of virtual devices is specified, results may be undesirable.

10. Programs developed using CMS/DOS may not be transferable directly to a DOS machine. The following considerations should be kept in mind:

    - The CMS/DOS linkage editor is designed to linkedit DOS programs for execution under CMS/DOS only. Programs transferred to a DOS machine should be re-link edited under DOS.

    - Programs assembled using CMS assembler may have incorrect ESDs. This is because the CMS assembler is not compatible with the DOS assembler. Programs transferred to a DOS machine should therefore be re-assembled under DOS.

Appendix G: A Sample EXEC Procedure for Copying DOS/VS Macros into a CMS MACLIB

You may wish to create the following EXEC procedure, DOSMAC, which will aid you in creating a DOS macro library under CMS.

Note: This procedure has not been formally tested by IBM; it is presented here for your convenience only.

To execute the following EXEC procedure, you must be in CMS/DOS mode. If a private source statement library is to be used, the appropriate ACCESS, ASSGN, and DLBL commands must be issued, specifying the DOS/VS disk on which that library resides. The procedure creates a DSERV listing on your CMS disk and uses the source statement directory listing to create an EXEC file that issues a separate ESERV command for each DOS/VS macro. You then can use the CMS Editor to delete all the ESERV commands for macros you do not wish to move at this time. The procedure then creates a CMS macro library with a MACLIB filename specified by you. If you do not specify a filename, the default is DOSMAC.

Note: If you have too many DOS/VS macros to move to your CMS disk, the MACLIB build process may exceed one of the CMS file system limitations and abnormally terminate. All macros prior to the one that caused the error message probably were cataloged correctly. Reinvoke the EXEC procedure and then use the CMS Editor to delete the ESERV commands for all the macros previously cataloged. You must also specify some other filename (such as DOSMAC2) for this new macro library.

Alternatively, if you want to avoid the abnormal termination of the MACLIB build process, you may want to delete some or all of the ESERV commands for the following DOS/VS macros the first time you invoke this EXEC procedure:

BTMOD	MCRAS	SGCCWT
CDMOD	MTMOD	SGEND
DAMOD	SDMODFI	SGPMAIN
DAMODV	SDMODFO	SGPSUB
FOPT	SDMODVO	SGSVC
IOINTER	SDMODVU	COBBG
IOTAB	SDMODW	COBF2
ISMOD		

Note: Check a DSERV listing and delete the ESERV commands for the largest DOS macros first. Then manually create a second set of ESERV commands, specifying those macros not included in the first CMS MACLIB.

Creating the DOSMAC EXEC Procedure

Issue the following command to create a EXEC procedure called DOSMAC EXEC:

```
EDIT DOSMAC EXEC
```

Enter the INPUT subcommand to get into input mode and key in the following lines. Note: Do not key in the numbers along the left side of the following example. The numbers refer to notes of explanation that follow the example.

```
    &CONTROL OFF
    &GENSWT = 0
    CP PURGE RDR ALL
    CP SP 9 * CLASS A
    &TYPE ENTER THE ADDRESS OF YOUR SYSRES VOLUME ( DEFAULT = 350 )
    &READ ARGS
    &IF &INDEX EQ 0 ACCESS 350 Z
    &IF &INDEX NE 0 ACCESS &1 Z
    SET DOS ON Z  ( VSAM
    &TYPE IF YOU WISH TO ASSGN AND DLBL A PRIVATE SOURCE STATEMNT LIBRARY
    &TYPE NOW IS THE TIME ( ENTER YOUR ASSGN ). IF YOU DO NOT ENTER A NULL LI
    &READ
    &TYPE A DLBL IS ALSO REQUIRED FOR SSL
    &READ
    -MACGEN &CONTINUE
    &TYPE ENTER THE NAME OF THE MACLIB TO BE CREATED THE DEFAULT IS DOSMAC
    &READ ARGS
    &IF &INDEX EQ 0 &LIB = DOSMAC
    &IF &INDEX NE 0 &LIB = &1
 1. CP SPOOL CONS START NOTERM
    DSERV SD ( TERM
    CP SPOOL CONS STOP TERM
    CP CLOSE 9
    READ $ESER EXEC
 2. COPYFILE $ESER EXEC A $ESERV EXEC A ( LRECL 80  REPLACE
    &BEGSTACK
    DEL 9
    F CP
    DEL 5
    TOP
    C / /&1 &2/*
    FILE
    &END
    EDIT $ESERV EXEC
    ERASE $ESER EXEC
    -STACKER &CONTINUE
    &BEGTYPE

 3. IF YOU WISH TO ALTER THE LIST OF MACROS NOW IS THE TIME TO DO SO

    YOU MAY BYPASS ALTERATION BY ENTERING A NULL LINE

    OR ELSE ENTER A NON-BLANK CHARACTER TO BEGIN ALTERATION

    ALTERATION IS ACCOMPLISHED VIA EDIT'ING THE EXEC FILE CONTAINING THE MACR

    YOU MUST ISSUE THE EDIT SUBCOMMAND FILE TO RE-ENTER THIS EXEC AND CONTINU

    &END
    &READ ARGS
    &IF &INDEX NE 0 EDIT $ESERV EXEC
    &CONTROL ALL
```

```
4.  EXEC $ESERV &STACK SPACE
    ASSGN SYSIN A
    ASSGN SYSLST PRINTER
    ASSGN SYSPCH PUNCH
    CP SPOOL D TO *
    &CONTROL ALL
    -GETNEXT &CONTINUE
    &READ ARGS
5.  &IF &2 NE E &GOTO -STAKTST
6.  &STACK LIFO FILE
    &STACK LIFO C /$/ / 4
    &STACK LIFO TOP
    &STACK LIFO I $DSPCH &3
    EDIT &3 ESERV
7.  EXEC ESERV &3
    ERASE &3 ESERV
8.  READ &3 MACRO
    &STACK LIFO FILE
    &STACK LIFO DEL
    &STACK LIFO BO
    &STACK LIFO DEL
    &STACK LIFO L /CATALS/
    EDIT &3 MACRO
    &IF &GENSWT NE 0 &GOTO -MACADD
    &GENSWT = 1
    MACLIB GEN &LIB &3
    ERASE &3 MACRO
    &GOTO -STAKTST
    -MACADD &CONTINUE
9.  MACLIB ADD &LIB &3
    ERASE &3 MACRO
    &IF &READFLAG EQ STACK &GOTO -GETNEXT
    -FINALE &CONTINUE
    &STACK QUIT
    &BEGTYPE
     THE MACLIB &LIB HAS BEEN CREATED AND THE FOLLOWING IS A MAP OF THE LIBRARY
    &END
    &STACK MACLIB MAP &LIB ( TERM
    &EXIT
    -STAKTST &CONTINUE
    &IF &READFLAG EQ STACK &GOTO -GETNEXT
    &GOTO -FINALE
```

Notes:

The following notes refer to the sample EXEC procedure shown above.

1. The output of the DSERV command is spooled to your virtual card reader and is read in as $ESER EXEC.

2. The $ESER EXEC file is copied, edited, and formatted as a CMS EXEC file. All DSERV header and trailer lines are deleted.

3. If you wish to delete any of the generated ESERV commands, enter any nonblank character. If you do not wish to delete any ESERV commands (or after you have deleted them), enter a null line.

4. Stack the remaining lines of the $ESERV EXEC in the console stack.

5. Read a line from the console stack and check that the first letter begins with E (for ESERV). If not an E, ignore the line and read the next one.

6. If it is an E, create a DSPCH fn for this macro. Note: PUNCH or DSPLY may be substituted for DSPCH.

7. Execute the ESERV command. The de-edited macro is spooled to your virtual card reader.

8. Read the macro file onto the CMS disk. Delete the CATALS statement.

9. Add the macro to the indicated CMS macro library.

For a large macro library, the ESERV process may take a substantial length of time, up to several hours.

For a detailed description of the ESERV command, refer to the _VM/370 CMS Command and Macro Reference_. For more information on how to use the ESERV command, see "Appendix D: Sample Terminal Session for DOS Programs" in the _VM/370 CMS User's Guide_.

For a detailed description of the DOS/VS ESERV control statements, refer to the _Guide to the DOS/VS Assembler_, Order No. GC33-4024.

Index

The entries in this Index are accumulative. They list additions to this publication by the following VM/370 System Control program products:

- VM/370 Basic System Extensions, Program Number 5748-XX8
- VM/370 System Extensions, Program Number 5748-XE1

However, the text within the publication is not accumulative; it relates only to the SCP or program product that is installed on your system. Therefore, there may be topics and references listed in this Index that are not contained in the body of this publication.

Technical Newsletter

This Newsletter No.	GN25-0837
Date	April 1, 1981
Base Publication No.	GC20-1801-10
File No.	S370-34 (VM/370 Release 6 PLC 17)
Prerequisite Newsletters/ Supplements	GN25-0776

IBM Virtual Machine Facility/370: Planning and System Generation Guide

This Technical Newsletter contains replacement pages for _VM/370 Planning and System Generation Guide_ to support Release 6 PLC 17 of the IBM Virtual Machine Facility/370.

Before inserting any of the attached pages into the _VM/370 Planning and System Generation Guide_ read _carefully_ the instructions on this cover. They indicate when and how you should insert pages.

Pages to _be Removed_	Attached Pages _to be Inserted*_
Title, Edition Notice	Title, Edition Notice
Preface iii-iv	Preface iii-iv
Contents vii-xiv	Contents vii-xiv
Summary of Amendments xvii-xx	Summary of Amendments xvii-xx
53-54	53-54
73-74	73-74.2
85-90	85-90.2
103-104	103-104
107-108	107-108
111-112.2	111-112.2
143-146	143-146
150.1-152	150.1-152
155-158	155-158
167-168	167-168
183-184	183-184
219-224	219-224
229-230	229-230
235-236	235-236
247-250	247-250
259-260	259-260
269-270	269-270.2
329-330	329-330
337-338	337-338
349-350	349-350
353-358	353-358.2
369-372	369-372.2
375-380	375-380
385-386	385-386
401-412	401-412
425-426	425-426
431-432	431-432.2
441-442	441-442
Index 447-478	Index 447-478
Reader's Comment Forms	Reader's Comment Forms

*If you are inserting pages from different Newsletters/Supplements and _identical_ page numbers are involved, always use the pages with the latest date (shown in the slug at the top of the page). The page with the latest date contains the most complete information.

IBM Corporation, Programming Publications, Department G60,
PO Box 6, Endicott, New York 13760

The AXSLINKS COPY file is a list of 1 to 64 GENLINK macro statements. The GENLINK macro defines the attributes of a link. The first GENLINK macro in the AXSLINKS file must contain the ID of the local RSCS station. You must also code the TYPE=driverid operand with a valid filename on this first GENLINK macro. You should code additional GENLINK macros, with no operands, for links you may want to define temporarily during an operating session.

The format of the GENLINK macro is:

```
GENLINK  ID=linkid,TYPE=driverid[,CLASS=c]
                                [,KEEP=holdslot]
                                [,LINE=vaddr]
                                [,TASK=taskname]
```

__where:__

ID=linkid is a 1- to 8-character alphameric location ID of the remote location to be served by the link. If this operand is not specified, the ID defaults to "undefined."

TYPE=driverid
: is a CMS filename of a file which is the TEXT file for the line driver program to be used to process files for the link. The appropriate line driver program to be specified depends on the type of remote telecommunications facilities to be used.

  The TYPE operand must be specified if ID=linkid is coded. If the TYPE operand is omitted, TYPE defaults to "undefined".

CLASS=c is the spooling class(es) of the files which can be processed by the active link. You can specify up to four spooling classes (single alphameric characters from A to Z and 0 to 9) with no intervening blanks, or *, which means all spool file classes may be processed. If the CLASS operand is not specified, the default is "*".

KEEP=holdslot
: is a decimal number from 0 to 16 which designates the number of virtual storage file tag slots to be reserved for exclusive use by the link. If the KEEP operand is omitted, a default "holdslot" value of 2 is assumed.

LINE=vaddr
: designates the virtual device address of a permanent telecommunications line port to be used for processing files on the link. If the LINE operand is omitted, the default is "undefined".

TASK=taskname
: is a 1- to 4-character alphameric identifier. It specifies the task name to be used by the line driver associated with the link. If the TASK operand is omitted, the default is "undefined".

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

Summary of Amendments

This Technical Newsletter incorporates minor technical and editorial changes.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

Technical Newsletter

This Newsletter No. GN25-0776
Date March 3, 1980
Supplement No.
Base Publication No. GC20-1801-10
File No. S370-34 (VM/370 Release 6 PLC 9)
Prerequisite Newsletters/Supplements None

IBM Virtual Machine Facility/370: Planning and System Generation Guide

This Technical Newsletter contains replacement pages for _VM/370 Planning and System Generation Guide_ to support Release 6 PLC 9 of IBM Virtual Machine Facility/370.

Before inserting any of the attached pages into the _VM/370 Planning and System Generation Guide_, read _carefully_ the instructions on this cover. They indicate when and how you should insert pages.

Pages to be Removed	Attached Pages to be Inserted*
Title, Edition Notice	Title, Edition Notice
Preface iii-vi	Preface iii-vi
Contents vii-xiv	Contents vii-xvi
Summary of Amendments xv-xviii	Summary of Amendments xvii-xx
41-48	41-48.4
111-112	111-112.2
119-124	119-124
143-144	143-144
147-150	147-150.2
153-154	153-154
209-210	209-210
291-292	291-292
313-314	313-314
Index 447-478	Index 447-478

*If you are inserting pages from different Newsletters/Supplements and _identical_ page numbers are involved, always use the pages with the latest date (shown in the slug at the top of the page). The page with the latest date contains the most complete information.

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

Summary of Amendments

This Technical Newsletter incorporates changes reflecting new and updated information in support of the following:

- IBM 3101 Display Terminal and miscellaneous maintenance updates.

  Note: Please file this cover letter at the back of the base publication to provide a record of changes.

IBM Corporation, Publications Development, Department D58, Building 706-2,
PO Box 390, Poughkeepsie, New York 12602

Printed in U.S.A.

IBM Virtual Machine Facility/370:
Planning and System Generation Guide
GC20-1801-10

READER'S COMMENT FORM

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

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

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

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

• What is your occupation? ____________________

• How do you use this publication:

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

Your comments:

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

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

GC20-1801-10

Reader's Comment Form

Cut or Fold Along Line

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

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

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

Fold | Fold

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

Your Name ______________________________

Company Name ____________________ *Department* __________

Street Address ______________________________

City ______________________________

State ____________________ *Zip Code* __________

IBM Branch Office serving you ______________________________

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601

GC20-1801-10

IBM VM/370: Planning and System Generation Guide Printed in U.S.A. GC20-1801-10

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 1059

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601