GC20-1801-0

Systems

IBM Virtual Machine Facility/370: Planning Guide

This publication is intended for those responsible for the planning and installation of a VM/370 system. It includes information on virtual machine concepts, operating systems in a virtual machine, and planning considerations for a VM/370 installation. The publication *IBM Virtual Machine Facility/370: Introduction*, Order No. GC20-1800, is a prerequisite for understanding this planning guide.

Preface

This publication provides the people responsible for planning and installation of a VM/370 system with the information they need to determine the facilities and support personnel that will be required.

All readers should have a general understanding of System/370 data processing techniques. Those responsible for installing and operating the VM/370 system should have a more detailed knowledge about teleprocessing systems. In addition, all users of this book should be familiar with the information contained in the publication *IBM Virtual Machine Facility/370: Introduction*, Order No. GC20-1800.

This book is divided into four main sections, plus Appendixes. "Section 1: Introduction" contains a brief description of VM/370 and its operating components.

"Section 2. CMS Considerations" supplies the reader with detailed information about the Conversational Monitor System (CMS), a virtual machine operating system that is part of VM/370.

"Section 3. Operating Systems–General Information" discusses several important considerations for all virtual machine operating systems.

"Section 4. Pre-System Generation Planning" describes the facilities and planning required to prepare to install a VM/370 system.

The Appendixes provide further details about particular aspects of VM/370, and are titled accordingly.

When the term 3330 is used in this publication, it refers to the IBM 3330 Disk Storage, Model 1 or the IBM 3333 Disk Storage, Model 1.

First Edition (August 1972)

Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest System/360 and System/370 SRL Newsletter, GN20-0360, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications Dept. 545 Technology Square, Cambridge, Massachusetts 02139. Comments become the property of IBM.

Contents

FIGURES

Section 1. Introduction

The IBM Virtual Machine Facility/370 (VM/370) System Control Program is composed of the control program (CP), which manages the resources of a single computer such that multiple computing systems appear to exist, and the Conversational Monitor System (CMS), which provides general interactive computing facilities. Figure 1 illustrates these relationships. Each one of the multiple computing systems created and controlled by CP is called a "virtual" computing system. Each virtual computing system, or virtual machine, is the functional equivalent of an IBM System/370.

The work to be done by the virtual machine is scheduled and controlled by some System/360 or System/370 operating system. The concurrent execution of multiple virtual machines is managed by the control program of VM/370.

VM/370

CP	CMS

Figure 1. VM/370 and Its Functional Components

A good example of a virtual machine operating system is the Conversational Monitor System (CMS), a major component of VM/370, which has been specifically designed to run in a virtual machine under the control of VM/370. CMS is essentially a time-shared system that depends upon the control program component of VM/370 to supply a virtual machine environment and real CPU time allocation. CMS provides, at a remote terminal, an extended range of conversational capabilities: creation and management of many types of files: compilation, testing, and execution of problem programs; and execution of application programs.

VIRTUAL MACHINE DEFINITION

A virtual machine, as implemented by VM/370, is the functional equivalent of an IBM System/370 and its associated devices. The operation of a virtual machine is controlled from a remote terminal (such as an IBM 2741 Communication Terminal), which acts as the system console of the virtual machine. The remote terminal has the capability of simulating all of the functions normally performed by the buttons, switches, and lights on the system control panel and the system console keyboard.

A user's virtual machine configuration is defined by his entry in the VM/370 user directory file.

SIMULATING THE SYSTEM CONTROL PANEL AT A REMOTE TERMINAL

On a real computing system, the system control panel contains the switches and indicators necessary to operate and control the system. On a virtual computing system, CP console functions (or commands) allow the user to control his virtual machine from the remote terminal, in the same way a system operator controls a real machine.

Virtual machine execution can be stopped at any time by pressing the terminal Attention (ATTN) key on the IBM 2741 Communication Terminal (RESET on the IBM 1050 Data Communication System).

Virtual machine execution can be restarted by typing the CP BEGIN command or by using the RESTART option of the CP SYSTEM command.

Using the CP STORE and DISPLAY commands, the operator of a virtual machine can store and display information in his virtual storage, registers, and status words such as the Program Status Word (PSW) and Channel Status Word (CSW).

Virtual storage can be cleared to binary zeros using the CLEAR option of the CP SYSTEM command. The IPL command enables the operator of a virtual machine to load operating systems.

Virtual I/O Operation

Other CP commands control virtual machine I/O operations. The READY command allows the user to simulate a device end interruption for a specified virtual device. The NOTREADY CP command enables the user to simulate a device changing from the ready to the not ready state for the virtual device specified.

The REWIND CP command allows the user to rewind a real tape drive attached to his virtual machine. The RESET console function enables him to clear all interrupts from the specified device.

Spooled Device Operation

Additional operation capabilities are available for spooled files. A user may determine via the CP QUERY command the file name, type, originating USERID, and all other attributes of files in his spooled reader, printer, or punch. The user can then place these files in a specific order (ORDER CP command) or remove unwanted spool files (PURGE CP command). The SPOOL command is available to alter the spooling control options or spool class in effect on a given device.

Each user is provided with one or more CP command privilege classes as part of his entry in the VM/370 user directory. Associated with each class are the group of commands which are permitted for that user. A user's privilege classes cannot be dynamically modified, but are under control of the system operator. The DIRECTORY service program may be invoked by the system operator to establish or change the privilege classes.

Commands are classified such that some are not available to the general user. Generally these commands are used to operate the real machine; such commands are those dealing with: real resource allocation, system control, spooling device control, and system programmer maintenance console functions.

A complete list of CP commands available to the various classes of users can be found in Appendix C.

Sending and Receiving System Console Messages

Virtual I/O to an online operator's console such as the IBM 3210 Console Printer-Keyboard Model 1 or 2, IBM 3215 Console Printer-Keyboard Model 1, or IBM 1052 Printer-Keyboard Model 7 is translated by CP into sequences for the remote terminal from which the user has accessed VM/370. The user enters operating system data, such as SET DATE or VARY, at his remote terminal, as though it was the online system console. No change is required to the operating system running in the virtual machine.

The REQUEST and CANCEL keys of the online typewriter are simulated by the ATTN key on the IBM 2741 or RESET LINE on the IBM 1052.

If a user's virtual machine requires an online operator's console other than one specifically supported by VM/370 (for example, an IBM 2250 Display Unit used as a system console), he will also need a VM/370 terminal (to access VM/370, simulate his system control panel, and control his virtual I/O).

Changing a User's Virtual Machine Configuration

CP commands provide the user with the ability to dynamically change the system configuration. This is desirable for the user who can run with a given I/O device configuration most of the time, but requires a different configuration for certain applications. Using the CP commands, the user can expand and change his I/O configuration without having to make permanent changes to his user directory. The changed configuration exists only for the current terminal session, or until the user specifically releases the I/O devices.

The spooling control options in effect for a given spooling device or class may be altered using the SPOOL command. The TERMINAL command allows the user to specify or alter the options used to control his terminal's input and output processing.

The CP DEFINE command allows the user to add to or alter an already logged-in virtual machine configuration. With this command, the user can add spooled devices (card readers, card punches, printers), 'temporary' disks, and dialable communication lines to his configuration. He can also change the virtual addresses of devices in his virtual machine, and redefine the size of his virtual storage. (The value specified becomes the new virtual storage size and cannot exceed the value specified by MSTOR (maximum allowed storage) in the user's directory entry).

A device may be removed from a user's configuration before he logs out by issuing a DETACH command. A user may detach any of his own devices, or the VM/370 system operator may detach dedicated as well as temporary real devices from the user. In this case, the user is informed of the operator's action. The detached devices become non-accessible to the virtual machine.

Minidisks

A minidisk is a special logical subdivision of a physical disk pack with its own I/O device address, disk label, and some user-determined number of contiguous cylinders (starting with virtual cylinder zero).

The user can add minidisks to his virtual machine using the LINK or DEFINE commands. Further information on minidisks may be found in "Section 3: Operating Systems—General Information."

Dedicated Real Devices

The system operator may dedicate a real device of any type to a single user by issuing the ATTACH command. The user of a virtual machine cannot issue an ATTACH command unless he is assigned privilege class B in the VM/370 user directory. Non-sharable devices such as tape drives must be attached to a user in order to be dynamically added to his virtual machine.

A real non-DASD device may not be attached to one user if it is currently in use by another user (dedicated, or contains nonsharable minidisks in use) or by the system (wholly in use by VM/370 for paging, spooling, etc.).

Section 2. CMS Considerations

Operation and Support

The Conversational Monitor System (CMS) is the major subsystem of VM/370. It provides a comprehensive set of conversational facilities to the user, the operator of the CMS system. It 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 supports the virtual machine device shown in Figure 2.

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 remote terminals are 2741's. The control program of VM/370 (CP) 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, 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 the CP IPL command is entered. CMS remains resident until another IPL command is entered or until the user logs out. 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 user disk. Files that the user wishes to retain for use across terminal sessions are stored on one of the user's disks. Information stored on a user's disk remains there until it is erased by the user. An exception is the temporary disk, which may be allocated by specifying a TEMP disk at address 192 in the user directory. Files written on this disk are lost at the end of the terminal session. CMS commands and input files may be entered into the system from the remote terminal. Output files, program results, and error and prompting messages may be directed back to the terminal.

The virtual card reader may be used as the input medium for files, source decks, and data to be processed by

Virtual IBM Device	*Virtual Address**	*Symbolic Name*	*Device Type*
3210,3215,1052	009	CON1	System Console
2314,3330	190	DSK0	System disk (read-only)
2314,3330	191	DSK1	Primary disk (user files)
2314,3330	192	DSK2	Disk (user files)
2314,3330	ccu**	DSK3	Disk (user files)
2314,3330	ccu	DSK4	Disk (user files)
2314,3330	ccu	DSK5	Disk (user files)
2314,3330	ccu	DSK6	Disk (user files)
2314,3330	ccu	DSK7	Disk (user files)
2314,3330	ccu	DSK8	Disk (user files)
2314,3330	ccu	DSK9	Disk (user files)
1403,3211,1443	00E	PRN1	Line printer
2540,2501,3505	00C	RDR1	Card reader
2540,3525	00D	PCH1	Card punch
2401,2402,2403, 2415,2420,3420	181-4	TAP1–TAP4	Tape drives

*The device addresses shown are those that are pre-assembled into the CMS resident device table. These need only be modified and a new device table made resident to change the addresses.

**The virtual device address (ccu) of a disk for user files can be any valid System/370 device address, and can be specified by the CMS user when he activates a disk. If the user does not activate a disk immediate after loading CMS, CMS automatically activates his A-disk at virtual address 191.

Figure 2. Devices Supported by a CMS Virtual Machine

user programs. The virtual card punch may be used for user output files, language processor object decks, and various other types of data. The virtual printer may be used for user 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 user directory entry for a CMS virtual machine.

```
USER USER 1 PASSWORD
  ACCOUNT NUMBER BIN7
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    LINK CMSSYS 190 190 R
    MDISK 191 2314 71 10 UDISK1 W RPASS WPASS
```

Additional devices may be included in the user directory or added dynamically to the configuration as needed.

The CMS command language permits the remote terminal user to converse with CMS. With this command language, the following functional areas may be used:

1. Language compilers.
2. A System Assembler.
3. CMS File management system.
4. Context editing and line editing.
5. Execution control.
6. Debugging capability.

Additionally, the CP commands available to all virtual machines under VM/370 can be invoked directly from CMS. Using these CP commands, messages can be sent to the operator or to other users, the virtual machine's configuration can be dynamically modified, and spooling facilities can be invoked. To the CMS user, the facilities of CP and CMS together will appear like those of a single integrated system.

To use CMS, the user must first gain access to a virtual machine via the CP LOGIN procedure. Once access has been gained, he must then IPL CMS. The user may then enter commands or data from the remote terminal (virtual operator's console). Each command, upon completion, returns control back to the user by posting a READ to the terminal keyboard.

The following is an example of a user gaining access to CMS:

```
vm/370 online
login user1
ENTER PASSWORD:
(Note: Printing of password inhibited for security.)
LOGIN AT 11:03:18 EST THURSDAY 12/21/72

ipl 190
CMS VERSION 1.0 11/30/72 08.30
```

The user can now issue any CMS command.

CMS Program Language Facilities

The language processors that can be executed under CMS include:

The VM/370 BASIC Language Processor
The VM/370 System Assembler
OS PL/I Optimizing Compiler
OS FORTRAN IV (G1, Code and Go, and H Extended)
OS Full American National Standard COBOL

The System Assembler and BASIC are distributed with VM/370. The other compilers are Program Products that must be ordered separately. CMS executes the OS compilers via interface modules. CMS commands are provided to invoke the OS compilers within the conversational environment of CMS.

For example, to compile a COBOL source program, you could issue the COBOL command as follows:

```
cobol proga (deck)
```

where PROGA is the filename of a CMS file containing fixed-length, 80-character records. The filetype is COBOL. The compiler output is controlled by a set of user specified options. The list of options specified follows the filename. The DECK option selected in the example causes the language processor output (object code) to be spooled to the virtual machine punch. A default value is assumed for any option not specified.

When executing the OS compilers under CMS, the normal mode of operation is to:

- Run the compilation to completion.
- Type out any diagnostics at the terminal.
- Generate a CMS disk file with the same filename as the source program and a filetype of TEXT, which contains the object deck created by the assembler or compiler.
- Direct the printed output of the assembler or compiler to the spooled printer or to a disk file with a filetype of LISTING.

The filename of files created by the assembler or compilers running under CMS is equal to that of the file being compiled (PROGA in the example).

Disk work files required by the assembler or OS compilers under CMS are automatically created during compilation and erased at the end of compilation. No cataloguing or scratching of data sets is required.

Object programs (TEXT files) produced under CMS and under OS in real or virtual machines may be executed under CMS if they do not utilize certain OS functions not simulated by CMS (OS macro functions that are simulated are discussed in "Execution".) Object programs using non-simulated OS macro functions must be transferred to an appropriate real or virtual OS machine for execution.

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

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.

Multi-tasking
Teleprocessing file support
ISAM
Backwards attribute
Spanned records for buffered files
Sort-merge
Checkpoint-restart
ASCII data sets
Track overflow

The CALL-OS BASIC Language Processor (Version 1.1), modified for use under CMS, is distributed with the VM/370 system and can be executed under CMS. The CMS BASIC command is provided to invoke the processor to compile and execute a BASIC source file. For example:

```
basic calc2 (long)
```

where CALC2 is the filename of a CMS disk file with a filetype of BASIC.

Note: Class A maintenance will be provided by the VM/370 Central Programming Service.

CMS File System

CMS provides interfaces with virtual disks, tapes, and unit record equipment. Most CMS commands assume that files are stored on disk. This means that files stored on media other than disk must be transferred to disk before issuing a CMS command that uses them. Similarly, most CMS commands that create files do so on disk (such as listing files from a compilation). In many cases the user will want to transfer these files to cards or tape, or to print them. Commands that handle the transfer of files between disk and other media are discussed under "CMS Tape Support" and "CMS Unit Record Support".

CMS Disk File Conventions

CMS can manage up to ten virtual disks (created on 2314, 2319, or 3330 storage devices) for each user. One of these is the CMS system disk (S-disk), maintained as a read/only shared minidisk, which normally has a device address of 190. User files may be accessed from up to nine active CMS disks. Usually, one of these is a read/write disk called the A-disk, which normally has a device address of 191.

CMS Disk Format

CMS disks are formatted into 800-byte physical records or blocks. A 3330 disk volume holds 14 physical blocks per track, while a 2314 or 2319 disk volume holds 15 physical blocks for every two tracks.

Each CMS disk has a six-character label (or volid) associated with it. Record 3 (on cylinder 0, track 0), of a CMS disk includes a ten-byte label, consisting of the following:

1. Four characters–CMS=
2. Six characters–desired label (blank filled if less than 6 characters; truncated if more than 6 characters)
3. Remaining bytes of the record are all binary zeros.

The CMS FORMAT command is used to initialize a disk area in the CMS format and to write a label onto a disk. All CMS disks must be formatted before being used for the first time.

Disks do not need to be formatted for each session unless the disk is a temporary disk dynamically added to the virtual machine configuration via the CP DEFINE command. Dynamically defined disks must be formatted each time they are added to the virtual machine configuration.

Each disk has an automatically maintained file directory that keeps an inventory of all files on that disk. The file directory is updated after each CMS command that changes the status of files on that disk. A file directory has a name in the form of a single alphabetic letter A, B, C, D, E, F, G, S, Y, or Z, which corresponds to the mode letter of the ten possible user disks.

The relationship between a file directory on a physical disk and the file directory name is established by the user during his terminal session via the ACCESS command.

The data extent of a particular CMS file is limited to one virtual disk. The file management system limits the number of files on any one virtual disk to 3400, and the number of records in a particular CMS file to 65,000.

File Format

Logical records are imposed upon constant physical blocks as shown in Figure 3.

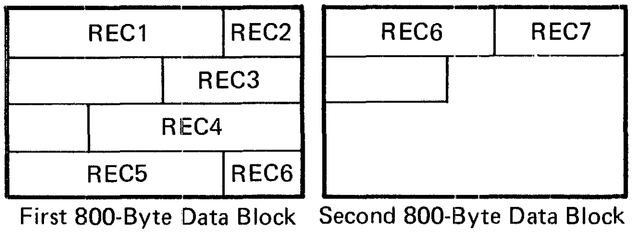


Figure 3. File Format of Logical Records

Data files may be fixed-length or variable-length, but a mixture within a file is not permitted. Where necessary, items are begun in one block and continued in the next, for example, record 6 in Figure 3. Physical blocks are randomly assigned to a file as space is required, and released when no longer in use. The physical blocks assigned to a file may be discontiguous, and are chained together by a series of pointers called a chain link, anchored in a File Status Table.

AUTOMATIC SPACE ALLOCATION FOR TEMPORARY WORK FILES

Space for required files (compiler work files, LISTING files, and object modules), as well as user created files, is automatically allocated by CMS. As a file grows, its space is automatically expanded, and it is automatically contracted if the space requirements should be reduced.

Disk Access

Disks may 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.

To access a disk, the user must:

1. Identify a disk as part of his virtual machine configuration. This is accomplished via the VM/370 user directory description for his virtual machine configuration, or by issuing a CP LINK or DEFINE command.
2. Identify the disk to CMS and assign it a file directory name. This is accomplished by issuing the ACCESS command after the user has loaded CMS. The CMS ACCESS command associates a particular disk with a given file directory name and, optionally, specifies which files on the disk are to be used and specifies the disk as read/only.

The following example shows how a user might add a temporary disk and a user disk to a CMS virtual machine.

```
ipl cms
cp link dept637  230  197  r  12601
cp define t3330a as 192  cyl 5
format 192 d
access 197 b
```

Note: The CP commands are noted by the prefix "cp" in the example.

The user first issues the LINK command to add a device at virtual address 197 to his virtual machine. (The disk added is defined in the VM/370 user directory for a virtual machine with a USERID of DEPT637 as device address 230 with a read password of 12601.)

The DEFINE command adds temporary disk space (from a CP pool of such space) to the user's virtual machine at address 192.

The CMS FORMAT command initializes the temporary disk area (192) in the CMS format.

The ACCESS command activates the disk at virtual address 197 (similar to VARY ONLINE in OS) and assigns the disk the file directory name B. The importance of the directory name is explained under "File Access–Order of Search".

If ACCESS is not the first command entered after CMS is loaded, as in the example above, an automatic ACCESS is performed to access a disk at device address 191 as the A-disk.

Both CP and CMS can control read/write access to disks, as is illustrated in Figure 4.

Access allowed by CP is determined by the user directory entry or the form of LINK issued by a terminal user for a particular disk.

The disk access allowed by CMS is determined by the CMS ACCESS command, and is explained further in "File Access–Order of Search".

CP Access	*CMS Access*	
	Read/ only	*Read/ write*
Read/only	Read/only	Read/only
Read/Write	Read/only	Read/Write

Figure 4. CP and CMS Disk Access

FILE IDENTIFIERS

Files in CMS are identified with three designators. First is a filename. Second is a filetype, which can imply certain specific characteristics or attributes to the CMS file management routines. Third is a filemode, which describes the location and the access characteristics of the file.

Filename

Filename (abbreviated as fn) may be any combination of from one to eight EBCDIC alphameric characters: the letters A-Z and a-z, the numerals 0-9, or one of the special characters $, #, or @. User files may be assigned any filename the user wishes, since filenames in themselves do not have any special implication in CMS.

For system files (residing on the S-disk), the filename of the command module is the name used to invoke a specific command.

Filetype

Filetype (abbreviated as ft) may be any combination of from one to eight EBCDIC alphameric characters: the letters A-Z and a-z, the numerals 0-9, and the special characters $, #, or @. Certain filetypes imply certain characteristics to CMS. The user may assign any filetype he wishes to any file. He should be aware that certain commands assume specific filetype designators and cannot successfully execute if the contents of the file are in any form other than that which the assigned filetype implies.

For example, the LOAD command reads from disk one or more specified files with a filetype of TEXT. The TEXT files specified must consist of relocatable object code such as that produced by the language processors. If LOAD was issued specifying a file with a filetype of TEXT, but this file was in fact a FORTRAN source program, LOAD would not execute successfully.

When no filetype is specified in a command, some CMS commands assume a filetype for the specified filename.

Many CMS commands create files for the user for temporary use, and assign specific filenames or filetypes (or filename-filetype combinations) to these files.

For example, execution of the PL/I Optimizing Compiler under CMS causes two files to be created, one with a filetype of TEXT, and one with a filetype of LISTING. The filename of each file will be the same as that of the PL/I source program compiled. For example, the command:

```
pliopt proga
```

compiles the PL/I source program named PROGA. If CMS is requested to list all the files of PROGA by the following command:

```
listfile proga  *
```

CMS responds with:

```
PROGA TEXT       A1
PROGA PLIOPT     A1
PROGA LISTING    A1
```

Compiling PROGA (filetype must be PLIOPT) causes PROGA TEXT and PROGA LISTING files to be created on the user's disk. Additionally, the compiler creates three temporary work files, PROGA SYSUT1, PROGA SYSUT2, and PROGA SYSUT3, which are erased when then PL/I compilation completes.

Note that files with the same filename form a logical group (that is, all files associated with PROGA), and similarly, all files with the same filetype form a logical group (that is, all compilation listings, all object programs).

The reserved or special filename-filetype combinations may be assigned by the user to any file, but he should be aware that:

1. If a new file is created with a filename-filetype combination that is the same as to that of an existing file on the CMS disk, the existing file is deleted without warning and replaced by the new file.
2. When searching for a file, CMS uses the first file found with the specified filename-filetype. If files exist on different disks with the same filename and filetype, the one chosen is determined by the order of search if a filemode is not specified.

Additionally, certain filename-filetype combinations have special meaning when executing user-written programs under CMS. For example, when executing a program written in BASIC, input and output files must have a filetype of 'BASDATA', with a filename as designated by the open statement. The record format is the standard CALL-OS BASIC data record format (that is, variable length of up to 3440 characters).

Filemode

The filemode designator is a two character field. The first character must be one of the ten letters A, B, C, D, E, F, G, S, Y, and Z. The second character must be numeric. The letter denotes the file directory on which the file resides. The numeric character specifies the mode by which the data is to be accessed.

Valid numeric values for filemode are:

0 Private: the file cannot be accessed unless the disk access is read/write.
1 File is accessible in a read/write status (if the disk is also accessible in read/write mode).
2 File is accessible in read/only mode. The file will not be written or updated in place, but it may be erased or replaced.
3 File is to be erased from a read/write disk after being read. This is normally used for temporary work files.
4 The file is maintained in the format produced by the simulated OS access methods under CMS.
5 Same as 1.
6-9 Reserved for future use.

Note: Commands in which a filemode may be specified, but is left blank (that is, the default is desired), or is entered as an asterisk, result in the following order of search:

Specified (FM)–that disk
Default (blank)–primary disk
Asterisk (*)–all disks

The filemode for a particular file may be indicated in one of the four ways shown in Figure 5.

The following rules have been established to govern the action of all commands relative to the four methods of specifying the filemode (fm; *, =; blank; null) and the two possible access modes (Read, Write).

Explicitly specified	fm	Filemode entered
Implicitly specified	* =	* or = entered
Specified by default	default	Filemode left blank
Not specified	null	Command issued which cannot contain a filemode specification

Figure 5. Methods of Filemode Specification

Reading:

fm when a mode designation such as B2 is specified within the command line, the unique file directory and all its extensions are searched. If the specified directory is itself an extension, its parent is *not* searched.

default if the command entered does not contain a filemode (that is, the user chooses to leave the filemode specification blank), the disk with a mode designation of "A" is searched.

*,null a search is made of all active disks in the predetermined, hierarchical order.

Writing:

fm when a mode designation such as A1 is specified, that unique disk is written upon. If the specified disk is an extension, its parent is *not* written upon. If the specified mode letter designates a read/only disk, the command aborts.

= when "=" is entered, it is used for commands which have read/write functions to indicate that writing should be done to the disk from which the reading took place.

default, null,* for commands with read/write functions, writing is attempted to the disk or the parent of the disk from which reading was done. If no room is available on either of the disks mentioned or if both disks are read/only, writing is done only to the primary disk.

Creating a file with an identifier equal to that of an existing file on the same disk causes the existing file to be erased and replaced by the new file. No warning message is sent to the user, since CMS assumes that the user intends to replace the old file.

FILE ACCESS–ORDER OF SEARCH

Many CMS commands allow the user to specify which file directory (and therefore which active disk) to search for the file. This is done by including the filemode as part of the specified file identifier in the CMS command line. When the user cannot specify the file directory name, active disks are searched in a predetermined, hierarchical order:

A, B, C, D, E, F, G, S, Y, Z

The user can modify this standard alphabetical order of search. A disk space may have a logical extension, such that a file directory may logically encompass the files of its own space plus the files of another file directory. This latter file directory is known as an extension to the former file directory. The first file directory is known as the "parent."

If A was the parent file directory and D its extension, a default file directory specification in a CMS command would cause CMS to first search the A file directory for the specified file and, if not found, then search the D-disk file directory. The modified order of search while the parent-extension relationship is in effect would be

A, D, B, C, E, F, G, S, Y, Z

Extensions are always read-only so that the user cannot write on a disk being used as an extension, even though he may be allowed write access to the disk.

Directory extensions are set by the user during his terminal session by issuing the CMS ACCESS command. For example:

ACCESS 193 D/A

Specifies that the D-disk file directory is to be a read/only extension of the A file directory. The A-disk must have been previously activated.

ACCESS 191 A/A

Specifies that the A-disk file directory is to be a read/only extension of itself. This prohibits writing on the active A-disk, even though the user would normally be allowed read/write access.

CMS LIBRARIES

CMS maintains simulated partitioned data sets which contain:

1. CMS and OS macros to be used at assembly time (MACLIB).
2. Object routines to be referred to at execution-load time (TXTLIB).

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

CMSLIB MACLIB	contains all of the CMS macros.
OSMACRO MACLIB	contains the selected OS macros from SYS1.MACLIB that are supported under CMS.
OSMACRO1 MACLIB	contains the remaining distributed OS macros from SYS1.MACLIB.
TSOMAC MACLIB	contains the OS macros distributed in SYS1.TSOMAC

A DOS user running a DOS virtual machine may punch the DOS macros on the virtual punch and create a new CMS macro library called DOSMACRO MACLIB. This library may be used to assemble DOS programs directly under CMS.

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

CMSLIB TXTLIB	contains the CMS system text library.
TSOLIB TXTLIB	contains selected TSO routines necessary to support certain features of the language Program Products.

Execution-time libraries are available with the Program Product language processors and are capable of executing under CMS.

The user can generate his own libraries and add, delete, or list entries in them via the MACLIB and TXTLIB commands. He can also specify which libraries (system and user) to use for program compilation and execution via the GLOBAL command, which allows up to eight libraries to be specified.

CMS TAPE SUPPORT

CMS supports up to four magnetic tape units at virtual addresses 181, 182, 183, and 184. They may be either 2401, 2402, 2403, 2415, 2420, or 3420 drives, or a mixture of tape drives.

Each tape-handling command allows the user to specify the mode of the tape (7-track or 9-track), density, and tape recording technique: odd parity, converter on, translator off, or odd parity, converter off, translator on.

If the mode is not specified for a 7-track tape, CMS issues a mode set indicating 7-track, 800 bpi, odd parity, converter on, and translate off. If the tape is 9-track, the mode is ignored and is assumed to be 1600 bpi for dual density drives and 800 bpi for single density drives.

As an alternative to specifying mode in each tape command, the user can issue a TAPE command that sets the mode for the tape and stays in effect until reissued. This must be done if a user program is to use tapes in other than the default mode.

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

Multi-volume tape files are not supported by CMS.

Note: These restrictions only apply to CMS users. 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 multi-reel tape files.

Tapes must be attached to the CMS user by the VM/370 operator before any tape I/O can take place.

CMS Tape-Handling Commands

Command	*Function*
TAPE	Dumps CMS formatted files from disk to tape, loads such files from tape to disk, and performs various control operations on a specific tape, such as setting tape modes, forward or backward spacing, and rewinding the tape. The TAPE command is used solely for CMS files. The files on tape are created in a unique format.
TAPPDS	Produces CMS disk files read from tapes produced by the OS IEBPTPCH service program. The tape must be in unblocked card image form. It may be unlabeled or may have a standard OS label.

Note: The MOVEFILE and DDR commands may also be used to operate on a tape.

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.

Card Reader

The READCARD command reads data records from the spooled card reader to a CMS disk. Input records of 133 or less characters are accepted. Column binary data is not acceptable. User card decks must have been read previously into the virtual reader before a READCARD command can be issued. This can be accomplished in one of two ways:

1. A card deck is placed in a real card reader and read by CP. The deck must be preceded by a special VM/370 identification (ID) card specifying the userid of the virtual machine to receive the card images. The card images are placed on a spool file in the specified virtual machine's virtual card reader. If the user is not logged on to the system, the deck remains in his virtual card reader until he logs in and issues the READCARD command.
2. One virtual machine transfers records from its virtual card punch or printer to a virtual card reader (its own or that of another virtual machine).

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. Punch stacker select is not supported, nor is punch feed read.

Printer

The PRINT command prints the specified disk file on the spooled printer. The user 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

The editing function is one of creating new files online or of modifying or selectively typing specific portions of existing files.

The CMS editor operates on fixed and variable length records. The maximum record length accepted by the editor is 133 characters. File characteristics such as record length, format, and tab locations can be specified by the user. The system includes defaults for certain filetypes (such as FORTRAN and EXEC).

Files are edited in virtual storage. The EDIT command has a SAVE subcommand that allows a user to periodically write the latest copy of the file on disk and return to edit mode at the point where the SAVE subcommand was issued. This command should be issued periodically to assure that the most current version of a file is maintained on a user's disk during a prolonged editing session.

The maximum size file that can be edited is determined by the amount of virtual storage available to the user, 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, the user may temporarily increase the size of his virtual storage.

When issuing the SAVE or FILE subcommands, the old copy of a file is not erased until after the new copy has been successfully written.

EXECUTION CONTROL

A string of user arguments may be passed to the user program from the terminal when execution is begun with the START command. If arguments are specified, the storage address of a parameter list is placed into general register 1. The parameter list is a string of double words, one argument per double word. The first argument is the entry operand. Other arguments are accessed with displacements of 8, 16, 24, and so on, from the address contained in register 1 when execution begins. For example:

```
load proga
start epg  071374
```

causes execution to begin at a control section or entry point named EPG. When execution begins, general register 1 points to a string of two double words, the first containing EPG, the second containing 071374.

If a program is to be executed frequently, the user may create a non-relocatable copy of it on a user disk. Subsequent invocation of this program causes the absolute module to be read from disk storage, ready to begin execution. This bypasses the task of having to statically relocate the object form of the program.

CMS INTERACTIVE EXECUTION

Once execution has begun, the user can interact with his program in several ways.

1. Write to operator and write to operator with reply macros in DOS or OS programs use the virtual system console (the remote terminal) to send messages to, and receive responses from the user. In this way, the user can easily write conversational application programs that can be executed from a remote terminal. He need not concern himself with the teleprocessing, terminal, or line handling considerations when designing his program. Console messages are automatically converted by VM/370 to the proper terminal and line information required by the user's remote terminal. In this way, batch-oriented problem-solving programs can be invoked from a remote terminal and run online with no changes required in the batch program.
2. The user can, before beginning execution, specify that particular input or output files are to be directed to his remote terminal. This is done via the CMS FILEDEF command. For example:

   ```
   filedef assemble terminal (upcase
   ```

   causes all program references to the file whose DCB DDNAME is designated as ASSEMBLE to be directed to the user's remote terminal. The upcase option translates all terminal I/O data to uppercase.
3. The user can create CMS card-image files using the CMS editor facilities. Those files may be transferred to the virtual card reader of an OS or DOS virtual machine for use by an application program that normally reads card input and writes printer output. The user IPLs that DOS or OS virtual machine and executes the problem-solving program (intercepting printer and punched output in a CP spool file). The user then IPLs CMS again, and after reading the proper data set onto a disk, may selectively scan the output at his terminal, or if the output is not voluminous, type it out at his terminal. As a result of his analysis, the user may change some input variables, correct the program or data, and repeat the "flip-flop" cycle. If the output printing volume is large, the user can route it to a 2780 terminal or to a high-speed CPU printer.
4. The user can stop execution of his virtual machine at any time or at any place in the program. Once the program is stopped, the user can issue CP commands to examine and change storage locations, register contents, and control words (such as the PSW and CSW). He can then restart execution at any specified point using the CP BEGIN command.
5. The user can use the DEBUG facilities of CMS, which are described under "CMS Debugging".

OS MACRO SIMULATION UNDER CMS

When in a CMS environment, and a language processor or a user-written program is executing using OS-type functions, it is not OS code that is executing. Instead, CMS provides routines that simulate the OS functions required to support OS language processors and their generated object code.

CMS functionally simulates the OS macros in a way that presents equivalent results to programs executing under CMS. Thus, while data management macros are simulated, data formats follow CMS conventions. The OS macros are supported only to the extent stated in the publications for the supported language processors, and then only to the extent necessary to successfully satisfy the specific requirement of the supervisory function.

The restrictions for COBOL and PL/I program execution listed in "CMS Program Language Facilities" exist because of the limited simulation by CMS of the OS macros.

Figure 6 shows the OS macro functions that are partially or completely simulated, as defined by SVC number.

SVC Number	*OS Function*
00	XDAP
01	WAIT
02	POST
04	GETMAIN
05	FREEMAIN
06	LINK
07	XCTL
08	LOAD
09	DELETE
10	REGMAIN
11	TIME
13	ABEND
14	SPIE
18	BLDL/FIND
19	OPEN
20	CLOSE
21	STOW
22	OPENJ
23	TCLOSE
24	DEVTYPE
35	WTO/WTOR
40	EXTRACT
41	IDENTIFY
42	ATTACH
44	CHAP
46	TTIMER
47	STIMER
48	DEQ
51	SNAP
56	ENQ
57	FREEDBUF
60	STAE
62	DETACH
63	CHKPT
64	RDJFCB
68	SYNAD
69	BACKSPACE

Figure 6. Simulated OS Supervisor Calls

OS Data Management Simulation

Even though the CMS disk format and data base organization are not compatible with OS, data sets are transferable between systems. To facilitate the execution of OS programs under CMS, the OS processing program must be presented with data in storage as OS would present it. For instance, when the OS language processors expect an access method to sequentially acquire input source records, CMS invokes its own sequential access method and passes data to the processors in the same storage format that the OS access methods would have produced. Therefore, data appears in storage as if it had been manipulated using an OS access method. The actual writing to and reading from the I/O device is handled by CMS file management.

If the access mode in the filemode specification is “4”, CMS continues to maintain the OS format within its own format on the DASD device. CMS will also simulate the specific methods of manipulating data sets. A data set may be accessed according to a scheme designated as:

direct — identifying a record by a key or by its relative position within the data set.

partitioned — seeking a named member within the data set.

sequential — accessing a record in a sequence relative to preceding or following items in the data set.

Since CMS does not simulate the indexed sequential access method (ISAM), no OS program which uses ISAM can execute under CMS. The data management macros which are simulated allow execution of programs using BDAM, BSAM, QSAM, and BPAM. The OS macros supported under CMS are:

OS Macro Function	*Usage*
GET/PUT	Access system-blocked data
READ/WRITE	Access system-record data
NOTE/POINT	Access or change relative track ad-
CHECK	dress test ECB for completion and errors

The FILEDEF Command

The CMS FILEDEF command allows the user to specify the I/O device and the file characteristics to be used by a program at execution time. In conjunction with the OS simulation scheme, FILEDEF simulates the functions of the Data Definition JCL statement.

FILEDEF may be used only with programs using OS macros and functions.

For example:

```
filedef file1 disk proga cobol a1
```

After issuing this command, user programs referring to FILE1 as the DCBDDNAM will access PROGA COBOL on the user’s A-disk.

```
fi tapein tap2 (recfm fb lrecl 50 block 100
9track den 800)
```

After issuing this command, user programs referring to TAPEIN will access a tape at virtual address 182. (Each tape unit in the CMS environment has a symbolic name associated with it, as previously described under "Operation and Support". The tape must have been previously attached to the virtual machine by the VM/370 operator.

CMS DEBUGGING

Three types of debugging facilities are available to the CMS interactive user:

1. Those provided for all virtual machines by VM/370 via the CP commands.
2. A CMS command to trace supervisor calls (SVCTRACE).
3. A CMS DEBUG environment that provides the user with online facilities for debugging programs.

The CMS command SVCTRACE allows the user to record information about SVC instructions issued while executing within CMS. Information recorded includes:

- The contents of virtual registers (general and floating point) before branching to and returning from SVC-called routines.
- Parameter strings as they exist when an SVC was issued.
- The location of the SVC instruction.
- The contents of the SVC old PSW.
- The storage locations of normal and error returns from called programs.

The CMS DEBUG environment is entered:

1. By issuing the DEBUG command.
2. Through the occurrence of an external interrupt.
3. When encountering a pre-established breakpoint (a simulated instruction address stop).

Once in the DEBUG environment, the user may use any of the DEBUG requests shown in parentheses in the following text. The user may display the contents of program status words (PSW), general registers (GPR), the channel status word (CSW) and the channel address word (CAW), and modify any of the above (SET).

Specified storage locations can also be dumped to the spooled printer (DUMP), typed (X) at the user's terminal, or modified in storage (STORE).

Up to sixteen breakpoints can be set by the user (BREAK). When a breakpoint is encountered during program execution, DEBUG mode is automatically entered and the breakpoint cleared.

The symbolic names known to the problem program are not known at time of program execution. Consequently, the facilities of DEBUG are most useful to assembler language programmers, or to those programmers who are knowledgeable about the location of their data instructions. As an operational aid, the DEFINE request allows the user to assign a symbolic name to a specific address and to refer to that address in other DEBUG requests by the assigned name.

The user either leaves the DEBUG environment and resumes execution (GO), returns to the CMS command environment (RETURN), or may halt execution and return to the CMS command environment (HX).

CMS STORAGE REQUIREMENTS

Virtual Storage

A minimum of 320K bytes for a CMS virtual machine, distributed as follows:

1. CMS nucleus–128K.
2. Application program–program-dependent or CMS disk-resident commands.
3. Loader tables–8K.

Auxiliary Storage

1. System residence for CMS–50 cylinders of 2314 or 2319, or 35 cylinders of 3330.
2. System residence for application programs (CMS commands, user programs, IBM Program Products)–program dependent.
3. Work space for application programs (CMS commands, user programs, IBM Program Products)–program dependent.

CMS COMMAND LANGUAGE EXTENSIONS

The user may enhance the functional range of CMS by adding his own commands to those in the system. A CMS command can be:

1. The name of a program resident in the nucleus or any CMS file on any CMS disk (with a filetype of MODULE).
2. The name of a file containing other CMS and CP commands and having a filetype of EXEC.

A program or file meeting either of these requirements can be invoked by entering its name from the remote terminal.

CMS Command Search

When a command is issued from the terminal, the file directories of all active disks are searched for a file with the specified filename and a filetype of EXEC. (The interrogation of the disks follows either the standard order of search, or that specified by the user if he has modified the search sequence).

If an EXEC file is not found, a check is made for abbreviations by first checking for user-defined synonyms and then standard abbreviations; if a match is found for a synonym or abbreviation, the requested command is expanded or changed to the full CMS command name, and the above searching sequence is repeated.

If the EXEC file search is not satisfied, the tables of the CMS nucleus-resident commands and transient area commands are searched for a match on the corresponding command program. If the program is located in one of the CMS tables, it is assumed to be in storagé, and control is transferred to the routine directly by a BALR instruction.

If the requested routine does not exist in storage, a LOADMOD command is issued to bring the command program into storage. In attempting to locate this program on disk, the disk directory is searched for a file with the specified filename (command name) and a filetype of MODULE, indicating that the file is in non-relocating form. The first file found which meets these requirements is loaded into storage, and control is transferred to it. If the MODULE file is not found, a check is made for abbreviations by first checking for user-defined synonyms, and then for a standard system abbreviation; if a match is found for a synonym or abbreviation, the requested command is expanded to or replaced by the original CMS command name, and the previously-described searching sequence is repeated, first searching the transient area and nucleus-resident tables, and then searching the file directories.

CMS Command Abbreviations

In order to make the entering of CMS commands easier, faster, and more convenient for the experienced user, CMS allows many command names, operands and options to be abbreviated (or truncated). Instead of entering a full command name such as ASSEMBLE, CMS allows a range of truncations from a minimum of "A" all the way up to the full word "ASSEMBLE". Most CMS commands have a minimum contraction specified. Users may create their own synonyms and abbreviations, or eliminate all of the standard CMS abbreviations.

Command abbreviations are interpreted based on the number of characters contained in the first word. These characters are looked up in a user-defined SYNONYM table (if one exists) or in a table of standard system abbreviations.

To create a SYNONYM table, the user must:

1. Create a user synonym file with a filetype of SYNONYM on one of his user disks.
2. Issue the SYNONYM command to set up a SYNONYM table, indicating the user file containing his own command names and minimum truncations allowed, and specifying whether it is to be used with or in place of standard CMS names.

THE EXEC PROCESSOR

The EXEC processor provides another type of command language extension. In some respects, an EXEC procedure provides some facilities similiar to an OS cataloged procedure. That is, it represents a sequence of commands. In this way, a single command (the filename of an EXEC file) can be used to invoke many CMS commands. EXEC files can be created by the EDIT or LISTFILE commands, or by reading an EXEC card file from the virtual card reader. They must be in fixed format, and must consist of one CMS command or EXEC control statement per card image. CMS commands must be in the same format as the command is entered from the terminal.

The EXEC processor can manipulate parameter strings, thus allowing the user to pass variable arguments from the terminal at the time the EXEC procedure is invoked.

Each CMS command in the EXEC file can have from one to thirty numeric variables. A numeric variable is made up of an ampersand (&) followed by an integer ranging from one to thirty, (that is, &1 &2. . .&30). Before the command in the EXEC file is executed, each variable is temporarily replaced by the corresponding argument specified when the EXEC file was invoked.

For example, assume that LLHS is a file whose filetype is FORTRAN. When the command EXEC FORTCLG LLHS is issued, LLHS replaces the &1 in all CMS commands in the EXEC file, FORTCLG. The file LLHS FORTRAN is compiled, and the file LLHS TEXT is loaded and executed. Note that each CMS command types at the terminal.

An example of an EXEC file to compile, load, and execute a FORTRAN program is as follows:

```
TYPE fortclg exec

FORTGI &1 &2
LOAD &1 &2 (START)

R; T=0:45/1:23 01:24:45

fortclg llhs
01:29:50 FORTGI LLHS
01:29:55 LOAD LLHS (START)
EXECUTION BEGINS.
APRIL 1968 DATA 5.320       1.920       5.600
R; T=0:55/1:44 01:30:45
```

A refinement of this concept is that the EXEC processor can manipulate command strings in a logical fashion. Logic statements can be placed in a file with the commands such that if errors occur from a command, no more commands will execute from that file. Logic statements are also provided to permit logic control during execution (&GOTO, &IF, and &LOOP). EXEC files can be nested or recursive.

The EXEC capability allows the user to develop his own command language or set of procedures. In some respects the EXEC processor is a generative system, much like a macro processor.

PROFILE EXEC Procedures

When a user initially loads CMS, an automatic search is made of the first disk to be accessed for a file with a fileid of PROFILE EXEC. If such a file exists, it is automatically executed before the first CMS command entered by the user is executed.

PROFILE EXEC is an ordinary EXEC file and, as such, may contain *any* valid EXEC-type statements. The only difference is in its name, which has special meaning that causes this automatic execution.

The facility allows the user to set up his own predetermined operating environment within CMS, and eliminates the need to enter any repetitious commands that may be required each time he starts to use CMS.

The automatic execution of a PROFILE EXEC file can be avoided by issuing the ACCESS command with the NOPROF option as the first CMS command after IPL, as follows:

```
ACCESS ccu mode (NOPROF)
```

Adding User-Written Commands

Any user program in non-relocatable form (such as one with a filetype of EXEC or MODULE) may be called directly as a command by issuing the filename (and any operands or parameters expected by the program) as an input line to CMS.

The standard command search sequence interrogates active user disks (A through G) for disk-resident commands before searching the system disk. This enables the user to substitute his own programs for disk-resident commands by creating a MODULE or EXEC file of the program with the same filename as that of the system command it is to replace.

CMS EXEC Procedures

If a flip-flop between CMS and another operating system is to be used frequently, a CMS EXEC procedure should be created and made available to users. See the previous discussion of "The EXEC Processor".

Figure 7 illustrates a simple EXEC procedure. This sample procedure causes a CMS file to be transferred to the virtual card reader. It also automatically transfers the OS printer output to the user's virtual card reader (cp spool 00E to *) so that later on, under CMS, the printer listings and storage dumps, if any, can be scanned at the terminal, or printed on the high-speed CPU printer, or remotely spooled to a 2780 Data Transmission Terminal.

```
cp close 00c
cp close 00d
cp close 00e
co stool 00d to *
punch proga job (noheader)
cp spool 00e to *
cp ipl 230
```

Figure 7. A Sample EXEC Procedure

This sample EXEC procedure, like an OS cataloged procedure, provides a simple, concise way of invoking multiple consecutive operating system facilities. The parameters are the names of those CMS files which are the components of the OS job stream. For example:

```
flipflop cobcomp lkedgo
IEE007A READY
set date=72.355,Q=(231)
start rdr,00c
start wtr,00d
start
```

COBCOMP JOB and LKEDGO JOB are the two CMS files containing the OS JCL card images, programs, and data necessary to run one or more jobs in an OS virtual machine.

When the job has completed, the user can examine the printed output by issuing the CMS command READCARD *, and then invoking the facilities of the CMS context editor. The printer output can then be printed on the CPU high-speed printer using the CMS PRINT command, or typed out at his remote terminal via the CMS TYPE command.

CMS BATCH OPERATION

The CMS Batch facility is executed in a virtual machine designated as the "Batch" machine. Unlike the normal CMS operating mode, which accepts commands from and interacts with the user's remote terminal, the Batch facility accepts commands in the form of control cards that constitute a job stream to be read from its virtual card reader.

OS and any combination of operating systems such as PCP, MFT, MVT, VS1, VS2, DOS, and DOS/VS may be executed as operating systems in a virtual machine created and controlled by VM/370. Programs or systems to be run in a virtual machine, however, cannot include any hardware or software timing dependencies. For example, if I/O devices (or the programs that control them) require program responses within a defined time for correct data transmission, they may not work correctly. Execution of programs under VM/370 which require dynamic modification of channel programs is allowed only in certain special cases; they can be executed in a virtual machine with the special VM/370 performance option called virtual=real, or if an OS ISAM program is executing in a virtual machine with the ISAM option specified in the user's VM/370 directory. Each operating system under VM/370 control requires that the virtual machine have an I/O configuration compatible with the one for which the operating system was originally generated.

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

- A terminal-oriented conversational system.
- A back-release system running application programs not upgraded to the current level of the production system.
- A current-release system with new Program Temporary Fixes (PTF's) applied.
- A new release or option being generated and tested.

The interactive execution of conventional operating systems in a virtual machine environment provides several convenient operating facilities. Some of these are:

System Programming Facilities:

- Testing of new or modified Supervisor Call (SVC) routines on a virtual machine.
- Applying and testing PTF's on a virtual machine.
- Generating and testing new releases of an operating system in a virtual machine.
- Hands-on console debugging as though on a dedicated real machine, via a remote terminal device.

Application Programming Facilities:

- Debugging while under operating system control, from a terminal.
- Designing of application programs without real storage constraints.
- Defining minidisks and other virtual devices to design and test a slightly different or larger machine configuration before the actual hardware is installed.

Operations Facilities:

- Training of operators in their own virtual machine.
- Defining a virtual machine and its devices as backup to some other real machine.
- Permitting divergent installation requirements to be run concurrently on a single real machine.

OPERATION OF A BATCH VIRTUAL MACHINE

The user of a batch virtual machine first logs into CP from his terminal via the CP LOGIN command. The appropriate configuration must either be in his virtual machine directory description, or dynamically altered to the proper configuration after login. The batch system residence volume must have been defined as a virtual device in the VM/370 user directory, or the real device on which it is currently mounted must now be logically associated with the user's batch virtual machine via the CP ATTACH command. The user then issues the IPL command, giving as the ipl device the virtual device address of the system residence volume.

The major differences between operating on the real machine and operating under CP are:

1. The CP commands are made available to the operator whenever the Attention (ATTN) key (or equivalent) is pressed twice.
2. The REQUEST key of a real operator's console is simulated by pressing the Attention key (or equivalent) twice.

To reply to messages from a virtual machine operating system:

1. Press the Attention (ATTN) key once to simulate the console Request key.
2. Enter the information requested.
3. Press the carriage return (RETURN) key on the 2741 Communication Terminal (or equivalent on other supported terminals).

The CANCEL key of the system console keyboard is simulated by the ATTN key on the 2741. To enable the CPU operator to make input line and character corrections to operating system responses, the VM/370 logical editing characters may be used.

When a virtual machine operating system requires a specified device, as in a MOUNT request, the user should do one of the following:

1. If the device is already defined as part of the virtual machine configuration, press the ATTN key twice to enter CP mode. Then enter the CP command READY ccu (with ccu replaced by the virtual I/O device address), followed by the CP BEGIN command. This is the virtual machine equivalent of physically making a device ready, if it is not already in a ready state.
2. If the device is not a standard device already defined as part of the virtual machine configuration, press the ATTN key twice to enter CP mode, and add the device to the virtual machine configuration. The CP LINK command can be used if the device is defined in the CP directory. If it is not, send a message to the system operator to attach it to the virtual machine. When the real device has been made ready, enter the BEGIN command.

Note: If the virtual device is an attached real device at the time a mount is requested, the real device can be taken out of and put back into ready status to cause an interrupt.

Tape drives, being nonsharable devices, are usually not defined in the user directory as part of a virtual machine configuration. Since this is the normal case, when a tape drive is required for a job, a system operator with privilege class B must attach an available real device to the virtual machine for the virtual device address specified by the user. When the job involving that tape drive is complete, the user can issue a DETACH command, naming his virtual device address. In this way, real tape drives are allocated to a virtual machine only for the duration of the jobs that need them.

The following example illustrates the loading of an OS/MFT system on a user's virtual machine, and the subsequent mounting of a virtual device at address 232. (For clarity, information entered by the user is shown in lower case, and messages typed out from CP and OS are shown in uppercase).

```
login os/mft
ENTER PASSWORD:

LOGIN AT 13:24:30 EST THURSDAY 12/07/72
ipl 230                          ipl the OS SYSRES
                                 device.
IEE007A READY                    OS ready message.
set date = 72.355,Q = (231)

start rdr, 00c
start wtr, 00e
start
IEF233A M 232, VOLABC,           OS requests operator to
  JOB,STEP                       mount a pack labeled
                                 "VOLABC" on 232.
!!CP                             The user presses the at-
                                 tention key twice to get
                                 out of OS mode and in-
                                 to CP mode
link to usera 330 as 232 w pass
                                 usera requests a disk
                                 with a virtual address of
                                 330 to be linked to his
                                 own virtual machine.
                                 He issues the CP link
                                 command to allow him-
                                 self to access that other
                                 user's disk at virtual ad-
                                 dress 232 in his virtual
                                 machine.
ready 232                        The user then keys in
                                 READY to simulate a
                                 ready interrupt to OS
                                 and then transfers con-
                                 trol back to the OS vir-
                                 tual machine.
begin
```

The virtual machine that is to run OS must have a configuration compatible with that for which the operating system's nucleus was generated. For example, the VM/370 user directory might contain the following information:

```
USER BATCH PASSWORD 512K
   ACCOUNT NUMBER BIN10
      OPTION REALTIMER  ISAM
      CONSOLE 01F 3215
      SPOOL C 2540 READER A
      SPOOL D 2540 PUNCH A
      SPOOL E 1403 A
      DEDICATE 230 BATCH1
      DEDICATE 231 BATCH2
```

Reading Cards in a Virtual Machine

When an operating system such as DOS, DOS/VS, or OS/PCP, MFT, MVT, VS1, or VS2 is running batch production jobs in a virtual machine, it may be occasionally necessary to dedicate a real card reader to that virtual machine. In this case, jobs would be entered through the card reader in exactly the same way they are entered on a standalone system, and no double spooling will occur.

If, however, there is no extra reader available to dedicate in such a manner, or if the virtual machine is being run from a terminal not located near a real card reader, other methods of handling card input are necessary.

Card images can be placed into the (spooled) virtual card reader of a user's virtual machine in three ways:

1. A card deck can be placed in the real card reader and read by VM/370. The deck must be preceded by a special VM/370 identification (ID) card specifying the virtual machine user's userid. VM/370 reads in the cards and transfers the card images to the specified virtual machine's virtual card reader as a spool file. When the virtual machine issues a read to the card reader, VM/370 presents the cards, one at a time, to the virtual machine.
2. A card deck can be placed in the card reader of a remote 2780 under control of a special VM/370 remote spooling service program called CP2780. The same VM/370 ID control card as described previously must precede the deck. The CP2780 program transfers the cards to VM/370, which transfers the card images to a specific user's input spool reader file.
3. A virtual machine's printer can transfer printer listing images and the punch can transfer punch images to its own (or to some other user's) virtual card reader. CP can handle up to 150-character records in virtual card readers. The CP SPOOL command provides the means of transferring these unit-record files. For example:

   ```
   spool punch to usera
   ```

   causes all cards punched to be transferred (after the punch file is closed) as an input file for usera's spooled card reader.

The user at the remote terminal would normally find it helpful to set the following spooling options in effect for his virtual card reader.

SPOOL 00C CONT NOEOF

CONT specifies that reading be "continuous" (that is, not to indicate an EOF condition to the virtual machine after each input file, but to continue reading all the card images until the reader files spooled to the virtual machine are exhausted). If this option is not in effect, a unit exception is reflected to the user's virtual machine at the end of each spooling file. This option eliminates the need to repeatedly enter the CP READY and CLOSE commands between the reading of each spooled file.

NOEOF indicates that the virtual reader's end-of-file button is assumed not to have been pushed. When all reader spool files are exhausted, an intervention-required status pending is reflected to the virtual machine. When additional reader files are transferred to the virtual card reader, a device end interrupt is reflected to the virtual machine, and card reading normally resumes automatically (because a device end interrupt "wakes up" the virtual machine).

The operator of a virtual machine operating system such as OS or DOS may choose to prepare his own job streams at his terminal using CMS editor facilities, and then IPL another operating system to process the job streams. This procedure is discussed under "Using Multiple Consecutive Operating Systems."

Printing and Punching in a Virtual Machine

VM/370 provides unit record spooling facilities. When operating systems that also provide unit record spooling facilities are running in a virtual machine, double spooling of printed and punched output may occur. If a significant amount of printing or punching is to be done, one of the spooling facilities should be eliminated.

1. To eliminate CP spooling, a real printer or punch can be temporarily attached to a virtual machine, or dedicated for a virtual machine in the VM/370 user directory via the DEDICATE record. In either case, CP will perform no spooling for that virtual device. Interrupt processing can be initiated by the virtual machine whenever it receives control of the CPU from CP.
2. CP spooling has priority over any virtual machine execution. The real printer and punch will be better utilized by CP than from a virtual machine. Use of CP spooling provides more efficient shared use of a single reader, printer, or punch among many virtual machines.
3. OS spooling can be eliminated by using OS JCL in the form UNIT=00E or UNIT=00D instead of SYSOUT=A or SYSOUT=B, where 00E and 00D are samples of the virtual device addresses.
4. DOS POWER spooling can be eliminated by using the Dedicated Printer Unit option.
5. When printer or punch spooling is performed by CP, the CP CLOSE command specifying the virtual printer

or punch address should be issued periodically. This will release previously finished and stacked output files to the CP spooling program and result in better utilization of the DASD spooling areas and of the unit record devices.

CMS has no native spooling facilities, and will not cause double spooling when running in a virtual machine. It does not support an attached or dedicated printer, punch, or reader, and must use the facilities of CP. CMS automatically closes the virtual printer or punch after the completion of each command that uses them.

MINIDISKS

General Information

When multiple virtual machine operating systems are being run concurrently, the external storage requirements would be highly uneconomical if direct access storage space for each of these systems had to be assigned on a one-to-one basis (that is, one real device for each virtual device).

If the system being run does not require the full capacity of a real device, a user can be assigned one or more minidisks. A minidisk is a logical subdivision of a physical disk pack with its own virtual device address, virtual cylinders (starting with 0, 1, 2, etc.) and a VTOC (or disk label identifier). Each user minidisk is preallocated the number of contiguous full cylinders he is apt to require, and that space is considered to be a complete virtual disk device.

Minidisks are controlled and managed by CP. If a system is to be run on both a virtual and a real machine, it is recommended that minidisks not be used for that system unless the minidisk starts at *real* cylinder zero.

Figure 8 illustrates the use and definition of minidisks. On the disk marked OSDOS1 are several minidisks, some formatted to OS requirements and others to DOS requirements. The directory entries for userid JFM (an OS user) describe 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 entries for userid JCE (a DOS user) describe the virtual device 130 as a 50-cylinder disk area on a real volume OSDOS1.

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

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) DSCB. Thus, for OS to support minidisks, a VTOC must be written whose space accounting DSCB is equal to the desired size of the minidisk. The remainder of the disk space on the real disk appears to OS to be permanently dedicated, and not assignable by the OS space accounting routines. When the desired minidisk size for OS or DOS is less than a full pack, the VM/370 service program MINIDASD should be used to perform this function.

DOS space allocation is specified in the XTENT job control card. It is the virtual machine user's responsibility to see that the XTENT cards refer to valid minidisk cylinders (for example, if a ten cylinder minidisk is being used, XTENT cards must refer to cylinders 0 through 9 only). A minidisk always starts at virtual cylinder zero.

VM/370 controls the boundaries of minidisks. Should an attempt be made to refer to a DASD address outside these boundaries, CP presents a command reject (seek check) I/O error to the virtual machine.

Track Characteristics

A minidisk is created by a suitable VM/370 user directory entry. Like real disks, minidisks must be formatted for use by the appropriate VM/370 service program. A minidisk may be initialized for subsequent data recording by executing one of the following service programs in a virtual machine:

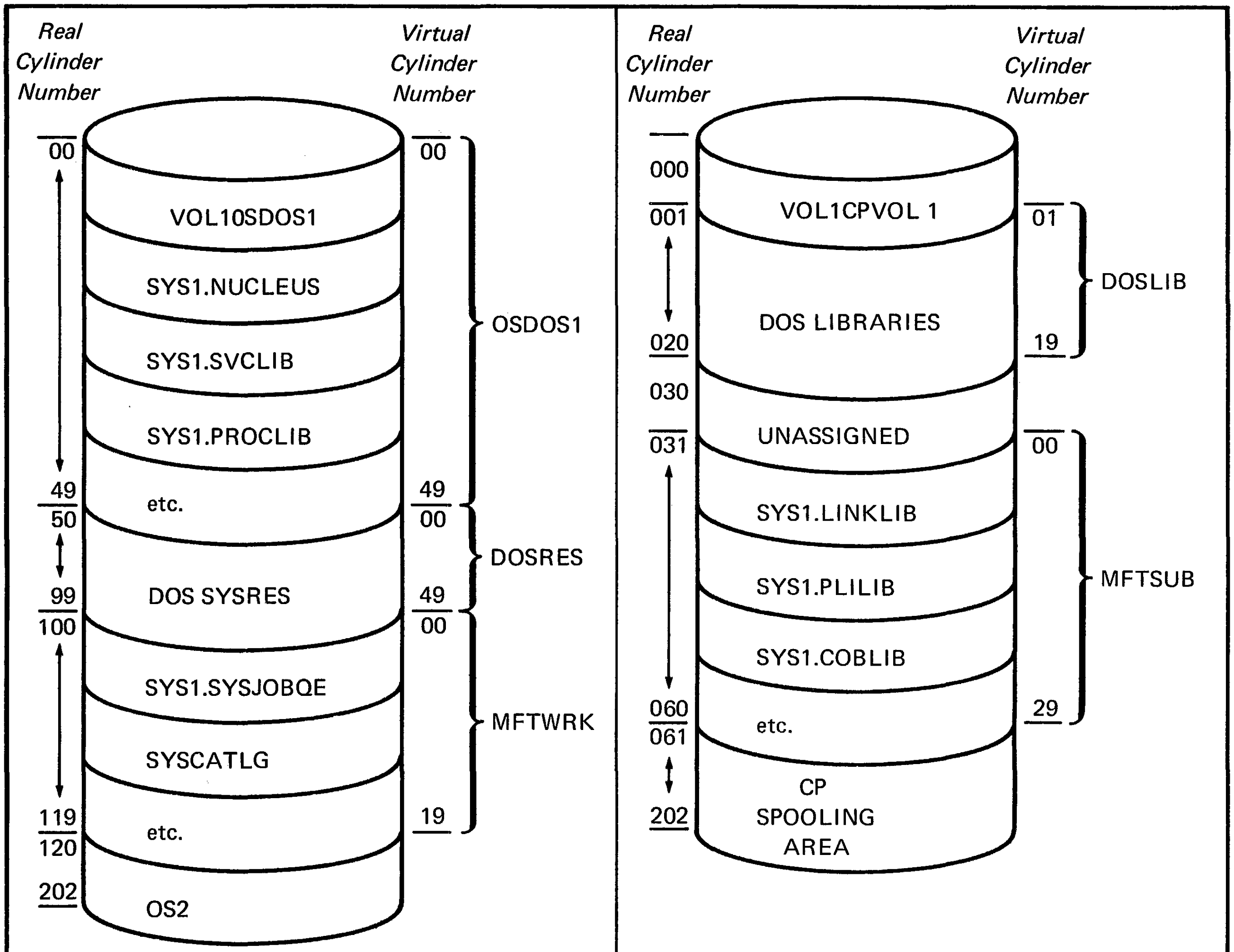
1. For CMS disks, the CMS FORMAT command formats the specified tracks into 800-byte blocks or physical records.
2. For CP disks, the CP FORMAT command is used to format specified tracks into 4096-byte blocks.
3. For OS and DOS disks, the CP MINIDASD service program writes read/only track descriptor records for each track, and clears the remaining portion of each track to binary zeroes. It also writes a Format 5 DSCB whose contents are the minidisk size.

Note: The standard DOS and OS initialize-disk utility programs flag defective tracks differently. OS can recognize defective tracks flagged by the DOS utility program, but DOS misinterprets OS-flagged defective tracks as an end of file (EOF) indicator of a DOS file. The MINIDASD program flags tracks the DOS way, so both DOS and OS can recognize defective tracks.

Alternate Tracks

Alternate tracks are automatically handled on the 3330 by the 3830 Control Unit. Minidisks 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.

On 2314 and 2319 devices, CP and CMS do not recognize or support traditional alternate track techniques for



VM/370 User Directory Entry for user JFM (An OS user)

```
USER        JFM      123       512K
   ACCOUNT     985
       CONSOLE  009  3215
       MDISK    230  2314  000 050  OSDOS1 W
       MDISK    231  2314  100 020  OSDOS1 W
       MDISK    232  2314  031 030  CPVOL 1 W
       SPOOL    00C  2540  READER A
       SPOOL    00D  2540  PUNCH A
       SPOOL    00E  1403  A
```

VM/370 User Directory Entry for user JOB (A DOS user)

```
USER  JCE  PASSWORD
    ACCOUNT  NUMBER  BIN14
       CONSOLE     01F    3215
       SPOOL       C      2540 READER
       SPOOL       D      2540 PUNCH
       SPOOL       E      1403
       MDISK       130    2314 050 050  OSDOS1 W
       MDISK       131    2314 001 020  CPVOL1 W
```

Note: VM/370 allows cylinders normally reserved for alternate track assignment (cylinders 200 to 202 on 2319 and 2314 disks, and cylinders 400 to 403 on 3330 disks) to be optionally used for normal data storage if included within the limits of a minidisk.

Figure 8. Use and Definition of Minidisks

their own use. DOS and OS minidisks, however, do recognize and support alternate tracks. The MINIDASD service program, if requested, can automatically assign the last cylinder in any minidisk as an alternate track cylinder. It is possible to assign all 203 cylinders to various user or system disk areas.

If a track assigned to a user minidisk area subsequently becomes defective, the following alternatives exist:

1. Run the CP FORMAT service program and flag the whole cylinder containing the defective track as permanently assigned (PERM). This will prevent CP from ever allocating that cylinder for CP paging, spooling, or temporary files. Do not include this

cylinder in the user directory disk allocation for any virtual machine's minidisk.

2. If the minidisk is used by either DOS or OS, reformat the minidisk (including the defective track) with the CP MINIDASD service program. An alternate track will be assigned at the end of the minidisk.
3. Set up the entire pack containing the defective track as an OS or DOS pack and format it with IEHDASDR for OS disks, or with the DOS Initialize Disk utility program (INPP) for DOS disks. Alternate tracks will be assigned in the traditional 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 represents a six character volume label.

Additionally, all user minidisks should have a label at virtual cylinder 0, track 0, record 3. Labels created by MINIDASD, IEHDASDR or INPP will be in the form:

VOL1xxxxxx

where xxxxxx represents a six character volume label.

A physical volume which holds only user minidisks may have the first of those minidisks starting at real cylinder 0. CP will be able to recognize the physical volume if the first minidisk has a label of the form described previously.

In Figure 8, the volume indicated as OSDOS1 has its real cylinder 0 allocated to a minidisk which is formatted for use by OS. The volume serial number of that minidisk must be OSDOS1, the label that is associated with the real volume. Since the minidisk label identifies the physical volume, changing it affects the directory entries of all users who have minidisks on that volume.

Since a user with read/write disk access is free to rewrite the label of his minidisk, real cylinder 0 should not normally be assigned as a user data area.

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

CHANNEL SWITCHING

CP does not itself include any channel switching facilities. However, if the real configuration includes a 2816 tape switching unit or a Two 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:

1. SYSGEN the virtual machine operating system for the appropriate hardware (in this case a 2816 tape switching unit on channels 5 and 6).
2. SYSGEN CP as though 580 and 680 are different devices (with different control units and channels).
3. 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 record in the CP user directory). Minidisks cannot be used.

RESERVE/RELEASE

VM/370 does not include any reserve/release support for virtual machines sharing DASD volumes. If the real configuration includes a Two Channel Switch feature, reserve/release can be made to operate under control of the virtual machine operating system. In this way, a virtual machine can share a DASD volume (either virtual or real).

For example, if 230 and 330 are alternate device addresses for a particular DASD facility to be shared by usera and userb (two virtual machines running on the same real computing system), then:

1. SYSGEN the virtual machine operating system for usera to support the appropriate hardware (device at 230, Two Channel Switch) and software (RESERVE/RELEASE).
2. SYSGEN the virtual machine operating system for userb to support the appropriate hardware (device at 330, Two Channel Switch) and software (RESERVE/RELEASE).
3. SYSGEN CP as though 230 and 330 were different devices (with different control units and channels).
4. Issue the CP ATTACH command, attaching device 230 to usera and device 330 to userb.

If the system generated for userb was to run in a real machine rather than a virtual machine then:

- SYSGEN the CP system with device 230 but not 330.

- Issue the CP ATTACH command, attaching device 230 to usera.

In both cases:

- The device addresses generated for systems to run in a virtual machine do not need to be the same as on the real machine.
- The devices used by virtual machines must be dedicated (attached or defined with a DEDICATE record in the CP user directory). Minidisks cannot be shared in this manner.

DISCONNECTING THE REMOTE TERMINAL

Once a virtual machine operating system has been loaded and jobs started, a terminal user may want to use his terminal for some other purpose while the batch jobs are running. The DISCONN command allows the virtual machine user to disconnect his terminal from the VM/370 system, but allows the virtual machine to continue operation.

The disconnect frees the terminal for other uses, but is useful only for operating systems that can run in an unattended mode. While in disconnect status, a "read" to the virtual machine operator's console will cause the virtual machine to be logged out; all "writes" or output messages to the virtual console are ignored.

The disconnect status remains in effect until either:

1. The user reconnects via LOGIN command before termination of the session, or
2. Forced log off of the disconnected virtual machine is caused by an attempted read to the virtual machine's system console.

A HOLD option is provided with the DISCONN command to prevent disabling the communication line in a switched line configuration.

The user reconnects to a disconnected virtual machine via the normal LOGIN procedure. His running virtual machine is placed in the CP command mode. To resume execution of the virtual machine operating system, the BEGIN command must be issued.

Example (a) shows the disconnect procedure and example (b) shows the subsequent reconnect.

Examples:

```
(a)     login usera
        ENTER PASSWORD:
        (printing of the password is inhibited)
        LOGIN AT 05:30:30 EST THURSDAY 12/07/72
        ipl 190
        CMS VERSION 1 LEVEL 0
        start
        !CP
        disconn
        DISCONNECT AT 05:35:30 EST THURSDAY
           12/07/72
(b)     login usera
        ENTER PASSWORD:
        (printing of the password is inhibited)
        RECONNECT AT 07:42:45 EST THURSDAY
           12/07/72
        begin
```

THE MULTIPLE-ACCESS VIRTUAL MACHINE

Multiple-access programs are those which execute in a virtual machine and directly control terminals. These terminals need not be of the type supported by CP as virtual operator consoles, but may be any type which can be supported by the virtual machine.

A subset of the lines of a real transmission control unit (TCU) can be defined as a virtual TCU for a virtual machine as shown in Figure 9.

In this example, two line appearances on the real 3705 have been defined as virtual transmission control units for two virtual machines named VM1 and VM2. The remaining line appearances may support virtual operator consoles.

The virtual machine operating system may be one like APL\DOS-360 (CP option) that supports a number of remote terminals, as shown in Figure 10.

When the terminals supported by the virtual operating system are of the same type as those supported by VM/370 as virtual operator consoles, the assignment of a real line appearance to a virtual TCU can be made dynamically.

To do this, virtual lines are defined in the virtual machine's user directory description via the SPECIAL record, or are added to the logged-in virtual machine via the CP DEFINE command.

Figure 10 illustrates a CP directory description for a multiple-access virtual machine to run APL\DOS-360 (CP option).

To connect to a multiple-access virtual machine, the CP DIAL command is provided. To connect to the virtual machine in the Figure 11, a user would enter the command as follows:

```
DIAL APLSYS
```

The CP system will match the terminal type with an equivalent virtual line that is available and enabled (in this example, 080, 081, 082, or 083). Once a connection is made, the terminal is under exclusive control of the virtual machine it is dialed into (in this example, the APL\DOS-360 virtual machine), and neither the terminal

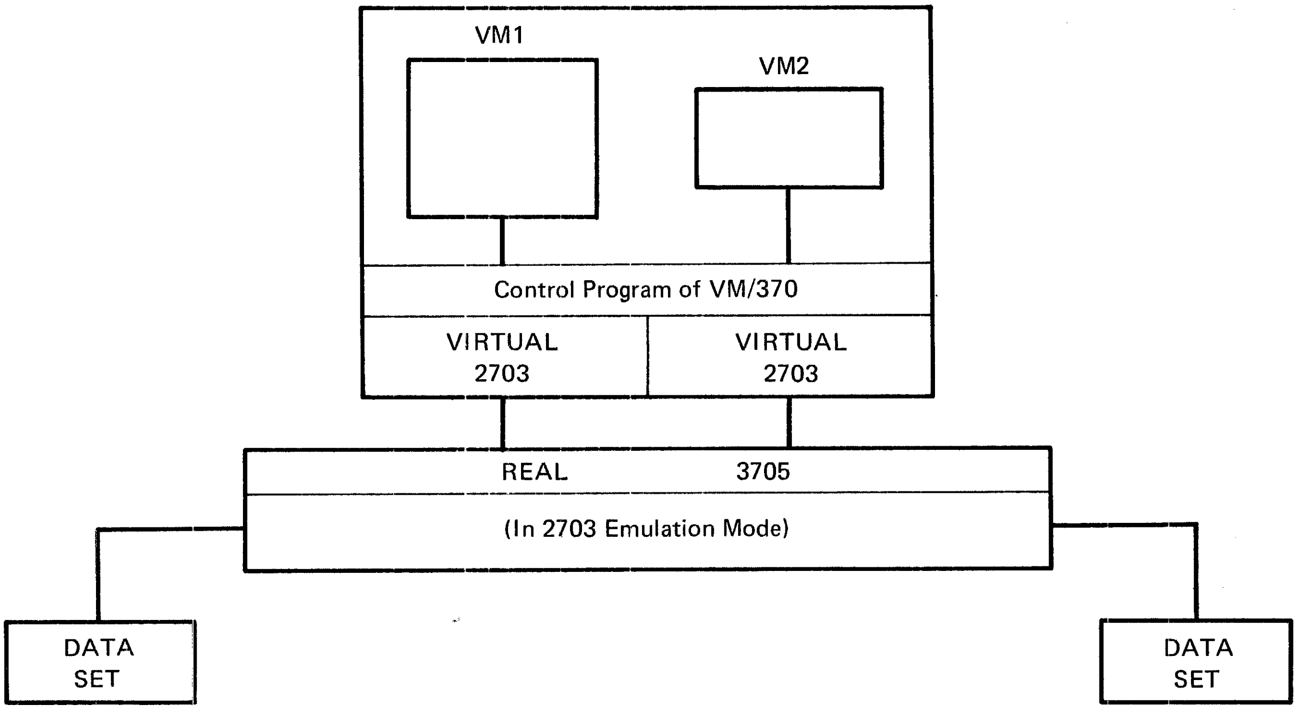


Figure 9. Virtual Devices: Teleprocessing Units

nor the user can issue any CP or CMS commands. It will remain connected until the user logs out of APL using standard APL logout procedures or until the APL virtual machine is logged off. The user is then free to log into CP or to DIAL another multiple-access system.

Dial-up terminals supported by the multiple-access system may be of a different type than those supported by VM/370 as virtual operator consoles. In that case, the CP DIAL command cannot be used. The user must dial the multiple-access system's telephone numbers directly.

Additionally, a communications system can be tested using multiple virtual machines in place of multiple real machines, as shown in Figure 12. The virtual 2701s

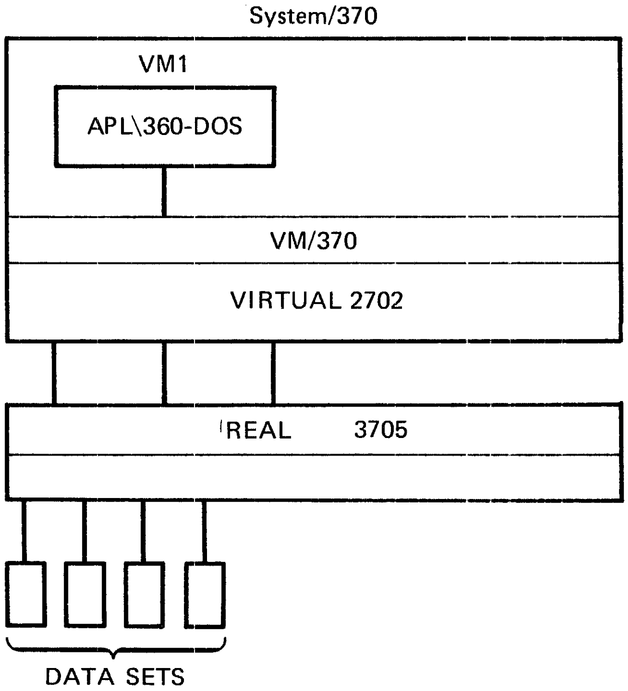


Figure 10. A Virtual Multiple-Access System

```
USER   APLSYS 051243  256K
       CONSOLE 01F  3215
            SPOOL     00C   2540    READER
            SPOOL     00D   2540    PUNCH B
            SPOOL     00E   1403    A
            DEDICATE 190  SYSRES
            DEDICATE 191  SYSWRK
            SPECIAL   080   2702    IBM
            SPECIAL   081   2702    IBM
            SPECIAL   082   2702    IBM
            SPECIAL   083   2702    IBM
```

Figure 11. VM/370 User Directory Description for an APL\DOS-360 Virtual Machine.

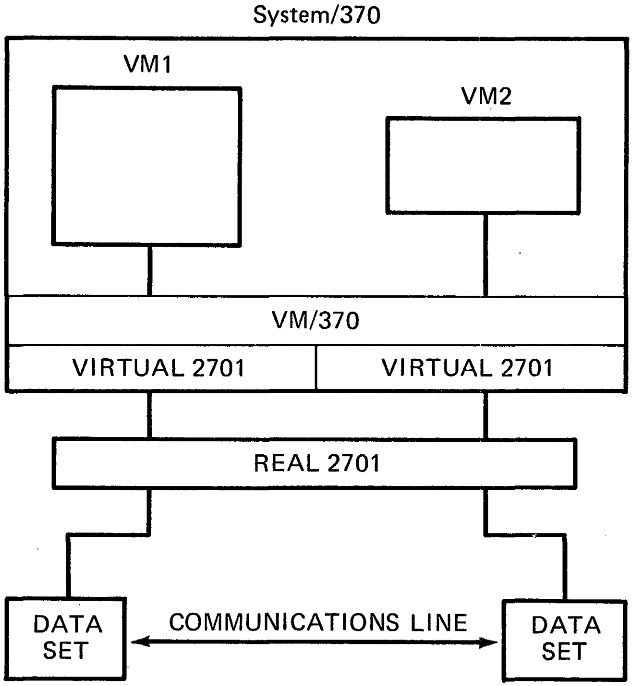


Figure 12. A Communications System Test

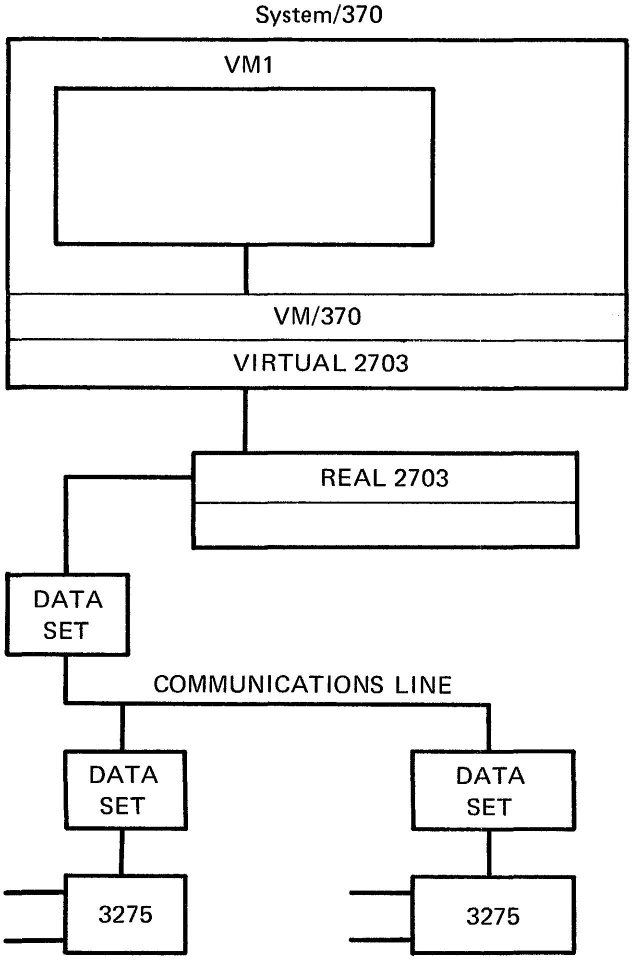


Figure 13. A Virtual TCU Running Remote 3270s

could each be defined as a one-line 2701, while on the real machine there exists a single two-line 2701.

Figure 12 assumes that the real transmission control unit is equipped with the appropriate data sets and line capability.

Figure 13 illustrates a virtual transmission control unit running remote 3270s.

When the terminals supported by the multiple-access system are not of the type supported by VM/370 as virtual operator consoles, the real line appearances must be:

1. Defined in the CP user directory description for the virtual machine via the DEDICATE record, as

   DEDICATE ccu

   where ccu is the real address of the appropriate line appearance on the real transmission control unit, or
2. Attached to the virtual machine by an operator with privilege class B as

   ATTACH ccu TO VM1 AS vccu

   where ccu is the real address of the appropriate line appearance on the real transmission control unit, and vccu is the address of the line appearance as generated in the virtual machine operating system.

THE ASP VIRTUAL MACHINE

The OS Attached Support Processor (ASP) user will find virtual machines useful in two ways.

First, a virtual ASP system can be run by two virtual machines (VM1 and VM2) together with a virtual channel-to-channel adapter. This is illustrated in Figure 14.

The virtual ASP system (VM1 and VM2) shown in Figure 14 may be used for bringing up a new ASP release, or for ASP system testing in a virtual environment concurrent with normal production.

This eliminates the need to dedicate the real ASP computer for new system testing.

The CP user directory for each of the virtual ASP processors must contain a record defining a virtual channel-to-channel adapter in the form:

```
SPECIAL 280 CTCA
```

where 280 is the address of the channel-to-channel (CTC) adapter as generated in the OS system.

When both virtual ASP machines have logged into VM/370, the CP COUPLE command must be issued by one of the virtual machine operators:

```
couple 280 to vm2 380
```

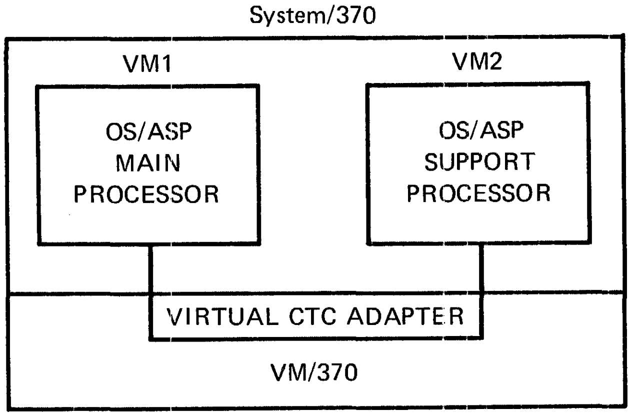


Figure 14. A Virtual Machine ASP System

where 280 is the address of VM1's virtual channel-to-channel adapter, VM2 is the userid of the second virtual machine, and 380 is the address of VM2's virtual channel-to-channel adapter. Once coupled, the operator of each virtual machine can then IPL his OS system and start running ASP.

Second, an additional virtual machine (VM3), with a real channel-to-channel adapter to a real System/360 or System/370, can be running as either the main or support processor in a production ASP system. This is illustrated in Figure 15.

The real channel-to-channel adapter must be defined in the VM/370 SYSGEN procedure via the RDEVICE macro with a device type of CTCA (DEVTYPE=CTCA). The virtual machine (VM3) must have this device assigned to it before the IPL. This can be done via the DEDICATE record in the virtual machines user directory description as follows:

```
DEDICATE 280 380
```

where 280 is the address of the channel-to-channel (CTC) adapter as generated in the OS system, and 380 is the address of the real channel-to-channel adapter as specified in the VM/370 system generation procedure.

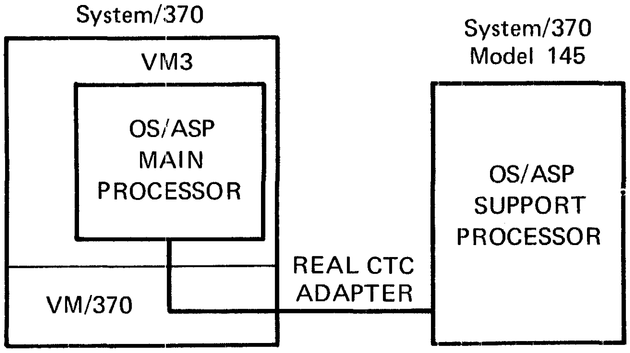


Figure 15. A Virtual Machine in an ASP Configuration.

If there is no DEDICATE record for the channel-to-channel adapter in the virtual machines description, a resource operator with privilege class B must attach the real channel-to-channel adapter to the virtual machine.

USING MULTIPLE CONSECUTIVE OPERATING SYSTEMS

The user of a virtual machine may require the facilities of more than one operating system during a single terminal session.

When an operating system such as OS, DOS, OS/VS1, or DOS/VS is being run from a remote terminal, the operator of the virtual machine may utilize CMS editor facilities to create and modify job streams.

The problem programmer who normally uses CMS to interactively create, modify, and test his programs may require facilities not supported or available in CMS for compilation (for example, programs using DOS macros) or for execution (for example, any DOS object program, or any OS object program that utilizes certain OS supervisory functions and access methods).

Note: A DOS macro library may be created for use within CMS in the same manner as an OS macro library may be. See the discussion of "OS Macro Simulation Under CMS".

The following technique for using multiple consecutive operating systems is termed a "flip-flop" approach. Job control cards, compiler or assembler source programs, and test data streams are created and modified at the terminal under control of the CMS editor. Actual execution of the job stream is done when control is "flipped" to an appropriate operating system with the necessary facilities.

In this way, problem programmers can use the terminal-oriented facilities of CMS to create and maintain source programs and JCL. When ready to compile or test, they can "flip" to a DOS or OS virtual machine, and after execution is finished, transfer the printer and punch output back to CMS for selective scanning and typing at the remote terminal.

This approach assumes the terminal user has created source program files and data files under CMS. To execute under another operating system (in this example, OS) he must also create JCL records which specify the compilation, link edit, or execution, as appropriate. These would be created in CMS and named with a distinctive filename and filetype (for example, PLICOMP JCL). Job control records, source program files, and data files can then be merged together in the virtual card reader to form a single OS job stream. A special combination of CP and CMS commands shown in Figure 16 accomplishes this job stream creation and transfer.

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

Figure 16 is a specific example of this approach. The virtual machine is in CMS mode at the start of the example. The command "SPOOL 00c cont eof" specifies that reading be continuous until all files spooled to the virtual machine are exhausted, and that the virtual end-of-file button on the reader has been pushed. "Noheader" specifies that no special control cards are to be inserted at the head of each punched file. Virtual device 230 is an OS system volume. All standard CMS responses (for example, R;T=0.04/0.12 09:36:08) have been omitted from the above example; however, the OS READY message is included to more fully illustrate the IPL sequence. Also, the Attention (ATTN) key (or equivalent) must be pressed once before entering each OS command, although this is not indicated in the example.

To effect this transfer, the virtual machine must have access to both operating systems being used. This can be done dynamically, before loading the new system, or access to both systems can be provided in the virtual machine directory description.

Figure 17 illustrates a virtual machine configuration and the corresponding user directory description records.

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

Note: The following are issued once under OS control.

```
IEE007A READY
set date=72.355,Q=(231)
start rdr,00c
start wtr,00e
start
```

Figure 16. OS Job Stream Transfer

Virtual device addresses 190 and 191 contain the CMS system and user disk area. Virtual device addresses 230 and 231 contain the OS system and user disk area. The two systems use a common card reader, card punch, printer, and console.

Operation of the flip-flop is simplified if:

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

If these two conditions do not exist, the virtual machine configuration must be dynamically modified before each IPL of a new system.

If the systems require online typewriter keyboards at different addresses, the CP DEFINE command can be used to change the address of the virtual system console. For example, if the OS system specified in Figure 17 required an online typewriter keyboard at address 01F, the command

```
cp define 009 as 01F
```

should be issued before loading 230. When the OS job stream is completed, the command

```
cp define 01F as 009
```

should be issued before loading CMS. Virtual storage size can be changed and virtual card readers, printers and punches can be added with this procedure.

If the systems expect different device types at the same address (for example, in Figure 17 CMS expects a 2314 at address 191, but the OS system might be generated to support a 3330 at that address), the common address must be assigned to the appropriate device each time a new system is to be loaded. In this case, running CMS with a disk at address 191, the following command should be issued before loading OS:

```
cp detach 191
```

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

Figure 17. Virtual Machine Directory Entry to Support a Flip-flop.

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

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

PERFORMANCE GUIDELINES

General Information

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

- The System/370 model used.
- The total number of virtual machines executing.
- The type of work being done by each virtual machine.
- The speed, capacity, and number of the paging devices.
- The amount of real storage available.
- The degree of channel and control unit contention, as well as arm contention, affecting the paging device.
- The type and number of VM/370 performance options in use by one or more virtual machines.

Performance of any virtual machine may be improved up to some limit by the choice of hardware, operating system and VM/370 options. The topics discussed in this section address:

1. The performance options available in VM/370 to improve the performance of a particular virtual machine.
2. The SYSGEN options and operational characteristics of operating systems running in virtual machines that will affect their execution in the virtual machine environment.

The performance of a specific virtual machine may never equal that of the same operating system running standalone on the same System/370, but the total throughput obtained in the virtual machine environment may equal or better that obtained on a real machine. This is mainly due to the fact that CP's multiprogramming capabilities allow far more programs to run concurrently than, for example, DOS or DOS/VS. Each job or task may take longer, but more tasks can be accomplished in a given period.

When executing in a virtual machine, any function that cannot be performed wholly by the hardware will cause some degree of degradation in the virtual machine's performance. As the control program for the real machine, CP initially processes all real interrupts. A virtual machine operating system's instructions are always executed in *problem* state. Any privileged instructions issued by the virtual machine will cause a real privileged instruction exception interruption. The amount of work to be done by CP to analyze and handle a virtual machine-initiated interrupt depends upon the type and complexity of the interrupt.

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

When planning for the virtual machine environment, consideration should be given to the number and type of privileged instructions to be executed by the virtual machines. Any effort that can be made towards reducing the number of privileged instructions issued by the virtual machine's operating system will reduce the amount of extra work CP must do to support the machine.

Virtual Machine I/O

To support I/O processing in a virtual machine, CP must translate all virtual machine channel command word (CCW) sequences to refer to real storage and real devices and, in the case of minidisks, real cylinders. When a virtual machine issues an SIO, CP must:

1. Intercept the virtual machine SIO interrupt.
2. Allocate real storage space to hold the real CCW list to be created.
3. Translate the virtual device addresses referred to in the virtual CCWs to real addresses.
4. Page into real storage and lock for the duration of the I/O operation all virtual storage pages required to support the I/O operation.
5. Generate a new CCW sequence building a Channel Indirect Data Address list if the real storage locations cross page boundaries.
6. Schedule the I/O request.
7. Present the SIO condition code to the virtual machine.
8. Intercept, retranslate, and present the channel end and device end interrupts to the appropriate virtual machine, where they must then be processed by the virtual machine operating system.

Handling SIOs for virtual machines by CP can be one of the most significant causes of reduced performance in virtual machines.

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

- Use of large blocking factors for user data sets to reduce the total number of SIOs (of up to 4096 bytes) needed.
- Use of pre-allocated data sets.
- Use of virtual machine operating system options (such as chained scheduling in OS) that reduce the number of SIO instructions.
- Substitution of a faster resource (virtual storage) for I/O operations, by building small temporary data sets in virtual storage rather than using an I/O device.

Frequently, there can be a performance gain when CP paging is substituted for virtual machine I/O operations. The performance of an operating system such as OS can be improved by specifying as resident as many frequently used OS functions (transient subroutines, ISAM indexes, etc.) as are possible. In this way, paging I/O will be substituted for virtual machine-initiated I/O. In this case, the only work to be done by CP is to place into real storage the page which contains the desired routine or data.

Two CP performance options are available to reduce the CP overhead associated with virtual machine I/O:

1. The virtual=real option removes the need for CP to perform storage reference translation and paging before each I/O operation for a specific virtual machine.
2. The dedicated channel option allows CP to bypass its device address translation and most channel scheduling and interrupt reflection routines for all devices on a channel.

Assignment and use of these options is discussed in "Preferred Virtual Machines."

Paging Considerations

References made by virtual machines to virtual storage addresses that are not currently in real storage cause a paging exception and attendant CP paging activity.

The addressing characteristics of programs executing in virtual storage have a significant effect on the number of page exceptions experienced by that virtual machine. Routines that have widely scattered storage references tend to increase the paging load of a particular virtual machine. When possible, modules of code that are dependent upon each other should be located in the same page. Reference tables, constants, and literals should also be located near the routines that use them. Exception or error routines that are infrequently used should not be placed within main routines, but located elsewhere.

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

Virtual machines that reduce their paging activity by controlling their usage of addressable space benefit that virtual machine, the VM/370 system, and all other virtual machines. The total paging load that must be handled by CP is reduced, and more time is available for productive virtual machine use.

CP provides two other performance options, locked pages and reserved page frames, to reduce the paging requirements of virtual machines. Generally, these facilities require some dedication of real main storage to the chosen virtual machine, and therefore, improves its performance at the expense of other virtual machines.

Locked Pages

The LOCK command, which is available to the system operator (with privilege class A), can be used to permanently fix or lock specific user pages of virtual storage into real storage. In doing so, all paging I/O for these page frames is eliminated.

Since this facility reduces total main storage resources (real page frames) that are available to support other virtual machines, only frequently used pages should be locked into real storage. Since page zero (the first 4096 bytes) of a virtual machine storage is referred to and changed frequently (for example, whenever a virtual machine interrupt occurs or when a CSW is stored), it should be the first page of a particular virtual machine that an installation considers locking. The virtual machine interrupt handler pages might also be considered good candidates for locking.

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

The normal CP paging mechanism selects "unreferred to" page frames in real storage for replacement by active pages. Unreferenced page frames are those whose contents have not been referred to during the last 50 milliseconds. Page frames belonging to inactive virtual machines will all eventually be selected and paged out if the main storage frames are needed to support active virtual machine pages.

When virtual machine activity is initiated on an infrequent or irregular basis, such as from a remote terminal in a teleprocessing inquiry system, some or all of its virtual storage may have been paged out before the time the virtual machine must begin processing. Some pages will then have to be paged in so that the virtual machine can respond to the teleprocessing request compared to running the same teleprocessing program on a real machine. This paging activity may cause an increase in the time required to respond to the request compared to running the teleprocessing program on a real machine. Further response time is variable, depending upon the number of paging operations that must occur.

Locking specific pages of the virtual machine's program into main storage may erase this problem, but it is not always easy or possible to identify which specific pages will always be required.

The Reserved Page Frames Option

A more flexible approach than locked pages is the reserved page frames option. This option provides a specified virtual machine with an essentially private set of real page frames, the number of frames being designated by the system operator. Pages will not be locked into these frames. They can be paged out, but only for other active pages of the same virtual machine. When a temporarily inactive virtual machine having this option is reactivated, these page frames are immediately available. If the program code or data required to satisfy the request was in real storage at the time the virtual machine became inactive, no paging activity is required for the virtual machine to respond.

This option is usually more efficient than locked pages in that the pages that remain in main storage are those pages with the greatest amount of activity at that moment, as determined automatically by the system. Although multiple virtual machines may use the LOCK option, only one virtual machine at a time may have the reserved page frames option active. Assignment of this option is discussed further in "Preferred Virtual Machines."

The reserved page frames option provides performance that is generally consistent from run to run with regard to paging activity. This can be especially valuable for production-oriented virtual machines with critical schedules, or those running teleprocessing applications where response times must be kept as short as possible.

The Virtual=Real Option

The VM/370 virtual=real option eliminates CP paging for the selected virtual machine. All pages of virtual machine storage, except page zero. are locked in the real storage locations they would use on a real computer. CP controls real page zero, but the remainder of the CP nucleus is relocated and placed beyond the V=R machine in real storage. This option is discussed in more detail in "Preferred Virtual Machines."

Since the entire address space required by the virtual machine is locked, these page frames are not available for use by other virtual machines except when the V=R machine is not logged on. This is a particularly expensive option for other virtual machine users, and in some cases for VM/370. (Paging activity on the system may increase substantially, since all other virtual machine storage requirements must be managed with fewer remaining real page frames.)

In some situations, the virtual=real option is desirable or mandatory:

1. To eliminate the double paging that might otherwise occur when running a virtual machine operating system (like DOS/VS or OS/VS) that performs paging of its own.
2. To allow programs that execute self-modifying channel programs or have a certain degree of hardware timing dependencies to run under VM/370.

CPU RESOURCES

CP allocates the CPU resources to virtual machines according to their operating characteristics, priority, and the system resources available. Virtual machines are dynamically categorized at the end of each time slice as interactive or noninteractive, depending on the frequency of operations to or from either the virtual system console or a terminal controlled by the virtual machine.

Virtual machines are dispatched from one of two queues, called Queue 1 and Queue 2. To be dispatched from either queue, a virtual machine must be considered executable (that is, not waiting for some activity or for some other system resource). Virtual machines are not considered dispatchable if the virtual machine:

1. Enters a virtual wait state after an I/O operation has begun.
2. Is waiting for a page frame of real storage.
3. Is waiting for an I/O operation to be translated by CP and started.
4. Is waiting for CP to simulate its privileged instructions.
5. Is waiting for a CP console function to be performed.

Queue 1 (Q1)

Virtual machines in Q1 are considered conversational or interactive users, and enter this queue when an interrupt from a terminal is reflected to the virtual machine. There are two lists of users in Q1, executable and non-executable.

The executable users are stacked in a first in, first out (FIFO) basis. When a non-executable user becomes executable, he is placed at the bottom of the executable list. If a virtual machine uses more than 50 milliseconds (ms) of CPU time without entering a virtual wait state, that user is placed at the bottom of the executable list.

Virtual machines are dropped from Q1 when they have accumulated 400 ms (or 0.4 seconds) of CPU usage, and are placed in an "eligible list". Virtual machines entering CP command mode are also dropped from Q1. When the virtual machine becomes executable again (returns to execution mode) it is placed at the bottom of the executable list in Q1.

Queue 2 (Q2)

Virtual machines in Q2 are considered non-interactive users. Users are selected to enter Q2 from a list of eligible virtual machines (the "eligible list"). The list of eligible virtual machines is sorted on a FIFO basis within user priority (normally defined in the USER record in the VM/370 user directory, but may be altered by the system operator).

A virtual machine is selected to enter Q2 only if its "working set" is not greater than the number of real page frames available for allocation at the time. The working set of a virtual machine is calculated and saved each time a user is dropped from Q2 and is based on the number of virtual pages referred to by the virtual machine during its stay in Q2, and the number of its virtual pages that are resident in main storage at the time it is dropped from the queue.

If the calculated working set of the highest priority virtual machine in the eligible list is greater than the number of page frames available for allocation, CP continues through the eligible list in user priority order.

There are two lists of users in Q2, executable and non-executable. Executable virtual machines are sorted by "dispatching priority". This priority is calculated each time a user is dropped from a QUEUE and is the ratio of CPU time used while in the queue to elapsed time in the queue. Infrequent CPU users are placed at the top of the list and are followed by more frequent CPU users. When a non-executable user becomes executable, he is placed in the executable list based on his dispatching priority.

When a virtual machine accumulates four seconds of CPU usage, he is dropped from Q2 and placed in the eligible list by user priority. When a user in Q2 enters CP command mode, he is removed from Q2. When he becomes executable (returns to virtual machine execution mode) he is placed in the eligible list based on user priority.

If a user's virtual machine is not in Q1 or Q2, it is because:

1. The virtual machine is on the "eligible list", waiting to be put on Q2, or
2. The virtual machine execution is suspended because the user is in CP mode executing CP commands.

To leave CP mode and return his virtual machine to the "eligible list" for Q2, the user can issue one of the CP commands that transfer control to the virtual machine operating system for execution (for example, BEGIN, IPL, EXTERNAL, and RESTART).

In CP, interactive users (Q1), if any, are considered for dispatching before non-interactive users (Q2). This means that CMS users entering commands which do not involve disk or tape I/O operations should get fast responses from the VM/370 system even with a large number of active users.

An installation may choose to override the CP scheduling and dispatching scheme and force allocation of the CPU resource to a specified user, regardless of its priority or operating characteristics. The favored execution facility allows an installation to:

1. Specify that one particular virtual machine is to receive up to a specified percentage of CPU time.
2. Specify that any number of virtual machines are to remain in the queues at all times. Assignment of the favored execution option is discussed in "Preferred Virtual Machines".

MULTIPROGRAMMING SYSTEMS UNDER VM/370

Virtual machines that are created by VM/370 are controlled in such a way as to provide concurrency of operation. When a multiprogramming system such as OS or DOS is run on a virtual machine, its resource algorithms interact with those of VM/370, as follows:

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

   It is often advantageous to run several virtual machines, each with one partition or region, rather than one virtual machine with multiple partitions. This may not be feasible for programs such as IMS, POWER, or DATA/360, which routinely require two or more partitions for operation, or for a multiple-partition production system which must also be run standalone on the same or on a different machine. When multiprogramming systems must be run in a virtual machine, use of the reserved page frames option or the virtual=real option may be advisable.

2. On a real machine, when a task is waiting for an I/O operation to complete, the lower priority tasks are given use of the CPU. Under VM/370, the I/O operations of a particular virtual machine are overlapped with the CPU execution of that and other virtual machines. Consequently lower priority tasks created by OS or DOS are given the CPU resource less frequently when executing in a virtual machine than when executing in a real machine.

Use of the favored execution option can insure that lower priority DOS or OS tasks will have a chance to execute. This option reduces the effect of a variable system load on the favored virtual machine.

CONFIGURATIONS

Devices Supported

The following devices will be supported by VM/370 except as otherwise noted below.

CPUs

IBM System/370 Model 135
IBM System/370 Model 145
IBM System/370 Model 155 II
IBM System/370 Model 158
IBM System/370 Model 165 II
IBM System/370 Model 168

All System/370 Models must have at least 245,760 bytes of real main storage.

Required Features and Facilities

- The System Timing facility (which includes the Clock Comparator and the CPU Timer) on the Model 135 and 145.
- The Floating-point feature
  - For the 135, feature #3900
  - For the 145, feature #3910
- 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.

Desirable Features

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

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

Direct Access Storage Devices

IBM 2314 Direct Access Storage Facility
IBM 2319 Disk Storage
IBM 3330 Disk Storage, Model 1
IBM 3333 Disk Storage, Model 1
IBM 2305 Fixed Head Storage, Model 1 (Models 165 II and 168 only)
IBM 2305 Fixed Head Storage, Model 2

Direct Access Control Units

IBM 3345 Integrated Storage Control Models 3, 4, and 5 on the Model 145 for 3330 Model 1 and 3333 Model 1
IBM 2835 Storage Control Model 1 for 2305 Model 1 (Models 165 II and 168 only)
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 Model 1
IBM 3830 Storage Control Model 2 for 3330 Model 1 and 3333 Model 1
IBM Integrated File Adapter (#4650) on System/370 Models 135 and 145 for 2319
IBM Integrated File Adapter on the System/370 Model 135 for 3330 Model 1 and 3333 Model 1
IBM Integrated Storage Control on the System/370 Model 158 for 3330 Model 1 and 3333 Model 1
IBM Integrated Storage Control on the System/370 Model 168 for 3330 Model 1 and 3333 Model 1

All of the above direct access devices are supported as VM/370 system residence, paging and spooling devices. All except the 2305 are supported by CMS.

Magnetic Tapes

IBM 2401, 2402, and 2403 Magnetic Tape Units
IBM 2415 Magnetic Tape Units, Models 1, 2, 3, 4, 5, 6
IBM 2420 Magnetic Tape Units, Models 5, 7
IBM 3420 Magnetic Tape Units, Models 3, 5, 7

Magnetic Tape Control Units

IBM 2804 Tape Control
IBM 2803 Tape Control
IBM 3803 Tape Control

Printers

IBM 1403 Printer Models 2, 3, 7, and N1 (with minimum of 132 print positions)
IBM 1443 Printer Model N1 (with 144 print positions)
IBM 3211 Printer

Readers/Punches

IBM 2501 Card Reader Models B1 and B2

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

Unit Record Control Units

IBM 2821 Control Unit
IBM 3811 Printer Control Unit

IBM Terminals and Other Devices

Terminals which are equivalent to those explicitly supported may also function satisfactorily. The customer is 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.
Note: 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 below.

1. The following system consoles are supported by VM/370 as real as well as virtual CPU operator's consoles.
   - IBM 3210 Console Printer-Keyboard Models 1 and 2.
   - IBM 3215 Console Printer-Keyboard Model 1
   - IBM 2150 Console with 1052 Printer-Keyboard Model 7
   - IBM System Console for System/370 Model 158 (in 3215 Emulator Mode) with the 3213 Printer Model 1 required
   - IBM 7412 Console (via RPQ AA2846) with 3215 Console Printer-Keyboard Model 1

   Note: The IBM 3066 System Console for the System/370 Models 165 II and 168 is not supported by VM/370 for its own use. It can, however, be attached to an OS virtual machine as an OS operator's console. VM/370 requires one of the following for use as the VM/370 system console:

   - A local terminal (via a transmission control unit)
   - A 1052 Model 7 (via the 2150 Console)
   - A 3215 Model 1 (via the 7412 RPQ Console)

   Only the latter two devices (the system consoles) can be used to generate the *initial* VM/370 system on a Model 168 or 165 II, therefore the initial VM/370 generation must be performed on a suitably equipped System/370.

2. IBM 2741 Communication Terminal on either switched or point-to-point lines with these features:
   - PTTC/EBCD (#9571, Part #1167963) or standard correspondence (#9808, Part #1167015) print elements
   - Transmit Interrupt (#7900) or Transmit Interrupt Control RPQ #E40681
   - Receive Interrupt (#4708)
   - Required with switched lines:
     - Data Set Attachment (#9114)
     - Dialup (#3255)
   - One of the following is required for private or leased point-to-point lines:
     - 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 (#4691-4696 for 4-wire shared leased line),

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

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

   - Print Inhibit (#5501)
   - Red Ribbon Control RPQ #868019
   - Typamatic Keys (#8341)
   - Pin Feed Platen (#9509)

3. IBM 1050 Data Communication System on either switched or point-to-point lines with these components:
   - 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)
   - Required with switched lines:
     - Data Set Attachment (#9114)
   - One of the following is required for private or leased point-to-point line:
     - 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 (#4635 for 4-wire limited distance line),

  or

- IBM Line Adapter (#4691-4694 for 4-wire shared leased line),

  or

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

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

- Automatic Ribbon Shift and Line Feed Select (#1295)
- EOB on Carrier Return RPQ #E28235

- ● 1052 Printer-Keyboard (Model 1 or 2) with the following feature:
  - PTTC/EBCD print element (#9571, Part #1167963)

The following 1050 components are also supported under VM/370. The error recovery used for these auxiliary components, however, requires intervention for retransmission of data. The Line Correction feature (#4795) is not supported by VM/370, even if installed on the terminal.

- 1053 Printer (Model 1)
- 1054 Paper Tape Reader (Model 1)
- 1055 Paper Tape Punch (Model 1)
- 1056 Card Reader (Model 3)
- 1058 Printing Card Punch (Model 2)

4. 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.

Transmission Control Units

1. IBM 2701 Data Adapter Unit

   Line Control for CPT-TWX (Model 33/35) Terminals

   - ● Telegraph Adapter Type II (#7885)

   For 2780 Data Transmission Terminals

   - ● Synchronous Data Adapter Type II (#7698)
     - EBCDIC code (#9060)
     - EBCDIC transparency (#8029)

   For 1050 and 2741 Terminals

   - ● IBM Terminal Adapter Type I, Model II (#4640)
     - Selective Speed (134.5 bps) (#9581)
     - 2741 Break Feature RPQ #M53193, and Break Command RPQ #858492

   As Needed:

   - ● Expanded Capability (#3815)–Required if more than 2 low speed adapters (either IBM Type I Model II, or Telegraph Type II), or more than 1 high speed adapter (Synchronous Data Adapter Type II), or 1 high speed and at least 1 low speed adapter are to be attached to the same 2701.
   - ● Expansion Feature (#3855–Required for each line adapter after the first.

2. IBM 2702 Transmission Control unit

   For 1050 and 2741 Terminals

   - ● Terminal Control Base for IBM Terminal Control (#9696)

     IBM Terminal Control Type I (#4615)
     - Selective Speed (134.5 BPS) (#9684)
     - 2741 Break (#8055) and Break Command IBM Terminal Control I RPQ #E46765 or Type I Terminal Interrupt (#8200)
     - Data Set Line Adapter (#3233) or IBM Line Adapter (#4635) (4-wire)

   Line Control for CPT-TWX (Model 33/35) Terminals

   - ● Terminal Control Base for Telegraph Terminal Control (#9697)

     Telegraph Terminal Control Type II (#7912)
     - Pluggable End Characters[1] (return key generates an interrupt) RPQ #E62920
     - Data Set Line Adapter (#3233)
   - ● Terminal Control Expansion (#7935)–required if both of the above terminal control bases are to be attached to the same 2702.

   As Needed:

   - ● 31 Line Expansion (#7955)

3. IBM 2703 Transmission Control unit

   For 1050 and 2741 Terminals

   - ● 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)
     - 2741 Break (#8055) and Break Command Terminal Control I RPQ #E53715 or Type I Terminal Interrupt (#8200)
     - Data Line Set (#3205) and Data Line Set

[1] Although not required, these features enhance the convenience and usability of these terminals.

Expander (#3206), and/or IBM Line Set 1B (#4687)

Line Control for CPT-TWX (Model 33/35) Terminals

- 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[1] (return key generates an interrupt) RPQ #E66707

For 2780 Transmission Terminals

- Synchronous Base (#7703, 7704, or 7706)

  Synchronous Terminal Control for EBCDIC (#7715)

  – Transparency (#9100)

  – Synchronous Line Set (#7710)

As Needed:

- Base Expansion (#1440)–required if more than one base type is to be attached to the same 2703.

4. IBM Integrated Communications Attachment (ICA) (#4640) (Available on the System/370 Model 135)

   Additional Lines (#4722-4728)

   For 2741 and 1050 Terminals

   - Terminal Adapter Type I Model II (#9721-9728)

     – Switched Network Facility (#9625-9632)

     – Write Interrupt (#9745-9752)

     – Read Interrupt (#9737-9744)

     – Unit Exception Suppression (#9729-9730) optional

   For 2780 Data Transmission Terminal Model 2

   - Synchronous Data Adapter Type II (#9649-9656)

     – Half Duplex Facility (#9617-9624)

     – EBCDIC Transparency (#9673-9680)

5. IBM 3705 Communications Controller (in 2701, 2702, or 2703 emulation mode)

   Note: The 3705 Emulator must be generated, loaded, and dumped by a DOS, DOS/VS, OS, or OS/VS virtual machine.

   - Required if 1050, 2741, and/or CPT-TWX (Model 33/35) terminals are to be controlled.

     – Attachment Base, Type I

     – Channel Adapter, Type 1 (#1541)

     – Business Machine Clock (#4650), one or more, as needed

     – Communications Scanner, Type 1 (#1641)

     – Line Interface Base (LIB), Type 1, 3, and/or 4 (see below)

   For 2741, 1050, 2780, and/or Line Control for CPT-TWX (Model 33/35) Terminals

   - Line Interface Base, Type 1 (#4701), if using the following:

     – Line Set, Type 1A (#4711), Low-speed local attachment (not for 2780)

     – Line Set, Type 1D (#4714), Medium-speed External Modem

   For 2741 Terminals

   - Line Interface Base, Type 3 (#4703)

     – Line Set, Type 3B (#4732), Limited Distance Type 1 Line Adapter, 4-wire

   For 2741 and 1050 Terminals

   - Line Interface Base, Type 4 (#4704)

     – Line Set, Type 4C (#4743), Leased line, Line Adapter, 4-wire

Additional Devices Utilized by VM/370

Direct Access Devices	*Paging*	*Spooling*	*CP SYSRES*	*CMS*
2314 Direct Access Storage Facility	x	x	x	x
2319 Disk Storage	x	x	x	x
3330 Disk Storage, Model 1	x	x	x	x
3333 Disk Storage, Model 1	x	x	x	x
2305 Fixed Head Storage[1]	x	x	x	

[1] The 2305 Model 2 can be attached to the System/370 Models 145, 155 II, 158, 165 II, and 168. The 2305 Model 1 can be attached to the System/370 Model 165 II and 168 only.

[1] Although not required, these features enhance the convenience and usability of these terminals.

- IBM 2780 Data Transmission Terminal Model 2, for remote spooling, with these features:

  - EBCDIC Transmission Code (#9762)
  - EBCDIC Transparency (#8030)
  - 144 Character Print Line (#5820, 5821)

Notes:

1. The Word Buffer feature (#8810), available only with the System/370 Model 145, as recommended for selector channels if 3330 Model 1 and 3333 Model 1 Disk Storage units are attached and required if a 2305 Model 2 Fixed Head Storage device is attached.
2. The standalone channels that attach to the System/370 Models 165 II and 168 require 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.
3. The requirement for the transmission control unit, line, and remote terminal can be eliminated for System/370 Models 145, 155 II, and 158 if the customer plans to run only two virtual machines using the primary and alternate system consoles. This permits a customer to install VM/370 earlier, and gain experience with multiple concurrent virtual machine operation and/or CMS time sharing before the installation of his teleprocessing terminals, control units, and lines.

Configurations Supported by CMS

- 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.
- Any virtual card readers, punches, and printers supported by VM/370 as spooling devices.
- Up to ten logical 2314, 2319, 3330 Model 1, or 3333 Model 1 direct access storage devices (3330 Model 1 and 3333 Model 1 logical disks under CMS can use a maximum of 246 cylinders each).
- Up to four 2400, 2415, 2420, or 3420 Magnetic Tape Units (7 or 9 track, 800 or 1600 bpi).

Two-Channel Switch

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

Note: If any I/O devices controlled by VM/370 for its own exclusive use are attached to control units with two channel switches, one of the channel interfaces in each of the control units *must* be disabled (that is, turned off).

Devices Used Only by an Operating System in a Virtual Machine and Not by VM/370

Any input/output device attachable 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 are used (except OS Indexed Sequential Access Method ISAM), and none of the other restrictions outlined in the publication *IBM Virtual Machine Facility/370: Introduction,* Order No. GC20-1800, 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: (1) unit record devices, which CP can simulate using spooling techniques; and (2) 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.

Minimum VM/370 Configuration

CPU	One of the System/370 Models previously designated.
Storage	At least 245,760 bytes
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	Telecommunications Control Unit (or the Integrated Communications Adapter on the System/370 Model 135)
One	Multiplexer Channel
One	Selector or Block Multiplexer Channel
One	Communications Terminal

The resident portion of the VM/370 control program requires approximately 80K bytes of real main storage plus an average of 2K bytes per active virtual machine.

Representative VM/370 Configuration

IBM 3145	512K Storage
IBM 3215	Console Printer-Keyboard
IBM 1403	Printer, Model N1–Two

IBM 3330	Model 1 or 3333 Model 1 Disk Storage–Four Drives attached to a 3830 Model 2
IBM 2305	Fixed Head Storage Facility, Model 2
IBM 3420	Magnetic Tape Units–Two
IBM 2540	Card Read Punch
IBM 3705	Communications Controller
IBM 2741	Communications Terminals (as needed)
Two	Block Multiplexer Channels (#1421-1422) with Word Buffer feature (#8810)
One	Multiplexer Channel

Estimating CP Storage Requirements

MAIN STORAGE

CP Resident Nucleus	80K
Control Blocks for representative machine configuration	4K *
Free Storage per active user	2K **

*Control blocks for machine configuration
96 bytes/real channel
64 bytes/real control unit
80 bytes/real device

Calculated for representative machine configuration (29 devices, 6 control units, 4 channels)

** Typical User
364 bytes for VMBLOK
40 bytes/virtual channel
40 bytes/virtual control unit
56 bytes/virtual device

SEGTABLE	128 bytes/machine
SWPTABLE	8 bytes/page
PAGETABLE	2 bytes/page

Typical User:

Machine	320K	364 + 128 + 800	1292
Reader	1		56
Printer	1		56
Punch	1		56
Disks	3	3 X 56	168
Channels	2	2 X 40	80
Console	1	1	56
Control Units	3	3 X 40	120
			1724

DASD:

	2314	*3330*	*2305*
CP Nucleus	2 cyl	1 cyl	3 cyl
Error Recording	2	2	2
Warm Start	1	1	1
Directory	1	1	1
Paging Space	32	18	40
Spooling Space	50	30	
Total System	88 cyl	53 cyl	47 cyl

Notes:

1. Paging and spooling space is installation dependent. The 2305 is normally used for paging only.

   Plus: Saved Systems
   System Residence Volumes (CMS, OS, etc.)
   User's dedicated disks and minidisks

2. The CP nucleus requires about 60 pages of disk space for resident and pagable functions; saved systems require 1 page for each page saved, plus an additional information page.

Sample Machine Configuration

IBM 3145, 512K of main storage
IBM 3215 Console Printer-Keyboard
IBM 1403 Printer
IBM 3330 Disk Storage–Four Drives
IBM 2305 Fixed Head Storage Facility, Model 2
IBM 3420 Two Magnetic Tape Units
IBM 2540 Card Read Punch
Two block multiplexer channels with Word Buffer feature (#8810)
One multiplexer channel
IBM 3705 Communications Controller (in 2701, 2702, or 2703 Emulation Mode)
IBM 2741; as many communication terminals as needed

VIRTUAL MACHINE OPTIONS

VM/370 provides three optional services to virtual machines. These options are:

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

2. The ISAM option, which allows the virtual machine to execute the self-modifying CCW command sequences generated by the OS ISAM modules. This option is not required for the proper functioning of ISAM in DOS. This option does *not* permit other types of self-modifying CCW sequences to function.

3. The ECMODE option, which allows the virtual machine to use the complete set of virtual System/370 control registers and the dynamic address translation feature of the System/370. Programming simulation and hardware features are combined to allow use of all the available features in the hardware.

The REALTIMER, ISAM, and ECMODE options increase the amount of CP overhead incurred by the virtual machines using them, and should not be given to a virtual machine unless they are required. These options are specified in the OPTION record in the VM/370 user directory. If a particular user requires an option only occasionally, an installation may choose to assign that user two virtual machines: one with the option specified, and one without; in this way, the additional overhead would be incurred only when necessary.

The REALTIMER can be set off by a user with the CP SET TIMER console function.

OS ISAM Channel Programs

Certain OS ISAM channel programs use a self-modifying operation that is not allowed under normal VM/370 processing. With the ISAM option selected, CP can scan the specific OS ISAM channel program to handle the self-modifying sequence properly. With the option selected and with the virtual machine using OS ISAM, CP restricts somewhat the location of the ISAM CCWs in the virtual machine. The programmer must assemble the ISAM program in such a way that four critical double words in certain channel programs cannot cross a page boundary. The following table lists the restricted channel programs and the critical double words:

Channel Program	*Critical Double Words*
CP1	C8, C9, C10, C10A
CP4/5	CA12, CA13, CA14, CA15
CP6	CA34, CA35, CA36, CA37
CP8	CB10, CB11, CB12, CB16
CP23	CS6, CS7, CS8, CS9
CP23	CS17, CS18, CS19, CS19A
CP26	CS34, CS35, CS36, CS37

Only those users with the ISAM option in the user directory have their CCW strings checked for self-modifying operation; thus not all users incur the additional CP overhead. This option is not needed for DOS ISAM.

Preferred Virtual Machines

VM/370 provides four special functions that create a special virtual machine operating environment. The functions are designed to improve the performance of a selected virtual machine. Although each function could be applied to a different virtual machine, this would not be normal practice if optimum performance is required for one specific virtual machine. Each function is discussed separately below.

1. Favored Execution

   VM/370 has the capability of providing an assured percentage of CPU time to a particular virtual machine. Assured in this sense means that VM/370 will attempt to provide up to the specified percentage of CPU time to that virtual machine, provided the virtual machine is functioning so that it can fully utilize the CPU time. At regular time intervals, the VM/370 dispatcher checks the CPU time used by the particular virtual machine. If the percentage has been exceeded, the machine is not run for the remainder of the interval. If the percentage is low, the virtual machine will have highest priority execution for the remainder of the interval. The percentage of CPU time assured is specified in the class A console function that invokes the option. The format of the console function is:

   SET FAVORED userid $\left\{\begin{matrix}\text{xx}\\ \text{OFF}\end{matrix}\right\}$

   where xx is any value from 1 to 99 and represents a percentage value. The OFF operand stops the operation of the function. Only a system operator with privilege class A can execute the command. The percentage of CPU time can be specified for only one virtual machine at a time.

   A second form of the SET FAVORED console function, with no CPU percentage specified, can be issued for several virtual machines.

   In both cases, virtual machines with the favored option active are kept in the dispatching queues at all times. When dropped from Q1 they are placed directly in Q2. They are not dropped from Q2 when they accumulate 4 seconds of CPU usage or when they enter console function mode. They are instead placed in the Q2 non-executable list. These virtual machines are thus considered for dispatching more frequently than other virtual machines.

2. Reserved Page Frames

   VM/370 uses chained lists of available and pageable pages. Pages for users are assigned from the available list, which is replenished from the pageable list.

   Pages which are temporarily locked in real storage are not available or pageable. The reserved page function gives a particular virtual machine an essentially "private" set of pages. The pages are not locked; they can be swapped, but only for the specified virtual machine. Paging proceeds using demand paging with a "reference bit" algorithm to select the best page for swapping. The number of reserved page frames for the virtual machine is specified as a maximum. The page selection algorithm selects an available page frame for a reserved user and marks that page frame "reserved" if the maximum specified for the user has not been reached. If an available reserved page frame is encountered for the reserved user selection, it is used whether or not the maximum has been reached.

   The maximum number of reserved page frames is specified by a class A console function of the following format:

   SET RESERVE userid xxx

   where xxx is the maximum number required. If the page selection algorithm cannot locate an available page for other users because they are all reserved, the algorithm will force stealing of reserved pages. This function can be specified in only one virtual machine at any one time.

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

3. Dedicated Channels

Since in most cases, the devices on a channel are shared between virtual machines (minidisks and dedicated disks) and system functions (paging and spooling), VM/370 must schedule all of the I/O requests to achieve a balance between machines.

Additionally, VM/370 must simulate the reflection of subsequent I/O interrupts to the virtual machines.

The VM/370 dedicated channel option allows VM/370 to bypass its device translation, channel scheduling, and interrupt reflection routines for all devices on that channel.

A virtual machine assigned a dedicated channel has that channel and all devices on that channel for its dedicated use.

Since no device translation is done by VM/370 for virtual machine I/O to a dedicated channel, all virtual machines using one or more dedicated channels must have real I/O devices attached:

a. virtual device addresses and real device addresses must be identical, and

b. minidisks are not allowed.

I/O operations to a device on a dedicated channel are not scheduled by VM/370, but are instead performed directly; the true condition code is reflected back to the virtual machine.

VM/370 interrupt stacking is bypassed by disabling the channel and having the hardware perform the interrupt stacking function.

Channels are dedicated to a virtual machine by using the ATTACH CHANNEL class B console function. A virtual machine may have one or more dedicated channels. The number of virtual machines using dedicated channels is limited only by the number of channels available on the real machine for dedication. A channel dedicated to a virtual machine cannot be used by VM/370 or by other virtual machines.

4. Virtual-Real

This option requires that the VM/370 nucleus be reorganized to provide an area in real storage large enough to contain the entire virtual=real machine. For the virtual machine, each page from page 1 to the end is in its true real storage location; only its page zero is relocated. The virtual machine is still run in dynamic address translation mode. Since the virtual page address is the same as the real page address, no CCW translation is required for the virtual machine. Since CCW translation is not performed, no check is made to ensure that I/O data transfer does not occur into page zero or any page beyond the end of the v=r machine's storage.

There are several considerations for the virtual=real (v=r) option that affect overall system operation:

a. The area of contiguous storage built for the v=r machine must be large enough to contain the entire addressing space of the largest v=r machine. The v=r storage size that a VM/370 system will allow is defined during system generation when the option is selected.

b. The storage reserved for the v=r machine can only be used by a virtual machine with that option specified in the VM/370 user directory. It is not available to other users for paging space, nor for VM/370 usage until released from v=r status by a system operator via the CP UNLOCK command. Once released, VM/370 must be loaded again before the v=r option can become active again.

c. The virtual machine with the v=r option operates in the pre-allocated storage area with normal CCW translation in effect until the CP SET NOTRANS ON command is issued. At that time, all subsequent I/O operations are performed from the virtual CCWs in the v=r space without translation. In this mode, the virtual machine must not perform I/O operations into page zero, nor beyond its addressable limit. Violation of this requirement may cause damage to the VM/370 system and to other virtual machines.

d. Since no CCW translation is being performed for virtual machine I/O when NOTRANS is ON, virtual machines cannot use minidisks while operating in this mode.

e. If the v=r machine performs a virtual "reset" or "ipl", then the normal CCW translation goes into effect until the CP SET NOTRANS ON console function is again issued. This permits simulation of an IPL sequence by CP. Only the v=r virtual machine can issue the command. A message is issued if normal translation mode is entered.

SYSTEM DEFINITION

Before starting the system generation procedures on a real machine, it is necessary to define certain parameters via card input. Three card decks must be prepared before starting to generate a VM/370 system:

- The Real I/O Configuration Deck (module name DMKRIO), which defines the I/O configuration on the real System/370 machine.

- The CP Control Deck (module name DMKSYS), which defines CP controlled DASD volumes, allocation, etc.
- The DIRECTORY Deck (normally a CMS file named USER DIRECT), which contains the VM/370 user directory entries that define the virtual machine configuration for each user.

VM/370 system generation procedures conducted on an existing VM/370 system, or changes to existing user entries, do not necessarily have to be accomplished via card deck input, but may instead be modified by using the EDIT facilities of CMS to alter the existing files.

PREPARING THE REAL I/O CONFIGURATION DECK (DMKRIO)

The Real I/O configuration deck consists of entries for each I/O device, control unit, and channel attached to the real System/370. Because of the nature of multiplexer channel operation, there may not be a one-for-one correspondence between real control units and RCTLUNIT macro entries. The macro sequence is:

Units Referred to	*Macro Name*
I/O Devices	RDEVICE
Control Units	RCTLUNIT
Channels	RCHANNEL
System Console	RIOGEN

The deck is placed in the reader as follows:

```
DMKRIO CSECT
          RDEVICE cards
                .
                .
                .
          RCTLUNIT cards
                .
                .
                .
          RCHANNEL cards
                .
                .
                .
          RIOGEN card
          END
```

All RDEVICE entries must appear first, followed by all RCTLUNIT entries, all RCHANNEL entries, and finally by the RIOGEN card. In addition the first card in the deck must be the DMKRIO CSECT card described above, and the last card must be a normal assembler END card.

RDEVICE Macro

The RDEVICE macro instruction is used to generate a Real Device Block (RDEVBLOK). An RDEVBLOK entry is required for each real I/O device in the I/O configuration. The format of the RDEVICE macro is:

Name	*Operation*	*Operands*
	RDEVICE	ADDRESS=[address|(address, number)] ,DEVTYPE=type [,MODEL=model] [,FEATURE=(feature[,feature]...)] [,CLASS=(class[,class] ...)] [,ADAPTER=adapter] [,SETADDR=sadnum]

Name Field: No name may be specified for the RDEVICE macro instruction. The macro generates a name by appending the device address to the characters RDV. For example, if the device address is 234, the name RDV234 is generated.

Operand Field:

ADDRESS=
specifies the real I/O device address or addresses of devices.

address
is three hexadecimal digits (ccu). They may be any hexadecimal characters 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.

number
is the number of RDEVBLOK entries to be generated. The value may be any number from 1 to 16. For example, if ADDRESS=(10, 5) is specified, RDEVBLOKS with device addresses 10, 11, 12, 13, and 14 would be generated. If this subparameter is omitted, a value of 1 is assumed for all devices except the 2314, 3330, and 2305, which have a default value of 8.

DEVTYPE=
specifies the type number of the device.

type
is a value that 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, 3066, 3158, 3210, 3211, 3215, 3277, 3284, 3286, 3330, 3410, 3420, 3505, 3525, or 3705. 2540R and 2540P refer to the same

IBM 2540 card read punch (as do 1442P and 1442R, and 2520P and 2520R). Each function must be specified in a separate RDEVICE macro.

MODEL=
specifies the model number, if any, of a tape device.

model
is a value that can be 1, 2, 3, 4, 5, 6, 7, or 8.

FEATURE=
specifies the optional features that are present on the device as one or more of the following values. These values can be written in any order.

Value	*Feature*
7-TRACK	7-track head on a tape drive
DUALDENS	Dual density on a tape drive
READWRIT	Simultaneous reading and writing on a 2401 or 2402
UNVCHSET	Universal character set printer

CLASS=
specifies a list of up to four spooling output classes separated by a comma. The CLASS parameter may only be specified for a 1403, 1443, or 3211 printer, or 2540P or 3525 card punch.

class . . . class
one alphabetic character per class, separated by commas.

ADAPTER=
specifies the terminal control or transmission adapter used to connect a telecommunications I/O device to its control unit. This parameter is required if a DEVTYPE of 2701, 2702, 2703, 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 or 3705.

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 3705.

TELE2
specifies that a CPT-TWX (Model 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 3705.

SETADDR=
specifies the set address (SAD) command to be issued for a telecommunications line attached to a 2702 control unit. The parameter is required if the device is a 2702.

Value	*Command*
0	SADZERO
1	SADONE
2	SADTWO
3	SADTHREE

Examples:
The following examples illustrate the use of the RDEVICE macro instruction 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
```

Usage: The RDEVICE macro instructions describe each device or group of devices attached to the System/370. These may be in any order, but they must be contiguous, and must precede all RCTLUNIT and RCHANNEL macro instructions in the Real I/O configuration deck (DMKRIO). The first RDEVICE macro instruction will also generate the label RDVSTART, which indicates the start of the Real Device Blocks to CP.

MNOTES: The following MNOTE messages may be generated by the RDEVICE macro instruction.

INVALID DEVICE ADDRESS
indicates that one or more of the device addresses coded was specified incorrectly.

INVALID DEVICE TYPE
indicates that the device specified was not a permitted specification.

MORE THAN 16 DEVICES
indicates that a value greater than 16 was specified in the number parameter of the ADDRESS operand.

INVALID MODEL NUMBER
the model number specified was not a permitted specification.

INVALID ADAPTER TYPE
indicates that the adapter specified was not a permitted specification, or that the ADAPTER operand was not coded.

INVALID SETADDR VALUE
indicates that the SETADDR value specified is not a permitted specification, or that the SETADDR operand was not coded when the DEVICE is a 2702.

INVALID FEATURE SPECIFIED
indicates that the feature specified is not a permitted specification.

INVALID CLASS PARAMETER
indicates that one or more of the classes specified was not an alphabetic character, or that more than four classes were coded, or that the syntax of the encoding is incorrect.

If any of the above errors are detected, the Real Device Block is not generated. The RDEVICE macro instruction will also generate an entry in a table of printers or a table of punches or a table of readers for spooling when DEVTYPE=1403, DEVTYPE=3211, DEVTYPE=1443, or DEVTYPE=2540P, DEVTYPE=3211, DEVTYPE=2540R, DEVTYPE=3505, or DEVTYPE=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 field is generated, but no entry is made in the printer or punch table; the device cannot be used for CP spooling.

RCTLUNIT Macro

RCTLUNIT macros define up to 8 or 16 devices. An RCTLUNIT macro that defines 8 devices must have an address divisible by 8; accordingly, an RCTLUNIT macro that defines 16 devices must have an address divisible by 16.

On the multiplexer channel, several device addresses may fall within the address range of one RCTLUNIT macro. When this occurs, only one RCTLUNIT macro need be coded, even though more than one real control unit may be present. 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 an address of 008, even though the system console and 2540 Card Read Punch have different real control units. The format of the RCTLUNIT macro is:

Name	*Operation*	*Operands*
	RCTLUNIT	ADDRESS=address ,CUTYPE=type [,FEATURE=feature]

Name Field: No name may be specified for the RCTLUNIT macro instruction. 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.

Operand Field:

ADDRESS=
specifies the address of the control unit.

address
is three 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 16-DEVICE is specified as a feature, the last digit must be 0. Otherwise, it may be either 0 or 8. If the second digit of the addresses of devices attached to the same physical control unit differs, the control unit must be represented by multiple Real Control Unit Blocks. For example, if an installation has a 2703 with device addresses 030 through 06F, the Real I/O configuration deck (DMKRIO) must contain four RCTLUNIT macro instructions with control unit addresses 030, 040, 050, and 060 respectively.

CUTYPE=
specifies the type number of the control unit.

type
is the device type number of the control unit. One of the following numbers must 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, 3066, 3158, 3210, 3215, 3272, 3345, 3411, 3505, 3525, 3705, 3803, 3811, 3830, ICA, IFA, ISC.

Even though some devices attach directly to the channel without a separate control unit, an RCTLUNIT macro instruction must be included for them.

FEATURE=
specifies the optional features present on the control unit.

16-DEVICE
indicates that the control unit is controlling more than eight devices. This may be specified for a 2403, 2803, 2848, 3803, 2702, 2703, 3705, 3272, 3830, ISC, or ICA.

Examples:

The following examples illustrate the use of the RCTLUNIT macro instruction to describe the control unit for a 3215 console printer-keyboard with address 009, a 2314, and a 3705 with lines 040 through 04B.

```
RCTLUNIT ADDRESS=008,CUTYPE=3215
RCTLUNIT ADDRESS=230,CUTYPE=2314
RCTLUNIT ADDRESS=040,CUTYPE=3705,
    FEATURE=16-DEVICE
```

Usage: The RCTLUNIT macro instructions describing the control units in the 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 RCUSTART, which indicates the start of the Real Control Unit Blocks to CP.

MNOTES: The following MNOTE messages may be generated by the RCTLUNIT macro instruction.

INVALID CONTROL UNIT ADDRESS
indicates that either the channel or control unit portion of the address is not in the range 0 to F, or that the last digit of the address is neither 0 nor 8.

INVALID CONTROL UNIT TYPE
indicates that the type number is not one of the permitted specifications.

INVALID CONTROL UNIT ADDRESS FOR 16-DEVICE FEATURE
indicates that FEATURE=16-DEVICE is specified but that the last digit of the control unit address is not 0.

INVALID CONTROL UNIT TYPE FOR 16-DEVICE FEATURE
indicates that FEATURE=16-DEVICE is specified but the type number is not one of the following: 2403, 2803, 2848, 3803, 2702, 2703, 3705, 3272, 3830, ISC, ICA.

INVALID FEATURE SPECIFIED
indicates that the argument to the FEATURE= keyword was not 16-DEVICE.

If any of the above errors are detected, the Real Control Unit Block is not generated. This results in undefined symbols in the Real Device Blocks for this control unit.

RCHANNEL Macro

An RCHANNEL macro instruction must be coded to define each real channel in the I/O configuration. The format of the RCHANNEL macro is:

Name	*Operation*	*Operands*
	RCHANNEL	ADDRESS=address ,CHTYPE=[SELECTOR|MULTIPLEXOR|BLKMXPR]

Name Field: No name may be specified for the RCHANNEL macro instruction. The 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.

Operand Field:

ADDRESS=
specifies the address of the channel.

address
is a hexadecimal number from 0 to F.

CHTYPE
specifies the type of channel.

SELECTOR
is a selector channel.

MULTIPLEXOR
is a byte-multiplexer channel.

BLKMPXR
is 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,TYPE=MULTI-
  PLEXOR
RCHANNEL ADDRESS=1,TYPE=SELECTOR
RCHANNEL ADDRESS=2,TYPE=BLKMPXR
```

Usage: The RCHANNEL macro instructions describing the channels in the installation's computing system 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 RCHSTART, which indicates the start of the Real Channel Blocks to CP.

MNOTES: The following MNOTE messages may be generated by the RCHANNEL macro instruction:

INVALID CHANNEL ADDRESS
indicates that the channel address specified is not in the range 0-F.

INVALID CHANNEL TYPE
indicates that the channel type specified is neither MULTIPLEXOR, BLKMPXR, or SELECTOR

If either of the above errors is detected, the Real Channel Block is not generated. This results in undefined symbols in the Real Control Unit Blocks for this channel.

RIOGEN Macro

The RIOGEN macro instruction is used to generate the channel index table and unit record and console tables. It must appear as the last macro instruction before the END card in the DMKRIO deck. The format of the RIOGEN macro is:

Name	*Operation*	*Operands*
	RIOGEN	CONS=xxx [,ALTCONS=xxx]

Name Field: No name may be specified for the RIOGEN macro instruction.

Operand Field:

xxx
specifies a hexadecimal device address that must have previously been specified in a RDEVICE macro instruction.

CONS
specifies the address of the primary system console. This device must either be a 3210, 3215, or 1052 (via a 2150 freestanding console), or a System Console for the 3158 plus a 3213 Printer (in 3215 Emulator Mode).

ALTCONS
specifies the address of the alternate console. This device, which should be located as close as possible to the primary system console, may be any device supported as a VM/370 login device. This console will be used if the main console is unavailable at initial program load time. If ALTCONS is not coded or the alternate console is not available, CP enters a disabled wait state with a wait state code of '005' in the instruction address register (IAR).

Alternate Console: Should the primary system console (specified in the RIOGEN macro) be inoperative, VM/370 automatically selects the alternate console (also specified in the RIOGEN macro) as the console for primary system operation. The system rings the CPU alarm to notify the operator that this selection was made.

When the device specified as the alternate console is a communication line, VM/370 enables the line. The operator can then establish a link with the computer through a terminal connected to the line. Once the line is established, VM/370 proceeds with the normal system initiation.

If the alternate console is an IBM 2741 Communication Terminal, it *must* use the EBCDIC transmission code.

If the primary operator's console becomes inoperative during a VM/370 session, pressing the INTERRUPT button on the real machine system control panel forces a disconnect of the operator, and allows him to reconnect using the alternate console or any other terminal with a communications line that has been previously enabled.

Examples:

```
RIOGEN CONS=01F, ALTCONS=050
RIOGEN CONS=009
```

PREPARING THE CP CONTROL DECK (DMKSYS)

SYSOWN Macro

The SYSOWN macro is used to generate the list of CP owned DASD volumes. A CP owned volume is either the CP system residence volume, or a volume which contains VM/370 paging, spooling, or temporary disk space (that is, contains a CP allocation table at CYL 0, REC 4 allocating paging, spooling, or temporary disk space). Even though a volume may have a VM/370 allocation table at CYL 0 REC 4, the allocation data will be ignored unless the volume appears as an argument in the SYSOWN macro instruction. The format of the SYSOWN macro is:

Name	Operation	Operands
	SYSOWN	(volid 1 [,pref]) [,(volid 1 [,pref]) . . .]

Name Field: No name may be specified for the SYSOWN macro instruction.

Operand Field:

volid 1
specifies a CP-owned volume serial number.

pref
indicates to VM/370 how allocatable space on the specified volume should be used. Pref may be either of the following:

TEMP
specifies that this volume is to be used for temporary and spool space allocation, and should only be used for paging space allocation in the event all volumes normally used for paging allocation are full or unavailable.

PAGE
specifies that this volume is to be used for paging and temporary disk allocation only.

If the pref option is not specified, TEMP is assumed.

Example:

SYSOWN (CPDRM1,PAGE),(CPDSK1,TEMP),
CPDSK2

MNOTES: The following MNOTE messages may be generated by the SYSOWN macro instruction:

INVALID PREFERENCE OPTION
indicates that the second positional sublist parameter following volid is not TEMP or PAGE.

MORE THAN 255 VOLUMES SPECIFIED
indicates that more than 255 owned volumes have been specified in the SYSOWN macro instruction.

DUPLICATE VOLUME SERIAL SPECIFIED
indicates that the same volume serial number was specified more than once.

NO VOLUMES SPECIFIED
indicates that the SYSOWN macro instruction was specified without any positional parameters.

WARNING–NO TEMP SPACE SPECIFIED
indicates that no volume was specified as being preferred for TEMP space allocation. This means that no spooling operations may be performed. Any data transfer channel program started to a virtual unit record output device will end with UC status in the virtual CSW and the Intervention Required bit set in the virtual sense byte.

SYSRES Macro

The SYSRES macro instruction is used to describe the characteristics of the CP system residence volume. The format of the SYSRES macro is:

Name	Operation	Operands

	SYSRES	SYSVOL=serial,
		SYSRES=address,
		SYSTYPE=(2314 | 3330 |
		2305),
		SYSNUC=start-cylinder,
		SYSERR=cylinder,
		SYSWRM=cylinder

Name Field: No name may be specified for the SYSRES macro instruction.

Operand Field:

SYSVOL=
specifies the volume serial of the system residence disk.

serial
is a character string with a maximum length of 6 characters.

SYSRES=
specifies the device address where the system residence volume is mounted. This address is only used when the VM/370 nucleus is generated and written on the disk. Thereafter, the system can be IPLed from any addressable disk device.

address
is a three digit hexadecimal device address.

SYSTYPE=
specifies the device type of the system residence device.

SYSNUC=
specifies the starting real cylinder number where the CP nucleus resides.

start-cylinder
is a one to three digit decimal number. (A 2314 or 2319 disk will require 2 contiguous cylinders for CP residence, a 2305 will require 3 contiguous cylinders, and a 3330 disk will require one cylinder.)

SYSERR=
specifies the starting cylinder where error recording records are written.

cylinder
is a one to three digit decimal number. Two contiguous cylinders are required for error recording.

SYSWRM=
specifies the cylinder which is used by CP for saving warm start information.

cylinder
is a one to three digit decimal number.

Notes:

1. The cylinders required for SYSNUC, SYSERR, and SYSWRM *must* be formatted using the CP FORMAT service program, and *must* be allocated as PERM space on the SYSRES volume.
2. VM/370 allows the 2314 "alternate track" cylinders 200-203 to be used for normal data if not needed for defective track assignment.

Example:

```
SYSRES SYSVOL=CPDSK1,SYSRES=230,
    SYSTYPE=2314,SYSNUC=198,SYSERR=4,
    SYSWRM=202
```

MNOTES: The following MNOTE messages may be generated by the SYSRES macro instruction:

INVALID DEVICE TYPE FOR SYSRES
indicates that the argument of the SYSTYPE parameter is not 2314, 2305, or 3330.

SYSVOL NOT IN OWNED LIST
indicates that the argument of the SYSVOL parameter is a volume serial number that has not been previously specified in the SYSOWN macro instruction. This error may be caused by the fact that the SYSOWN macro instruction does not appear in an assembly before the SYSRES macro instruction.

SYSOPR Macro

The SYSOPR macro instruction is used 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 format of the SYSOPR macro is:

Name	*Operation*	*Operands*
	SYSOPR	[SYSOPER=(OPERATOR | userid)] [SYSDUMP=(OPERATNS | userid)]

Name Field: No name may be specified for the SYSOPR macro instruction.

Operand Field:

SYSOPER=
specifies the userid of the virtual machine to be assigned to the system operator. If SYSOPER is not specified, the userid "OPERATOR" will be used.

userid
is a character string up to 8 characters long.

SYSDUMP=
specifies the userid of the virtual machine whose spool input will contain the system dump file after a system restart. If SYSDUMP is not specified, the userid "OPERATNS" will be used.

userid
is a character string up to 8 characters long.

Example:

```
SYSOPR SYSOPER=OP,SYSDUMP=CPSYS
```

SYSCOR Macro

The SYSCOR macro instruction is used to generate the internal control block called the CORTABLE. The format of the SYSCOR macro is:

Name	*Operation*	*Operands*
	SYSCOR	RMSIZE=(xxxxxK | yyM)

Name Field: No name may be specified for the SYSCOR macro instruction.

Operand Field:

RMSIZE=

specifies the amount of real main storage available for VM/370. This value limits the amount of main storage used by VM/370 if it is less than the total amount of main storage available.

xxxxx

is a 3 to 5 digit number used to denote the amount of storage in terms of "K" bytes, where 1K byte = 1024 bytes. This value may range from 240K to 16392K. It must always be a multiple of 2.

yy

is a 1 or 2 digit number used to denote the amount of storage in terms of "M" bytes, where 1M = 1024K bytes. This value may range from 1M to 16M.

Examples:

SYSCOR RMSIZE=256K
SYSCOR RMSIZE=1M

MNOTES: The following MNOTE message may be generated by the SYSCOR macro instruction:

INVALID RMSIZE SPECIFIED

indicates that the real machine size was either less than 240K, greater than 16M, or not a multiple of 2K.

SYSTIME Macro

The SYSTIME macro instruction is used 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 format of the SYSTIME macro is:

<table>
<tr><th><i>Name</i></th><th><i>Operation</i></th><th><i>Operands</i></th></tr>
<tr><td></td><td>SYSTIME</td><td>[ZONE=(0 | h | (h,m) | (h,m,s) |
(h,,s))]
[,LOC=(EAST | WEST)]
[,ID=(GMT | XXX)]</td></tr>
</table>

Name Field: No name may be specified for the SYSTIME macro instruction.

Operand Field:

ZONE=

specifies the time zone differential from Greenwich Mean Time. If ZONE is not specified, a value of 0 hours (Greenwich Mean Time) will be used.

h

is a number which 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.

m

is a number which represents minutes.

s

is a number which represents seconds.

LOC=

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=

specifies the name of the time zone. The default for ID is GMT.

xxx

is a 3 character string.

Examples:

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)

MNOTES: The following MNOTE messages may be generated by the SYSTIME macro instruction:

INVALID LOC SPECIFIED

indicates that the argument of the LOC parameter is not EAST or WEST.

ZONE GREATER THAN 13 HOURS

indicates that the total value of the hours, minutes, and seconds subfields of ZONE is greater than 13 hours.

ZERO LENGTH ID SPECIFIED

indicates that a zero length ID is specified.

INVALID ZONE

indicates that either the HOURS subfield of ZONE is missing, or more than 3 subfields were specified.

SYSLOCS Macro

The SYSLOCS macro instruction is a required macro used to generate internal pointer variables.

Name	*Operation*	*Operands*
	SYSLOCS	

Name Field: No name may be specified for the SYSLOCS macro instruction.

Operand Field: No operand is required for the SYSLOCS macro. If one is specified, it is ignored.

Example:

```
SYSLOCS
```

VM/370 SERVICE PROGRAMS USED DURING SYSTEM GENERATION

The following VM/370 service programs are required for system generation:

VM/370 Directory Processor (DMKDIR)
DASD Dump/Restore (DMKDDR)
DASD Format/Allocate (DMKFMT)

Many commands and keywords used to control these service programs may be truncated (abbreviated). Where such truncation is permitted, the shortest acceptable version of the command or keyword is represented by capital letters, with the optional part represented by lower case letters. Other notational conventions are as follows:

1. Uppercase letters and punctuation marks (except as described in items 3 through 5) represent information that must be coded exactly as shown.
2. Except as noted previously and in item 6, lowercase letters and terms represent information that you must supply.
3. Information contained within brackets [ ] represents an option that you can include or omit, depending on the requirements of the statement.
4. Options contained within braces { } represent alternatives, one of which must be chosen.
5. An ellipsis (a series of three periods) indicates that a variable number of items may be included.
6. Underlined elements represent an option that is assumed if a parameter is omitted.
7. When options appear in a string, they are separated by a vertical bar (logical OR).

VM/370 Directory Creation—General Information

The VM/370 user directory is maintained on the VM/370 system residence disk, and is pointed to by the VOL1 label (cylinder 0, track 0, record 3). A VM/370 directory processor program (DIRECT, module DMKDIR) processes the installation-prepared user description records and writes the VM/370 user directory on disk.

To create a VM/370 user directory:

1. Prepare directory input cards or card-images.
   a. Prepare user description records containing the accounting data, options, and virtual machine configurations for each userid to appear in the directory. A description of these records can be found in "User Directory Entries".
      Virtual machines capable of performing the following functions must be included:
      - Operator's virtual machines to:
        Control the VM/370 sessions.
        Control allocation of machine resources.
        Control spooling activity.
        Maintain online disk areas.

      Virtual machines capable of performing the following functions should be included:
      - System Analyst's virtual machines to:
        Perform system analysis.
        Modify certain VM/370 functions.
      - Maintenance virtual machines to maintain or operate:
        The CP system.
        The CMS system.
        Other operating systems run in the virtual machine environment.
        The hardware.
        The Installation Verification Procedure.
   b. Prepare a directory record which defines the device on which the user directory is to be written. This record must be the first in the input to the directory program, and is followed by the sets of records describing the user virtual machines.
2. Using the CP FORMAT service program, format and allocate the cylinders to be used for the directory. Cylinders must be allocated as DRCT. The number of cylinders required can be calculated with the following formula:

For 3330	For 2314, 2319
NR/57	NR/32

$$\text{where NR} = \frac{\text{Number of USER Cards}}{169} + \frac{\text{MDISK cards X 2 + all other cards including user cards}}{170}$$

Note: The user should initially format and allocate space for two VM/370 user directories. If this is done, he can then build a new user directory whenever needed, without overlapping the current one, and without formatting and allocating space each time a new directory is created. When writing a user directory, space is allocated from available cylinders on a cylinder basis, and a minimum of 2 cylinders are used as DRCT. Once a new user directory is successfully written, cylinders used for the old directory (marked as temporarily allocated during directory creation) are marked as allocated. In this way, records in the DRCT cylinders are freed and can be reused for the next directory creation. If space for two directories is not initially allocated, a new directory space must be allocated each time a new directory is created.

3. Execute the VM/370 Directory service program. This program can be run under CMS (using the DIRECT command). The same version of the Directory program is provided in object deck form (a three card loader, followed by the DMKDIR text deck), and may be IPLed directly from either a real or virtual reader.

   Under CMS, input records must be in a CMS file with a default file ID of "USER DIRECT". The DIRECT command loads the directory creation module. If no 'filename' is specified, the program looks for a file of USER DIRECT. Otherwise, it looks for a file of 'filename DIRECT'.

   If the file is not found, or if an error occurs during processing, the user directory is not created and the old directory remains unaltered.

   Normal completion posts the DASD address of the new user directory in the VOL1 label, and if updating the active system directory, it places the new user directory in use by VM/370.

   The virtual machine running the directory program must have write access to the volume to contain the new directory. If a directory is being written on the active VM/370 system residence volume, this access is provided through the current directory entry for a user who is to update the VM/370 system directory.

Example: A virtual machine for online directory maintenance.

```
USER OPERATIONS PASSWORD 256K 1M ABC
  ACCOUNT NUMBER BIN2
    CONSOLE 01F 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 W RPASS WPASS
    MDISK 331 3330 0 404 SYSWRK W RPASS WPASS
    MDISK 230 2314 0 203 UDISK1 R RPASS WPASS
    MDISK 231 2314 0 203 UDISK2 R RPASS WPASS
    MDISK 232 2314 0 203 BATCH1 R RPASS WPASS
    MDISK 233 2314 0 203 BATCH2 R RPASS WPASS
```

Using CMS and the context editor, the user named OPERATONS can create, alter, and maintain a card image file of the user directory input. When ready to write a new user directory, he issues the command:

```
DIRECT filename
```

where filename is a CMS file (normally named USER) with filetype DIRECT containing the necessary directory records.

Loading the DMKDIR object deck via the card reader is the same as CMS operation with two exceptions:

1. After IPL, the program asks the operator for the address of a card reader containing the directory input.
2. Normal completion updates the directory, and the next IPL of VM/370 places the directory in use.

Note: Updating of the active VM/370 user directory by a class A, B, or C user will place the directory in use by the system.

Directory Creation Service Program (DMKDIR)

The VM/370 Directory service program records the configuration of each user's virtual machine in the VM/370 user 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 may be placed in the card reader and IPLed directly, or run in a virtual machine under CMS. For system generation, the standalone facility must be used.

The CMS file named DIRECT may be updated with the CMS context editor to provide additional directory entries.

The CMS DIRECT Command

```
DIRECT [ filename [ filetype [ filemode ] ] ] [ (EDIT) ]
```

The DIRECT command filename and filetype default to a CMS file identification of "USER DIRECT". Any or all of the default can be overridden by the command line. The EDIT option allows the user to run the program without updating the directory on disk.

User Directory Entries

Input for the directory entry statements must be in the following formats, with one or more blanks or commas as delimiters. All entries are positional from left to right. If any entries are defaulted, all remaining entries on that input line must be defaulted, with the exception of the OPTION card. Its entries are self-defining and not positional.

All data after the last possible operand on any card is ignored. Also, blank cards and cards with the first operand an asterisk (*) are ignored.

If any input card is found to be in error, the program continues to build the directory, checking all input cards for validity before termination. After an abnormal termination or an EDIT run, the old directory is not altered, and the new directory is not saved.

```
DIRectory ccu devtype volser
```

The DIRECTORY statement defines the device on which the directory is allocated. It must be the first statement.

ccu The input device address, specified in 1 to 3 hexadecimal digits.

devtype Four decimal digits that represent a supported device type suitable for the VM/370 user directory (2314, 2319, 3330 or 2305).

volser The volume serial numbers of the directory volume (1 to 6 decimal digits).

For example:

DIRECTORY 234 3330 VMDSK1

specifies that the directory is to be written on a 3330 device at address 234 on a volume labeled VMDSK1.

```
                                         ┌lc   ┌ld   ┌cd   ┌es  ┐
User userid pass [stor [mstor [cl [pri   │ON   │ON   │ON   │ON  │
                                         └OFF  └OFF  └OFF  └OFF ┘
                    ┐ ┐ ┐
                    │ │ │ ]]]]
                    ┘ ┘ ┘
```

The USER card defines a user machine description and creates a user directory entry. It delimits the user directory entries for one user. A separate USER card must be punched for each entry required.

userid Is a one to eight character user identification (userid), used to identify the virtual machine. Any characters, numbers or special characters may be used.

pass Is a one to eight character password, which must be entered by the user to gain access. Any characters, numbers or special characters may be used.

stor Is one to eight decimal digits that defines the user's normal virtual storage size. The value entered must be a multiple of 4K. The last character must be K or M. The default is 128K. The minimum machine size is 8K. All entries not specified in a 4K increment will be rounded up to the next 4K boundary. The maximum size is 16M.

mstor Is one to eight decimal digits that defines the maximum virtual storage size the user is permitted to define as his storage after logging on the system. It must be a multiple of 4K. The last character must be K or M. The default size is 1M. All entries not specified in a 4K increment will be rounded up to the next 4K boundary. The minimum size is 8K.

This entry allows a user to temporarily use a larger size storage for large, infrequently used programs, but normally only use (and be billed for) the smaller "normal" size.

cl Is one to eight letters from A to H denoting the CP command privilege class or classes given to this user. The default is G, for general user. The privilege classes are:

A – System Operator
B – Resource Operator
C – System Programmer
D – Spooling Operator
E – System Analyst
F – Service Representative
G – General User
H – Not used

pri Is a 1 or 2 digit priority number from one to ninety-nine used by the dispatching scheme. 1 is the highest priority, 99 is the lowest, and 50 is the default.

The following special VM/370 logical edit characters 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 characters using the TERMINAL command. The default value for all characters is ON.

le Is a one character logical line end symbol or a two character hexadecimal representation of the symbol. ON will give the system default of #. OFF will turn the feature off.

ld Is a one character logical line delete symbol or a two character hexadecimal representation of the symbol. ON will give the system default of ¢. OFF will turn the feature off.

cd Is a one character logical character delete symbol or a two character hexadecimal representation of the symbol. ON will give the system default of @. OFF will turn the feature off.

es Is a one character edit escape character or a two character hexadecimal representation of the symbol. ON will give the system default of ”. OFF will turn the feature off.

```
Account number [distribution]
```

The ACCOUNT card defines a one to eight character account number and an eight character distribution identification. If specified, this card must follow the USER card and precede the first DEVICE card. The ACCOUNT card is optional. If the ACCOUNT record is omitted, the number and distribution will default to the userid in the USER record.

number Is a one to eight character account number which will be punched in the accounting data for this virtual machine. The userid from the user record is also punched in the accounting data.

distribution Is a one to eight character distribution identification which will be printed or punched with the userid as a separator for spooled output for this user. This value is optional and defaults to the userid from the USER record if omitted.

```
Option Realtimer Ecmode Isam Virt=real
```

The OPTION card selects specific options available to the user. This card must follow the USER card and precede the first device card. The OPTION card is optional. Any or all options may be present in any order.

Note: The ECMODE option must be selected to perform concurrent maintenance on the VM/370 system as discussed under “Concurrent Maintenance Facilities”. The ECMODE option must also be specified for virtual machines using operating systems which themselves use the Dynamic Address Translation facility, such as OS/VS1, OS/VS2, and VM/370 itself.

REALTIMER Provides a timer for the virtual machine that will be updated during virtual CPU run time and also during virtual wait time. This option is required for virtual machines running systems or programs that go into a wait state expecting a timer interrupt. This facility can also be obtained by issuing the CP console function SET TIMER REAL.

ECMODE Allows the virtual machine to run in extended control mode. This feature is required for virtual machines that intend to ipl systems that operate in System/370 extended control mode, such as VM/370 itself.

ISAM Provides special channel command word translation routines which permit OS ISAM programs (which dynamically modify their CCWs) to operate properly in a virtual machine. This is required only for virtual machines that will IPL OS and use OS ISAM access methods.

VIRT=REAL This option is primarily a performance option which 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 (except for OS ISAM programs which are handled with the ISAM option). VIRT= REAL can be specified for any number of virtual machines but only one virtual machine can use this facility at any given time.

```
Ipl    iplsys
```

The IPL card contains a one to eight character name of the system (or 1 to 3 digit I/O device address) to be IPLed, for the user, at LOGIN time. This card must follow the USER card, and precede the first device card (CONSOLE, MDISK, or SPOOL). The IPL card is optional. This entry may be overridden by the user at login time by specifying “LOGIN USERID NOIPL”.

```
Console      ccu      devtype
```

The CONSOLE record identifier specifies the virtual console device.

ccu Is the 1 to 3 character virtual device address.

devtype 1052
3210
3215

For example: CONSOLE 009 3210

```
Mdisk ccu devtype { cylr  } cyls volser [mode [pr [pw
                  {T-disk}
          [pm]]]]
```

The MDISK record identifier describes a physical extent on a DASD device to be owned by that user.

ccu Is the 1 to 3 character virtual device address.

devtype 2305
2311 TOP (top half of a 2314-2319)
2311 BOTTOM (bottom half of a 2314-2319)
2314
2319
3330

cylr Is a three digit decimal cylinder relocation factor which specifies the cylinder on a real disk that corresponds to cylinder 0 of the virtual disk. If T-disk is specified, disk space is obtained at login time from pre-allocated system disk space. This space must be initialized or formatted by the user when he logs in and is a part of his virtual configuration until he logs off or detaches the disk, then the data area is returned for re-allocation for another T-disk area. To maintain security this area should be erased before returning it.

cyls Is a one to three digit decimal number specifying the number of cylinders.

volser Is the volume serial number of the DASD unit.

mode Specifies the primary access requested (read/only, write, or multiple), 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 will not be 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 will not be 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 event, a read link is to be given, with an error 1 or 2, and the 'DEV XX FORCED R/O' message.

MW Specifies that a write link is to be given to the disk in any event.

If the mode is omitted from the statement, the default is to W.

pr Is the password that allows sharing in read mode.

pw Is the password that allows sharing in write mode.

pm Is the password that allows sharing in multiple-write mode.

For example:

```
MDISK 230 3330 5 10 WORK01 W
```

specifies an MDISK record for a minidisk with read/write access to 10 cylinders located on a real 3330 labeled WORK01, beginning at real cylinder 5.

```
MDISK 191 2314 50 15 CPDSK4 W RDPASS WRX2*
```

specifies an MDISK record 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 record (see the description of the LINK card in this section) or the LINK command. Note that PW cannot be given without PR, and PM without PW and PR.

```
Spool    ccu    devtype    [class]
```

The SPOOL card specifies the input and output unit record device used for spooling.

ccu Is a 1 to 3 character virtual device address.

devtype 1403
1443
3211
2540 Punch
Reader
3525
2501
3505

class For spool output devices, class is a one character output class which governs the eventual punching or printing of the real spooled output. For spool input devices, the one character input class is used to control access to spool files by virtual card readers. This parameter is required for all output devices defined on the spool record. The default class is A.

For example:

SPOOL 00E 1403 A

specifies a SPOOL record for a virtual 1403 at address 00E. The output class is A.

```
Dedicate    ccu { rdev           }
                { [VOLID] volser }
```

The DEDICATE card specifies that a device is to be dedicated to this user. If the device is a unit record device, input or output is not to be spooled by VM/370.

ccu Indicates a one to three character virtual device for this user.

rdev Indicates a one to three character real device address.

VOLID Is an optional keyword used if the VOLSER is less than 4 characters.

volser Indicates a 1 to 6 character volume serial number of a DASD device.

For example:

DEDICATE 0B8 0B0

specifies a device record for a device at real address 0B0. Its virtual address is 0B8.

DEDICATE 250 MYPACK

specifies a dedicate record that defines for the virtual machine as virtual address 250 the real device where DASD volume MYPACK is mounted.

```
Link    userid    ldev    [ccu  [mode] ]
```

The LINK record specifies a virtual device that this user has access to, but is defined as an MDISK record for the user specified as userid.

userid Indicates a 1 to 8 character user identification of the user who owns the disk you wish to link to.

ldev Indicates a 1 to 3 character virtual device address of the device owned by USERID.

ccu Indicates a 1 to 3 character virtual device address of the user's device. It will default to the same address as the link to device.

mode Specifies the primary access requested (read/only, write, or multiple), 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 link is not given 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; in this case, a read/only link is to be given.

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, with an error 1 or 2, and the 'DEV XXX FORCED R/O' message is issued.

MW Specifies that a write link is to be given to the disk in any event.

If the mode is omitted from the statement, the default is to "R".

```
Special      ccu      devtype
```

The SPECIAL card specifies I/O devices or control units which can be made available to the user, but are not specified by some other card.

ccu indicates a 1 to 3 character virtual device address.

devtype 2701 $\begin{Bmatrix}\text{IBM}\\ \text{Tele}\end{Bmatrix}$ (virtual 270x only)

2702 $\begin{Bmatrix}\text{IBM}\\ \text{Tele}\end{Bmatrix}$

2703 $\begin{Bmatrix}\text{IBM}\\ \text{Tele}\end{Bmatrix}$

CTCA (channel to channel adapter)

TIMER (pseudo-timer device)

DASD Dump Restore Service Program (DMKDDR)

The DASD Dump Restore (DDR) service program may be used to DUMP, RESTORE, COPY, TYPE, or PRINT a disk. The program may also be IPLed directly from the card reader, or invoked under CMS via the DDR command.

For the purpose of the initial system generation from the PID tape containing the VM/370 starter system, the stand-alone DDR deck must be IPLed from the tape.

The DDR Command Line

```
DDR [filename filetype [filemode] ]
```

The DDR command invokes the program under CMS. If no filename and filetype are provided the program prompts for control statements from the terminal, or it reads the control statements from the CMS file. The filemode defaults to an asterisk (*) if a value is not provided.

Control Statements

All DDR control statements may be entered from the system console or a card reader. Only columns 1 to 71 are inspected by the program. All data items after the last possible parameter for the statement are ignored. An at sign (@) will be treated as a logical backspace and a cent sign (¢) will cancel the whole line and result in a new "read" to the console or card reader. An output tape must have its cylinder header records in ascending sequences; therefore, the extents must be entered in sequence by recorded cylinders. Only one type of function–dump, restore, print, or copy–may be executed in one step, but up to 20 statements describing cylinder extents may be entered. The function statements will be delimited by detection of an input or output statement, or a null line if the console is used for input. If additional functions are to be performed, then the sequence must be repeated. Only those statements needed to redefine the I/O definitions are necessary for subsequent steps. All other I/O definitions will remain the same.

The PRINT and TYPE statements work differently in that they operate on only one data extent at a time. If the input is from a tape created by the DUMP function, it must be positioned at the header record for each step. The PRINT and TYPE statements have an implied output of either the console (type) or the SYSPRINT device (print). Therefore, PRINT and TYPE statements need not be delimited by an input or output statement.

DDR Functions

DUMP

requests the program to move data from a direct access volume onto a magnetic tape or tapes. The data is moved cylinder by cylinder. Any number of cylinders may be moved. The format of the resulting tape is:

Record 1: a volume header record, consisting of data describing the volumes.

Record 2: a track header record, consisting of a list of count fields to restore the track, and the number of data records written on tape. After the last count field the record contains key and data records to fill the 4K buffer.

Record 3: track data records, consisting of key and data records packed into 4K blocks, with the last record truncated.

Record 4: the end of volume or end of job trailer label, containing the same information as the next volume header record.

RESTORE

Is used to return data which has been dumped by this program. Data can be restored only to a DASD volume of the same or equivalent device type as it was dumped from. It is possible to dump from a real disk and restore to a minidisk as long as the device types are the same.

COPY
Is used to copy data from one device to another device of the same or equivalent type. Data may be reordered on a cylinder basis from input device to output device.

PRINT
Is used to print a translation of each record specified on the SYSPRINT device.

TYPE
Is used to type a translation of each record specified on the console.

I/O Definition Statements

```
{INput }  ccu type [volser] (SKip nn MOde {1600}
{OUTput}           [altape]               { 800}
                   {REwind}
                   {UNload} [)]
                   {LEave }
```

The input or output card is used to describe all TAPE and DASD units used.

ccu	Indicates the 1 to 3 character device address.
type	2314, 2319, 3330, 2305-2, 2400, 2420, or 3420 (no 7 track support).
volser	Is the volume serial number of a DASD device.
altape	Is the address of an alternate tape drive.

The options are delimited by a left parenthesis. The right parenthesis is optional. The options only apply to tape and may be entered without redefining the input or output definition. The default options are SKIP 0 and UNLOAD. (No mode will be set).

Note: The defaults are assigned when the device is defined. All options entered will then override any previous definition. All options are self-defining and not positional.

SKIP nn will forward space file the number of times specified. nn is any number up to 255.
Note: The SKIP option is reset to zero after the tape has been positioned.

The MODE option will cause all output tapes that are opened for the first time and at the load point to be set to the specified mode. All subsequent tapes mounted are also set to the specified mode. If no mode option is specified, then no mode set is performed.

```
1600 = 9 track 1600 bpi X'C3'
 800 = 9 track  800 bpi X'CE'
```

REWIND	rewinds the tape at the end of a function.
UNLOAD	rewinds and unloads the tape at the end of a function.
LEAVE	leaves the tape positioned after the tape mark at the completion of a function.

```
SYsprint  ccu
```

This statement describes a printer device that is used to print data extents specified by the PRINT statement. It is also used to print a map of the cylinder extents from the DUMP, RESTORE, or COPY statement. If the SYSPRINT statement is not provided, the default is 00E.

ccu indicates the 1 to 3 character device address.

```
{DUmp   }  {ccc  }
{COpy   }  {CPvol} [To] [ccc[Reorder] [To][ccc] ]
{REstore}  {All  }
```

If the 'ccc' parameter is used, then only those cylinders specified are moved, starting with the first track of the first 'ccc', and ending with the last track of the second 'ccc'. The REORDER parameter causes the output to be reordered, starting at the specified cylinder. If ALL is specified, then all cylinders are dumped. If CPVOL is specified, then cylinder 0 and all allocated directory and permanent space is dumped or copied as indicated by the I/O definition statements from the input device. In the case of a tape input function specifying CPVOL, all the data is restored or copied. It is up to the user to ensure that all the data not restored or copied is in the proper format, as described in the VM/370 allocation record. The TO and REORDER keywords are optional. Consider the following example:

```
INPUT 191 3330 SYSRES
OUTPUT 180 2400 181 (MODE 800
SYSPRINT 00F
DUMP CPVOL
INPUT 130 3330 MINI01
DUMP 1 TO 50 REORDER 51
60 70 101
```

This example sets the mode to 800 bpi, then dumps all pertinent data from the volume labeled 'SYSRES' onto the tape that is mounted on unit 180. If the program runs out of room on the first tape, it continues dumping onto the alternate device (181). While dumping, a map of the cylinders dumped is printed on unit 00F. When the first function is complete, the volume labeled 'MINI01' is dumped onto a new tape. Its cylinder header records are labeled 51 to 100. A map of the cylinders dumped is printed on unit 00F. Next, cylinders 60 to 70 are dumped and labeled 101 to 111. This extent is added to the cylinder map on unit 00F. When the DDR processing is complete, the tapes are unloaded and the program stops.

{PRint Type}	cc1 [hh1 [rr1]] [To cc2 [hh2 [rr2]]]

The PRINT and TYPE functions print or type a hexadecimal and EBCDIC translation of each record specified. The input device must be defined as direct access or tape. The output is directed to the system console for the TYPE function, or to the SYSPRINT device for the PRINT function. (This causes redefinition of the output unit definition.)

cc1 The starting cylinder. This must be the first operand after the command.

hh1 The starting track. If present, it must follow the cc1 operand. The default is track zero.

rr1 The starting record. If present, it must follow the hh1 operand. The default is home address and record zero.

TO A keyword operand delimiting the starting and ending address. If not present, then only one cylinder, track, or record is printed, depending upon the definition of other operands.

cc2 The ending cylinder. This must be the first operand after the TO operand.

hh2 The ending track. If present, it must follow the cc2 operand. The default is the last track on the ending cylinder.

rr2 The record ID of the last record to print. The default is the last record on the ending track.

Consider the following examples:

```
PRINT 0 TO 3
```

Prints all of the records from cylinders 0, 1, 2, and 3.

```
PRINT 0 1 3
```

Prints only one record, from cylinder 0, track 1, record 3.

```
PRINT 1 10 3 TO 15 4
```

Prints all records starting with cylinder 1, track 10, record 3, and ending with cylinder 1, track 15, record 4.

MINIDASD Service Program (DMKMDA)

The MINIDASD program formats real or virtual VM/370 disk volumes for OS and DOS use. It is not to be used to format CP disk areas (for paging, spooling, and so forth), nor is it to be used to format CMS disk areas. It initializes disk surfaces as the OS DASDI program does, except that: (1) it can execute under CMS, and (2) it has the ability to format minidisks.

Initializing a Minidisk

The first function of the MINIDASD program is to initialize a minidisk. A minidisk can be initialized with or without a surface analysis (i.e., a test for defective tracks); a surface analysis should be included when a minidisk is initialized for the first time.

Initialization With Surface Analysis: The MINIDASD program:

- Checks for tracks that have been previously designated as defective (flagged) and have had alternates assigned. The program automatically assigns alternates (disk devices only). This test must be suppressed when a disk recording surface is being initialized for the first time.
- Performs a surface analysis of each track and automatically assigns alternates, if necessary (non-drum storage minidisks only). Tracks that are available for disposition as alternates are checked first.
- Writes a standard home address, a track descriptor record (record 0), and erases the remainder of each track.
- Writes IPL records on track 0 (records 1 and 2).
- Writes volume label on track 0 (record 3) and provides space for additional records, if requested.
- Constructs and writes a volume table of contents (VTOC).
- Writes IPL program, if requested, on track 0 (2314, 2305, or 3330).

Initialization Without Surface Analysis: The MINIDASD program:

- Checks for tracks that have been previously designated as defective (flagged) and have had alternates assigned. The program automatically assigns alternates (disk devices only). This test must not be suppressed.

- Writes a standard home address, a track descriptor record (record 0), and erases the reaminder of each track.
- Writes IPL records on track 0 (records 1 and 2).
- Writes volume label on track 0 (record 3) and provides space for additional records, if requested.
- Constructs and writes a volume table of contents (VTOC).
- Writes IPL program, if requested, on track 0 (2314, 2305, or 3330).

Note: Defective tracks are flagged and alternate tracks are assigned when the 3330 storage volumes are initialized at the factory. A MINIDASD job that initializes a 3330 will perform the "Quick DASDI" function, which reads alternate tracks, decrementing the total count of alternates by one whenever an alternate is found defective or assigned, writes a volume label and VTOC, and writes an IPLTEXT if requested. No surface analysis is performed and no home address or record 0 is written on the primary tracks. The BYPASS and FLAGTEST options are ignored.

Data is supplied to the program from either a specified CMS file or the card reader, and from the real or virtual console. The CMS file or card reader holds the control records for the program.

The MINIDASD program requires the following function-defining statements to initialize direct direct access volumes:

1. JOB Statement
2. MSG Statement
3. DADEF Statement
4. VLD Statement
5. VTOCD Statement
6. IPLTXT Statement (optional)
7. LASTCARD Statement (optional)
8. END Statement

These statements must appear in this sequence.

JOB Statement

The JOB statement indicates the beginning of a job.

[name]	JOB	[user information]

MSG Statement

The MSG statement defines an output device for operator messages. It follows the JOB statement and precedes a group of function-defining statements that are associated with one of the independent utility programs.

[name]	MSG	TODEV=xxxx
		TOADDR=cuu

TODEV=xxxx
specifies the device type of the message output device.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the message output device.

DADEF Statement

The DADEF statement defines the direct access volume to be initialized.

[name]	DADEF	TODEV=xxxx TOADDR=cuu [IPL=YES] [VOLID=serial] [VOLID=SCRATCH] [MODEL=n]	1
		[FLAGTEST=NO] [PASSES=n]	2
		[BYPASS=YES]	3

[1] Applicable to initialization with or without surface analysis.
[2] Applicable to initialization with surface analysis.
[3] Applicable to initialization without surface analysis.

TODEV=xxxx
specifies the device type of the direct access device.

TOADDR=cuu
specifies channel number (c) and unit number (uu) of the device.

IPL=YES
specifies that an IPL program is to be written on the minidisk. An IPL initialization program must be written on a device to be used for system residence.

If IPL is omitted, no IPL Program is written.

VOLID=serial
specifies the volume serial number of the minidisk to be initialized.

If "serial" matches the volume serial number found on the minidisk to be initialized, the operation proceeds. If it does not match, the operator is notified.

VOLID=SCRATCH
specifies that no volume serial number check is to be made.

MODEL=n
specifies a decimal model number (1 or 2). This parameter corresponds to the 2305-1 and 2305-2 respectively.

FLAGTEST=NO (applicable with surface analysis)
specifies that the program is not to check for previously flagged tracks before surface analysis is attempted on this device. (FLAGTEST=NO applies only to disk storage devices, and should be specified when the disk recording surface is initialized for the first time.)

Note: Since no check is ever made for previously flagged tracks on drum volumes, FLAGTEST=NO is not coded when these devices are initialized.

PASSES=n (applicable with surface analysis)
specifies that the program's defective-track checking feature is to make n number of passes (from 1 to 255) per track.

If PASSES is omitted, one pass is made per track.

BYPASS=YES

specifies that the program's defective-track checking feature is to be bypassed.

If BYPASS is omitted, tracks will be checked and those found defective will automatically be assigned alternates.

VLD Statement

[name]	VLD	NEWVOLID=serial
		{VOLPASS=1}
		{VOLPASS=0}
		[OWNERID=xxxxxxxxxx]
		[ADDLABEL=n]

NEWVOLID=serial

specifies a 1- to 6-character volume serial number.

VOLPASS=1

specifies that the volume security bit is to be set to 1.

VOLPASS=0

specifies that the volume security bit is to be set to 0.

If VOLPASS is omitted, the volume security bit will be set to 0.

OWNERID=xxxxxxxxxx

specifies a 1- to 10-character field that identifies the owner of the volume.

If OWNERID is omitted, no identification is given.

ADDLABEL=n

specifies a number between one and seven that indicates the total number of additional labels for which space is to be allocated.

If ADDLABEL is omitted, 0 is assumed.

VTOCD Statement

The VTOCD statement contains information for controlling the location of the volume table of contents.

[name]	VTOCD	STRTADR=nnnnn
		EXTENT=nnnn

STRTADR=nnnnn

specifies the 1- to 5-byte track address, relative to the beginning of the minidisk, at which the volume table of contents is to begin. The VTOC cannot occupy track 00 or any alternate track.

EXTENT=nnnn

specifies the length of the volume table of contents in tracks. The number of entries per track for each type of device is given below.

Device	*VTOC Entries/Track*
2301	63
2314	25
2311	16
2305-1	18
2305-2	34
3330	39

IPLTXT Statement

The IPLTXT statement separates service program control statements from IPL Program Text Statements. Itgn is required only when IPL text is included. The statement consists of the operation IPLTXT, followed with blanks.

When IPL text is included, END must start in column 2.

LASTCARD Statement

The LASTCARD statement is required only when a MINIDASD job or a series of stacked MINIDASD jobs is followed by other statements on the control statement input device. It must follow the last END statement applying to a MINIDASD job. It consists of the operation LASTCARD, followed with blanks.

END Statement

The END Statement denotes the end of job. It appears after the last function-defining statement.

[name]	END	[user information]

The example below shows the control statements that might be prepared to initialize the minidisk LIBRES for OS data set residence. Note that the label of the real volume, CPVOL1, cannot be used as the VOLID operand.

```
DADEF    TODEV=2314,TOADDR=231,
           VOLID=SCRATCH
VLD      NEWVOLID=LIBRES,
           OWNERID=OPERATIONS
VTOCD    STRTADR=10,EXTENT=5
END
```

The desired size of the minidisk is specified through the real or virtual console. The user types an appropriate three-digit value in response to the message 'ENTER 3 DIGIT NUMBER OF CYLINDERS TO BE INITIALIZED'. The value specified includes the space reserved for alternate track assignment; a user assigned n cylinders has n-1 of these initialized for his use, and the nth used for any alternate track assignment. If a complete volume is formatted, this value should be the maximum address that can be specified for that device type. The minimum size of a minidisk can be computed from the formula below. In it N represents the minimum number of cylinders, and K represents the number of recording heads that the device has. The SIZE-OF-VTOC value should be in tracks, and the result of the division should be rounded to the next highest integer.

N = 2 + (SIZE-OF-VTOC / K)

Assigning an Alternate Track

The second function of the MINIDASD program is used to (1) test a track and, if necessary, to assign an alternate, or (2) to bypass testing, and automatically assign an alternate.

Assigning an Alternate (With Testing): An alternate track will be assigned for a track specified for testing and found defective. If the defective track has had an alternate previously assigned, a new alternate is assigned. If the defective track is an unassigned alternate, it is flagged to prevent its future use. The alternate track address is made known to the operator.

If a track is tested and found to be "not defective," no alternate is assigned. The operator is notified by a message.

Assigning an Alternate (Without Testing): The program's defective track checking feature can be bypassed, and an alternate track can be assigned for any track, whether it is defective or not. If the specified track is an alternate, a new alternate is assigned. If the specified track is an unassigned alternate, it is flagged to prevent its future use.

Note: A GETALT function that specified a 3330 device will automatically assign an alternate track without testing, regardless of the FLAGTEST parameter setting.

GETALT Statement

Any number of alternate tracks on a minidisk can be assigned in a single job by including one GETALT statement for each track.

[name]	GETALT	TODEV=xxxx
		TOADDR=cuu
		TRACK=ccccchhhh
		VOLID=serial
		[FLAGTEST=NO]
		[PASSES=n]
		[BYPASS=YES]
		[MODEL=n]

TODEV=xxxx
specifies the device type of the direct access device.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the direct access device.

TRACK=ccccchhhh
specifies the address of the track for which an alternate is requested, where cccc is the cylinder number and hhhh is the head number.

VOLID=serial
specifies the volume serial number of the minidisk to which an alternate track is to be assigned. If "serial" matches the volume serial number found on this minidisk, the alternate track assignment proceeds. If it does not match, the operator is notified.

FLAGTEST=NO (used when testing before assigning an alternate)
specifies that the program will not check for a previously flagged track before a surface analysis is attempted on this track (disk storage devices only).

PASSES=n (used when testing before assigning an alternate)
specifies that the program's defective track checking feature is to make n number of passes (from 1 to 225) when performing a surface analysis on nn this track. If PASSES is omitted, one pass will be made on this track.

BYPASS=YES
specifies that the program's defective track checking feature is to be bypassed.

If BYPASS is omitted, the program assigns an alternate only if it finds that the specified track is defective.

MODEL=n
specifies a decimal model number (1 or 2). This parameter corresponds to the 2305-1 and 2305-2, respectively.

Note: A list of defective tracks is provided with new IBM Disk Storage volumes. This list should be referred to the first time the MINIDASD program is to be used. After initialization, the GETALT function can then be included in a MINIDASD job to assign an alternate track for each track on the list. Subsequent MINIDASD jobs will "remember" those defective tracks, unless the FLAGTEST=NO option is specified for those jobs.

Invoking MINIDASD Under CMS

Issuing a MINIDASD command invokes the MINIDASD program under CMS. The format of the MINIDASD command is:

MINIDASD	fname ftype

The filename and filetype of the CMS file that contains the input control records for the MINIDASD program must be specified as operands.

During execution, MINIDASD first checks for the existence of the specified filename filetype. It then interrogates the user as to the size of the minidisk to be created and verifies the value entered. It then prints out the input control records. If alternate tracks are assigned, it prints out the addresses of the bad tracks and of the assigned alternate tracks.

Invoking MINIDASD as a Standalone Program

To initiate MINIDASD as a standalone program, you need to prepare control statements just as you would for execution under CMS. In the real or virtual card reader (or some other IPL device) you should place the loader program deck, the MINIDASD deck, and the control cards.

When MINIDASD is loaded, it asks the operator for the address of the device that contains the control card file and for the device type. The operator response is as follows:

```
INPUT=devtype address
```

where devtype is the device type (such as 2540, 2400, or 3420), and address is the 3 hexadecimal digits that represent the real device address.

The MINIDASD execution proceeds just as it does under CMS. After examining the control statements, the program requests the operator to specify the desired size of the minidisk. It also lists the control records and the DASD assignment results as required.

Formatting Volumes—General Information

All direct access volumes used by the VM/370 system (for paging, spooling, system residence, directory, or temporary disk allocation) must be properly labeled, formatted, and allocated. The CP FORMAT service program (DMKFMT module) is used to prepare disks for use by CP. A CMS FORMAT program is also available and must be used for format CMS disks.

Volumes used only by virtual machines (minidisks or dedicated volumes) do not need to be formatted.

An object deck version of the CP FORMAT service program is a standalone program and can be IPLed from a virtual or real card reader in, respectively, a virtual or a real machine. (If run in a virtual machine, the virtual machine must have write access to the volume being formatted.) The program accepts control statements from the operator's console (commands) or from the IPL device (card reader).

Cylinders to be used by CP for system use (paging, spooling, etc.) must be preformatted with fixed length unblocked records of 4096 bytes.

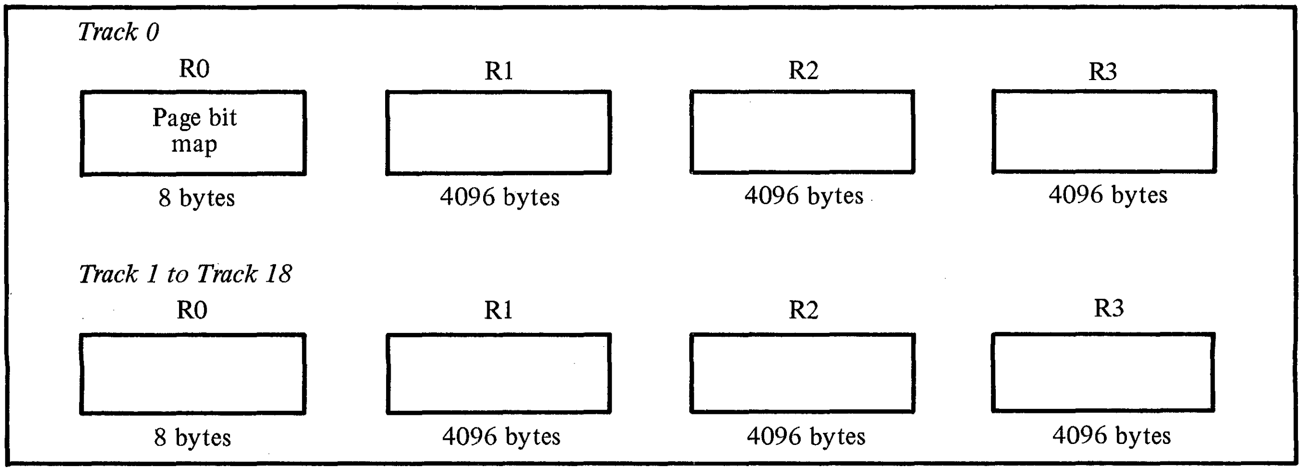
Device capacity when formatted for CP use is:

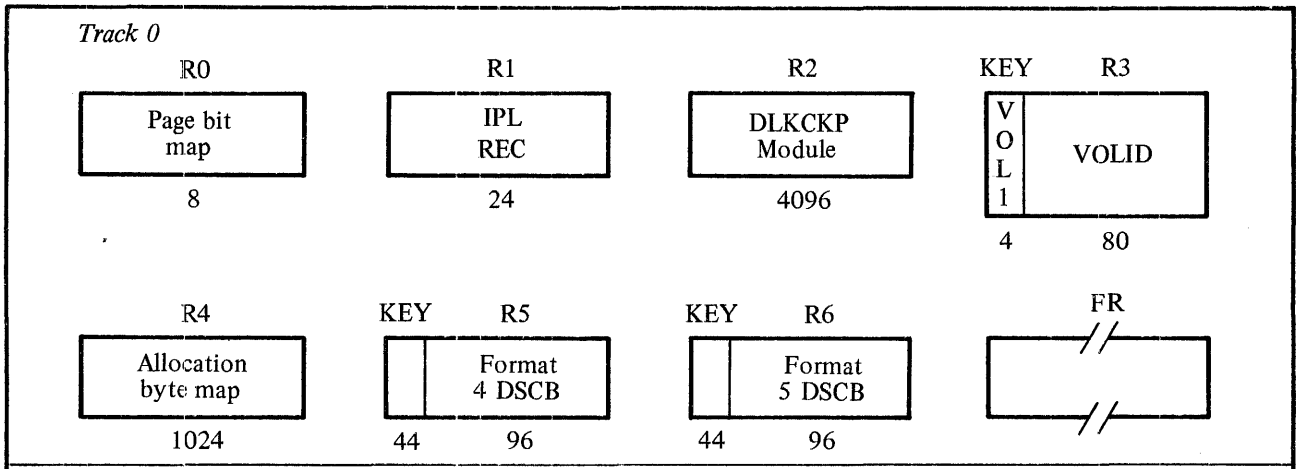
2314	32 records/cylinder (8 records/5 tracks)
3330	57 records/cylinder (3 records/track)
2305	24 records/cylinder (3 records/track)

An eight byte "page bit map" in record 0, track 0 of each cylinder is used by CP to determine which records are in use and which are available for allocation.

The format operation writes 4096 byte blocks on all cylinders being formatted. The service program does write-checking to verify that parts of the track are not defective. Bad surfaces are flagged as allocated (in use) in the page bit map and are not used. Alternate tracks are not assigned for CP-owned DASD devices.

For example, the 3330 track format for all formatted cylinders except cylinder 0 is as follows:





Note: Track 1 to Track 18 are the same as the non-zero cylinders.

Figure 18. 3330 Cylinder 0 Format

Cylinder 0 Format

All volumes containing space for CP use (paging, spooling, etc.) must have a properly formatted cylinder 0.

Cylinder 0 is formatted like other cylinders except that the space associated with the first three 4096-byte blocks is reserved for system use.

This area is then formatted as illustrated in Figure 18. The contents of each record in cylinder zero are as follows:

R0 Page bit map–indicates which pages on the cylinder are in use and available. The first three pages of cylinder 0 are always marked in use.

R1 IPL record–puts the system into wait state if storage volume is IPLed before CP nucleus is built.

R2 Checkpoint record– used by CP to save and retrieve information for a warm start.

R3 Volume label–Same as OS VOL1 label. On CP system residence volume, area in data record will mark the beginning of the system directory. A label is automatically written when cylinder 0 is formatted. The owner field of the label record contains "VM/370" if there is allocation data present in R4.

R4 Allocation Byte Map–each byte identifies a cylinder and specifies its usage (paging, spooling, directory, etc.). This map is filled in by the ALLOCATE function of the DMKFMT service program.

R5 Format 4 OS DSCB type label–for compatibility with OS.

R6 Format 5 OS DSCB type label–for compatibility with OS. Label indicates to OS that no space is available on this volume.

FR Is one or more filler records.

FORMAT/ALLOCATE Service Program (DMKFMT)

The FORMAT/ALLOCATE service program is used to format, allocate, and label direct-access volumes for paging, spooling, and CP file residence. This service program is executed as part of CP system generation procedures and may also be executed as a standalone program to:

1. Format direct-access volumes for CP use.
2. Allocate specific disk areas to particular functions or to CP use.
3. Write 6-character volume serial number labels.

Note: The FORMAT program should be used with care, since it destroys existing data (if any).

FORMAT control statements may be supplied in card form via a card reader, or may be entered at the system console. All error messages regarding improper specification of control statements are printed at the console. If the FORMAT service program encounters any unusable disk surfaces, it flags the associated track addresses, but it does not select any alternate tracks.

FORMAT Card Input

Control statements for card input are punched starting in column 1, and each field is separated from the adjacent field by a comma. Two commas in a row cause the insertion of a default value. Three commas in a row cause the insertion of two default values.

Note: The only default values permitted are those for cylinder volume specification.

The control card entries for the FORMAT program must be in the following order:

- FORMAT function

  FORMAT,devadr,devtype,volser,startcyladr,
  endcyladr

- ALLOCATE function

```
ALLOCATE,devadr,devtype,volser
TEMP,startcyladr,endcyladr
PERM,startcyladr,endcyladr
TDSK,startcyladr,endcyladr
DRCT,startcyladr,endcyladr
END
```

- LABEL functions

FORMAT, ALLOCATE, and LABEL are FORMAT control words and may be abbreviated to one letter, just as some other VM/370 commands may.

TEMP, PERM, TDSK, and DRCT are all functions of ALLOCATE. These cards can follow the ALLOCATE card in any sequence. There should be no overlap between cylinder allocation and desired functions. If, however, an ALLOCATE function does overlay the previous cylinder allotment, then the previous cylinder space allotment is truncated to the beginning of the next cylinder allotment. For example:

Disk Storage Allocation	*First Cylinder*	*Last Cylinder*
1st Entry TEMP	000	202
2nd Entry PERM	010	050
3rd Entry TDSK	040	050
4th Entry DRCT	000	004
5th Entry END		

The result of this disk allocation is:

Disk Storage Allocation	*First Cylinder*	*Last Cylinder*
DRCT	000	004
TEMP	005	009
PERM	010	039
TDSK	040	050
TEMP	051	202

Once an ALLOCATE card has been entered, all cards following it until an END card is encountered are assumed to be part of a single allocation. You cannot perform any of the FORMAT service functions on another disk volume until the END card is found.

The various operands that can be coded in a control card are as follows:

devadr
: a 3 digit hexadecimal number that identifies the address of the device that the FORMAT program is to act upon.

devtype
: 4 or 6 characters that define devices supported by the FORMAT program. Supported device types are 2314, 2319, 3330, and 2305-2.

volser
: 6 characters that represent the volume serial number.

startcyladr
: The physical starting address on the DASD on which the FORMAT function is to be performed. The start cylinder address is entered as decimal digits.

endcyladr
: The last physical address on the DASD on which the FORMAT function is to be performed. The end cylinder address is entered as decimal digits.

The following keywords should be entered, one per card, for each function desired:

TEMP — An ALLOCATE function that indicates that the following operands identify temporary storage space reserved for spooling or paging activity.

PERM — An ALLOCATE function that defines the area on a direct-access device where the permanent records of CP reside.

TDSK — An ALLOCATE function that defines the pooled space available for virtual machine users after they have logged into the VM/370 system.

DRCT — An ALLOCATE function that indicates that the following described cylinders are reserved for directory files.

Some examples of FORMAT control card input are as follows:

FORMAT:

```
FORMAT,232,3330,MYDISK,000,006
FORMAT,232,3330,MYDISK,,,
FORMAT,232,3330,MYDISK,,00
FORMAT,232,3330,MYDISK,001,,
```

ALLOCATE:

```
ALLOCATE,232,3330,MYDISK
TEMP,000,050
PERM,055,060
TDSK,100,108
DRCT,110,120
END
```

LABEL:

```
F,232,3330,MYDISK,LABEL
```

FORMAT Console Input

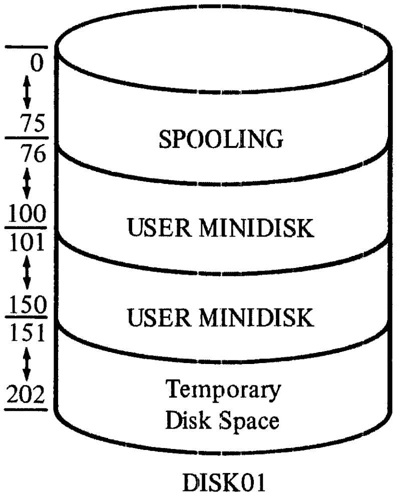
If you prefer, you can control the CP FORMAT service program at the system console, instead of preparing a deck of control cards. If the program finds no control cards at the card reader, it issues a prompting message to the system console. The proper response causes the prompting message for the next operand to appear until the FORMAT, ALLOCATE, or LABEL function is completely defined; then the function is executed.

After execution, the prompting process again until all DASD allocation requirements are fulfilled.

The sequence for console typewriter processing of the FORMAT service program (after making the operator's console ready), is as follows:

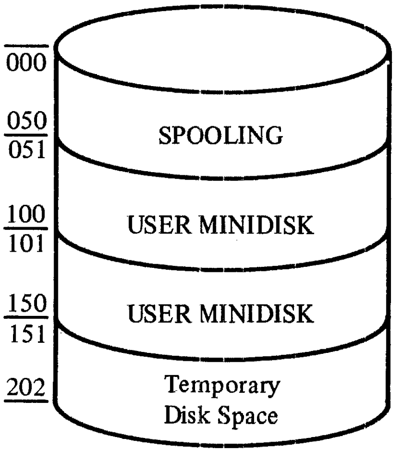
1. Load the card reader with a loader, followed by the FORMAT deck.
2. IPL the card reader.
3. Respond to the first inquiry printed at the system console.
4. Respond to other messages.

The ALLOCATE function is invoked by a series of cards (or commands) as shown in the following examples:

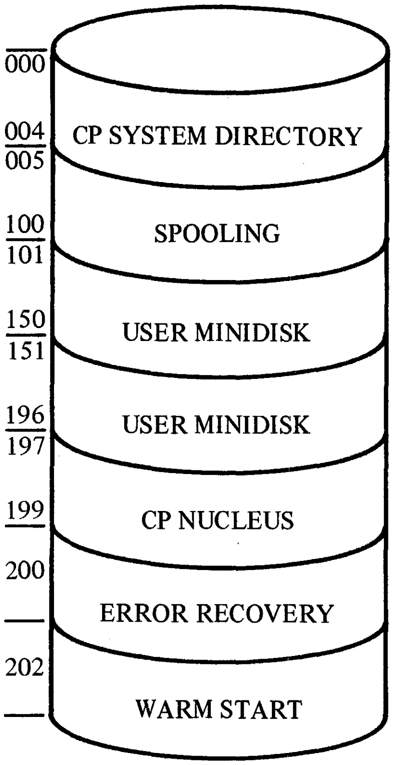


```
ALLOCATE 230,2314,
  DISK01
TEMP 000,075
PERM 076,150
TDSK 151,202
END
```

Note: Before invoking the ALLOCATE function, cylinder 0 must have been formatted and labeled as DISK01. The area associated with the first three 4096 byte blocks will not be used for spooling but will contain system information (page allocation map, label, etc.)



```
FORMAT 230,2314,
  DISK01,000,050
FORMAT 230,2314,
  DISK01,151,202
ALLOCATE
TEMP 000,050
PERM 051,150
TDSK 151,202
END
```



```
FORMAT 230,2314,
  DISK01,000,
FORMAT 230,2314,
  DISK01,197,
ALLOCATE
DRCT 000,004
TEMP 050,100
PERM 101,202
END
```

Note: To be used by CP, the volume serial numbers of these CP formatted volumes must:

1. Appear in the SYSOWN macro in the DMKSYS module. (This is explained further in "System Definition".) The CP system residence volume must appear in this list.

   or

2. Be attached to the SYSTEM by the CP system operator.

SYSTEM AND VIRTUAL MACHINE ACCOUNTING

The accounting data gathered by VM/370 can help in installation analysis of the operation of the overall system. A second important use of accounting data is, of course, in billing VM/370 users for the time and other system resources they use.

The accounting records are of two types: the virtual machine user records, and records for dedicated devices assigned to virtual machine users. These records are prepared as 80-character card images and sent to a punch file at various times. Output class C is reserved for accounting records.

Accounting Records for Virtual Machine Users

Accounting cards are punched and selected to pocket 2 of any class C card punch when a user logs out of the system or detaches a dedicated device or T-disk space. (If the real punch is a 2540, the accounting cards are put in pocket 3). These records should be kept for system accounting purposes. The information on the accounting card is as follows:

Column	*Contents*
1-8	Userid
9-16	Account number
17-28	Date and Time of Accounting mmddyyhhmmss
29-32	Number of seconds connected to VM/370 System
33-36	Milliseconds of CPU time used, including time for VM/370 supervisor functions.
37-40	Milliseconds of virtual CPU time used.
41-44	Number of page reads
45-48	Number of page writes
49-52	Number of virtual machine SIO instructions for non-spooled I/O.

Column	*Contents*
53-56	Number of spool cards to virtual punch
57-60	Number of spool lines to virtual printer
61-64	Number of spool cards from virtual reader
65-68	Reserved
79-80	Accounting card identification code*

* The accounting card identification code is one of the following:

01	Virtual machine accounting card
02	Dedicated device accounting card
03	Temporary disk space accounting card

Accounting Records for Dedicated Devices

Accounting cards are also punched and selected to pocket 2 when a previously dedicated device is released from a user by DETACH, LOGOUT, or releasing from DIAL. A dedicated device is any device assigned to a virtual machine for that machine's exclusive use. These include devices dedicated by the ATTACH command, those being assigned at login by directory entries, or by a user DIALing a system with virtual 2702 or 2703 lines. The information on the accounting card is as follows:

Column	*Contents*
1-8	Userid
9-16	Account number
17-28	Date and Time of Accounting mmddyyhhmmss
29-32	Number of seconds connected to VM/370 System
33-36	Device type code in decimal
37-38	Number of cylinders of temporary disk space used (if any)
39-78	Unused
79-80	Accounting card identification code (as previously described)

If a punch is started for 2 classes with NOSEP specified, accounting cards are not uniquely separated from data decks. If started with SEP specified, the operator is prompted when a user has a deck to be punched. He can thus remove any accounting cards before starting the punch. After data is through punching, accounting cards may be punched.

User Accounting Options

The user may insert his own accounting procedures in the accounting routines. Operator responsibilities in such cases should be defined by the installation making the additions. When designing such accounting procedures, the user should understand that:

1. The accounting routines are designed to be expanded by the user. The entry point provided in the accounting module for installation use is called DMKACON. Users desiring to perform additional accounting functions should modify the following copy files:

   ACCTON (account on)–for action at login time. This is provided as a null file. It can be expanded to provide additional functions at login time.

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

2. CP has no provision for writing the accounting records to disk.

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

GENERATING NAMED SYSTEMS

By taking advantage of the SAVESYS command, system resources are not committed to perform an IPL each time a saved system is requested by a user. Instead, the named system is referenced by the respective DMKSNT entry.

When adding, changing, or deleting, the DMKSNT module must be reassembled.

The procedure for generating a named system consists of two steps:

1. Configuring and assembling the NAMESYS macro (DMKSNT)

2. Loading the system to be saved and then invoking the SAVESYS command.

When allocating DASD space for named systems provide an extra page for information purposes, do not overlay this area with subsequent named systems.

Configuring the NAMESYS Macro (Module DMKSNT)

The NAMESYS macro is assembled by the installation system programmer and is used to describe the location of the saved system. Shared segments may be specified, but they must consist of reenterable code, with no alteration of its storage space permitted.

Input to the NAMESYS macro is specified in the following format:

```
label NAMESYS       SYSSIZE=nnnK,SYSNAME=cccccc,
                       VSYSRES=cccccc,
                    VSYSADR=ccu,SYSVOL=cccccc,
                       SYSCYL=nnn,
                    SYSSTRT=(cc,p),SYSPGCT=nn,
                    SYSPGNM=(nn,nn,nn-nn, . . .),
                    SYSHRSG=(n,n, . . .)
```

where:

label Is any desired user label.

SYSSIZE Is the minimum storage size needed to operate the saved system.

SYSNAME Is the name given the system to be used for identification by SAVESYS and IPL.

VSYSRES Is the volume serial number of the DASD volume containing the system to be saved.

VSYSADR Is the virtual address of the DASD volume containing the system.

SYSVOL Is the volume serial number of the DASD designated to receive the saved system. This must be a CP-owned volume.

SYSCYL Is the cylinder address of the minidisk for the system to be saved.

SYSSTRT Designates the starting cylinder and page address on SYSVOL at which this named system is to be saved. During the SAVESYS and IPL processing, this will be used to make up the "cylinder page and device" address for the DASD operations. These numbers are to be specified in decimal.

SYSPGCT Is the total number of pages to be saved.

SYSPGNM Are the numbers of the pages to be saved. Specifications may be done as groups of pages or as single pages. For example: if pages 0, 4, and 10 through 13 are to be saved, use the format: SYSPGNM (0,4,10-13).

SYSHRSG Are the segment numbers designated as shared. The pages in these segments will be set up at IPL time to be used by any user loading by this name.

For example:

```
DMKSNTBL CSECT
FSTNAME NAMESYS     SYSSIZE=256K,SYSNAME=
                       CMS,VSYSRES=CPDSK1,
                    SYSVADDR=190,SYSCYL=
                       100,SYSVOL=CPDSK2,
                    SYSSTRT=(400,1),SYSPGCT=
                       10,
                    SYSPGNM=(0-5,10-13),
                       SYSHRSG=(1,2)
```

Using the SAVESYS Command

The system to be saved must first be loaded by device address in the traditional manner. The point of capture of the system to be saved should be determined by the installation system programmer. Once loaded, the command SAVESYS NAME must be issued, where NAME corresponds to the identification of the saved system.

VM/370 SYSTEM OPERATION

System Initialization

To initiate the operation of the VM/370 System, the CP system residence volume must be loaded.

System initialization is simplified if:

1. All VM/370 resident volumes (specified in the SYSOWN list for paging and spooling) are mounted and ready at IPL time.
2. Volumes containing user minidisks (such as the CMS system residence volume) are mounted and ready at IPL.

Volumes can be attached to the system during operation, although this may cause some inconvenience to minidisk users.

Once loaded, CP will read the labels of all available DASD devices, and calculate the real machine's storage size. If the configuration differs from that specified during system generation (SYSOWN volumes not mounted, storage size not equal to SYSCOR), a message is printed at the system console and operation continues.

The primary system operator of the VM/370 system is assumed to be the first user with privilege class A to log in to VM/370.

The VM/370 system attempts to automatically LOGON the system operator with a predefined userid as specified in the SYSOPR macro. If this is not possible (USERID not in the VM/370 user directory or not class A), VM/370 will perform a prompting operation at the main system console and an operator with privilege class A must LOGON.

After a successful operator LOGON, VM/370 prompts him for the type of system start desired (whether this is a WARM START or not).

At this point the system is ready for normal use. The operator must now:

- Establish a message of the day to be issued to users at the time they LOGON.
- Enable communications lines to permit users to LOGON.

Normal Operation

Certain functions can be performed during the normal operation of the VM/370 system.

Assignment of special privileges to logged in users (FAVORED, RESERVE, PRIORITY, LOCK and UNLOCK commands): These functions can be performed only by a primary system operator with privilege class A.

Routine handling of spooled input and output: Card decks for users must be fed in the real card reader, printed and punched output properly distributed, and the unit record equipment and spool data files controlled. The control functions for unit record equipment and spool data files can be performed only by a class D operator.

Attaching and detaching of user and system volumes: Devices used by virtual machines in dedicated mode must be attached and detached as appropriate. Resource control over the real System/370 computing system can be performed by an operator with class B.

Operators with these privilege classes must be logged in to perform these functions. Below is a directory entry for an operator to perform all of these functions:

```
USER OPERATOR OPASS 8K 1M ABD
```

The system and spooling operators do not require virtual devices or options.

Multiple virtual machines for operators may be set up, each with all or some of the associated privilege classes. The primary system operator *must* have class A and class B codes assigned in the directory to properly initiate VM/370 operation.

System Termination

The VM/370 system will shut down in one of two ways:

1. By an operator-initiated shutdown.
2. By an abnormal shutdown when the system may be forced to abnormally terminate operation.

The normal shutdown is initiated by a class A operator entering the command

```
SHUTDOWN
```

from the operator's console. The SHUTDOWN command has immediate effect. Before issuing this command, a message or warning should be sent to all users, and unit record I/O activity should be brought to an orderly termination by a class D operator. Accounting information and spool file data is saved and can be retrieved when the system is restarted.

The dump unit for VM/370 system failures is specified in the SYSDUMP macro during VM/370 system generation, but can be changed by class A or class B system operator.

If the dump unit is set to disk (by default or via the SET DUMP AUTO command) at the time of system failure, the system will dump all or parts of real storage (also specified at system generation, or by a class A or B operator) to the specified disk and automatically restart the VM/370 system.

When automatically restarted, the system will preserve all accounting information and spool file data on disk, perform an automatic login of the primary system operator, restore the system LOGMSG, and continue system operation. It is not necessary to re-enable the lines, since dump and re-IPL do not disturb their previous condition. A message is sent to the terminal users that they must log in again.

If, at the time of the system failure, the dump unit is set to a printer or tape, the system will write the dump on the specified unit, preserve the spooling and accounting data, and stop. The operator must then re-IPL the VM/370 system as for normal system start up, specifying a WARM START to preserve the accounting and spool file data. Communication lines must be re-enabled in this case to permit users to log in again.

System ABEND Dumps

There are conditions within the CP program that may force an abnormal ending condition (ABEND) and cause the dumping of system registers and storage. The device that receives these records can be a tape, printer, or disk device.

Dumping operations are caused by any program interrupt or system restart condition. These interrupt conditions will cause routines to gather data from registers and storage and place this data on a previously-defined device. The quantity of data to be dumped is also defined by the system CP command SET DUMP.

For example:

```
SET DUMP AUTO
SET DUMP raddr
SET DUMP raddr ALL
SET DUMP AUTO ALL
```

SET DUMP AUTO places the VM/370 system dump on a preselected file device. (The disk dump area is automatically selected at system initialization time.) The device type and address can be verified by entering the QUERY DUMP command.

SET DUMP raddr is used when the dump device is to be a high speed printer or tape device. The real hexadecimal 1 to 3 digit device address is substituted for the raddr operand.

The ALL operand used in conjunction with SET DUMP AUTO or SET DUMP raddr causes all of storage to be dumped to the DUMP device. If the ALL option is not specified, then the system defaults to dumping only those areas that pertain to CP, and not those areas that pertain to virtual machine operations.

If the system operator elects to dump to a disk file, an additional operation is necessary to transform these records into readable output for programmers or system analysts. This may be accomplished with the CMS VDUMP function.

If the records are dumped onto a tape drive, other CMS command options must be invoked for printout.

Only ABEND dumps that are a result of using the SET DUMP AUTO command are spooled as a special virtual card reader file. This card reader file is assigned during system generation to a specific virtual machine user via the SYSOPR macro. The CMS VDUMP function is used to print these CP ABEND dumps.

The VDUMP function invokes two CMS programs. The first one creates a CMS file from the CP disk dump data, and the second one prints the dump from the CMS file.

CONCURRENT MAINTENANCE FACILITIES

At system generation time, two additional virtual machines should be created beyond those needed by normal users. One is for hardware maintenance, and one is for software maintenance. The IBM FE Program Systems Representative should be consulted when the configurations for these virtual machines are being determined.

The hardware maintenance covers two areas:

1. The CPU, which must be maintained in a dedicated environment, since there is no method currently available which allows concurrent maintenance of the CPU, main storage, or channels while running problem programs.
2. Input/Output equipment, which can be maintained using online tests (OLT) under OLTSEP. The OLTSEP program can be run in its own virtual machine.

If desired, 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 maintenance, a virtual machine must be defined in the VM/370 system directory for the IBM Service Representative.

The virtual machine should have enough virtual storage defined to run OLTSEP. Normally, the Service Representative will require that the device being tested be dedicated to his virtual machine (accomplished dynamically via the ATTACH console function).

The virtual machine for hardware maintenance should be the minimum configuration required to run online tests, and provide access to CMS with a read/write minidisk. User 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 pack. An EREP program will be provided to run under CMS and allow editing and printing of all VM/370-recorded machine check and channel check errors.

For example, a virtual machine for hardware maintenance would be configured as follows:

```
USER SRMAIT PASSWORD 256K 1M FG
  ACCOUNT NUMBER BIN5
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    LINK CMSSYS 190 190 RR
    MDISK 191 2314 51 10 UDISK1 WR RPASS
             WPASS
```

Virtual device 191 is a CMS A-disk to contain programs required by the Service Representative (EREP, etc.). Input/output units to be tested would normally be attached to the virtual machine for the duration of the test. Privilege class F is provided to allow the Service Representative to specify an intensive error recording mode for a particular device being tested, and to examine or clear the system error recording area.

The virtual machine for software maintenance should have the minimum configuration necessary to recreate (virtually) problems which occur on the real machine. The ECMODE option must be specified in the directory OPTION card for this machine. User class G should be assigned to this machine. If it is desired to allow the ability to examine the real storage addresses, user class E should be additionally assigned to this user's directory record.

Appendix A: Migration From CP-67/CMS To VM/370

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 will not run on a System/370, and the VM/370 system will 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.

Note: To avoid possible ambiguity, minimize confusion, and reduce the number of times that the full names of the components of CP-67/CMS and VM/370 have to be spelled out, the following code names will be used throughout this appendix:

- CP-67 The Control Program component of CP-67/CMS
- CMS-67 The *Cambridge* Monitor System component of CP-67/CMS
- CP The Control Program component of VM/370
- CMS The *Conversational* Monitor System component of VM/370

The Conversational Monitor System (CMS) functionally extends, but is substantially different from the Cambridge Monitor System (CMS-67). To a lesser degree, the same is true of the control program components, although the differences are less extensive. Some commands have completely unchanged, certain command names have changed, some individual commands, operands, and options have been dropped, and the functions moved to other commands. Most commands, however, have additional operands or options which offer more function and more precise control. Important differences between the old and new components are summarized separately following.

Major CMS Differences

1. Although the CMS disk file formats are unchanged, 2311 disks are not supported for CP or CMS files.
2. Filename and filetype naming conventions have been changed. All characters of each must be alphabetic (A-Z, a-z), numeric (0-9), or one of the following three special characters: $, #, @. Other special characters are not permitted.
3. Because the user may now have up to 10 disks, the logical directory identifications (filemode letters) have been changed to reflect a more "natural" 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, the primary disk the A-disk, and the temporary or dynamic disk being the D-disk.
4. Because of significantly expanded capabilities, the CMS nucleus no longer fits within 12000 (hex) bytes. Consequently, all existing CMS-67 MODULE files have to be recreated in the new system, and possibly recoded and re-assembled as well.
5. CMS will not function on a real CPU without CP.
6. Because many NUCON fields have been changed or rearranged, many user programs which refer to NUCON have to be reassembled with the new CMS macro libraries.
7. Modules which refer to fields containing sizes, limits, and quantities within the CMS nucleus might have to be reassembled and then regenerated.
8. TXTLIB files must be regenerated because the dictionary format has been changed.

The following CMS-67 commands are *not* included in the Conversational Monitor System of VM/370.

CMS-67 Command	*Function Now Performed by*
$	RUN
ALTER	RENAME
BLIP (and VSET BLIP)	SET BLIP
BRUIN	[NONE]

CMS-67 Command	*Function Now Performed by*
CEDIT	EDIT
CHARDEF	SET INPUT/OUTPUT
CLROVER	SVCTRACE
CNVT26	COPYFILE
COMBINE	COPYFILE
CPFUNCTN	CP
CVTFV	COPYFILE
DUMPF	PRINT/TYPE
ECHO	CP's ECHO
FORTRAN	FORTGI, FORTHX, GOFORT*
IPL	CP's IPL
KE	[NONE]
LISTF	LISTFILE
LOGIN	ACCESS
LOGOUT	CP's LOGOUT
MAPPRT	[CMS file with filetype NUCMAP]
OFFLINE	READCARD/PRINT/PUNCH
OSTAPE	TAPPDS
PLI	PLIOPT*
PRINTF	TYPE
REUSE	INCLUDE (RESET)
SCRIPT	[NONE]
SETERR	SVCTRACE
SETOVER	SVCTRACE
SNOBOL	[NONE]
TAPEIO	TAPE
TAPRINT	MOVEFILE
TPCOPY	MOVEFILE
WRTAPE	MOVEFILE
USE	INCLUDE
VSET	SET

* Commands used to invoke IBM Program Products, which must be ordered separately.

CMS-67 operands and options NOT included in the Conversational Monitor System (in most cases, new operands and options perform similar functions)

ALTER (now called RENAME)–NOUP, "*" in new filename, filetype, or filemode.

ASSEMBLE *multiple* filenames (. . . fnN), LTAPn, LDISK, DIAG. In addition, the meaning of DECK has been changed to "route the output object (text) file to the spooled card punch," and a new option LOAD now means what DECK used to.

COMPARE NOSEQ

ERASE "*" not allowed for filemode when filename and filetype are also specified as an asterisk.

FILEDEF (options totally different from CMS-67 command) CON, DSK, DSK-mm, PRT, PUN, RDR, BAT, CASE LOWER, CASE UPPER, BLKSIZE, MODEnn.

FORMAT C, SYS

GENMOD entry1 and entry2 without TO and FROM keywords and outside parenthesis. Filemode of "P2" not allowed.

GLOBAL PRINT.

LISTF	(now LISTFILE)–SORT, ITEM, NAME, TYPE, MODE, REC, YEAR, TIME.
LOAD	SLCxxxxx, SLC12000, SINV, PINV, SREP, PREP, SAVTO, XEQ, SLIBE. In addition, LIBE has a different meaning.
LOADMOD	filemode *without* filetype.
MACLIB	PRINT, LIST.
PRINTF	(renamed TYPE)–"*" is not permitted for filename, filetype. "n3" is not allowed.
START	(NO)
SYNONYM	P, PUSER
TAPE	LOAD n, SCAN n, SKIP n, REWIND, SLOAD n, WRITEOF, TAPn, TAP2.
TXTLIB	GENERATE, DELETE, PRINT, LIST.
UPDATE	P.

COMPATIBILITY OF VM/370 WITH CP-67/CMS

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 3.1 and not under VM/370 indicated that the syntax and function are identical.

Status:

A–CP/67	Syntax accepted in VM/370
R–CP/67	Syntax not accepted–function supported by another command format.
N–CP/67	Syntax not accepted–function not supported in VM/370.

Status	*CP/67 Command*	*VM/370 Equivalent*
R	ACNT	ACNT ALL
A	ATTACH cuu TO $\begin{Bmatrix}\text{SYSTEM}\\ \text{userid}\end{Bmatrix}$ AS $\begin{Bmatrix}\text{volid}\\ \text{vaddr}\end{Bmatrix}$	
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 $\begin{Bmatrix}\text{SYSTEM}\\ \text{userid}\end{Bmatrix}$
A	DIAL system	
N	DIRECT LOCK	
N	DIRECT UNLOCK	
A	DISABLE line	
A	DISABLE ALL	

Status	*CP/67 Command*	*VM/370 Equivalent*
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]	
R	DMCP T[–hexloc2]	DMCP T0 [–hexloc2 –END]
A	DMCP Thexloc1 [–hexloc2]	
A	DRAIN	
N	D_U_M_P	
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]	

Status	*CP/67 Command*	*VM/370 Equivalent*
R	LINK userid xxx yyy $\begin{bmatrix}\text{W}\\\text{R}\end{bmatrix}$ (NOPASS)	LINK userid xxx yyy $\begin{bmatrix}\text{W}\\\text{R}\end{bmatrix}$
A	LOCK userid fpage lpage	
A	LOGIN userid	
R	LOGIN userid xxx	LOGIN userid MASK
A	LOGOUT	
R	LOGOUT xxx	LOGOUT 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	
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 $\begin{Bmatrix}\text{ON}\\\text{OFF}\end{Bmatrix}$	TERMINAL APL $\begin{Bmatrix}\text{ON}\\\text{OFF}\end{Bmatrix}$

Status	*CP/67 Command*	*VM/370 Equivalent*
R	SET ATTN $\left\{\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}\right\}$	TERMINAL ATTN $\left\{\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}\right\}$
R	SET CARDSAVE $\left\{\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}\right\}$	SPOOL READER $\left\{\begin{matrix}\text{HOLD}\\ \text{NOHOLD}\end{matrix}\right\}$
A	SET DUMP xxx	
A	SET $\left\{\begin{matrix}\text{LINEDIT}\\ \text{RUN}\end{matrix}\right\}$ $\left\{\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}\right\}$	
A	SET LOGMSG	
A	SET LOGMSG NULL	
A	SET LOGMSG n	
N	SET MAX	
A	SET MSG $\left\{\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}\right\}$	
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 $\left\{\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}\right\}$	
A	SHUTDOWN	
A	SLEEP	
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

Status	*CP/67 Command*	*VM/370 Equivalent*
A	UNLOCK userid fpage lpage	
A	WNG userid text	
A	WNG ALL text	
R	XFER xxx $\left\{\begin{array}{l}\text{TO userid}\\\text{OFF}\end{array}\right\}$	SPOOL xxx $\left\{\begin{array}{l}\text{TO userid}\\\text{OFF}\end{array}\right\}$

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 CMS Version 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 have been effected; but, in some cases, old keywords and functions will be supported.
3. Command Name Alterations: where commands have undergone name changes, an optional SYNONYM file may be included during nucleus system generation.
4. Keyword Changes: some keywords within a command have been modified, deleted or added.
5. Major Modifications: improvements to commands and system functions have 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 the user may now have up to 10 disks, the logical directory identifications (filemode letters) have been changed to reflect a more "natural" 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/NODECK has a new meaning, that is, route the object (TEXT) file to the spooled card punch; the LOAD/NOLOAD option now invokes the function formerly performed by DECK/NODECK, that is, the writing of the TEXT file to 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 CP.
   - All TXTLIB files must be recreated using the TXTLIB command.
   - Because many fields have been changed in the CMS nucleus or rearranged, many user programs which refer to these fields have to be reassembled with the new CMS macro libraries.
   - Modules which refer to fields containing sizes, limits, and quantities within the CMS nucleus might have to be reassembled and then regenerated.
   - SYSLIB MACLIB has been renamed to CMSLIB MACLIB.

- All EXEC files should be checked for command name and operand changes, filemode usage, etc., and changed to conform to VM/370 CMS. Major changes are:

&TYPEOUT	to &CONTROL
&PRINT	to &TYPE
&INDEX0	to &RETCODE

- 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 V1.0 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 a parameter, 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
```

Appendix B: 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 the publication *IBM Virtual Machine Facility/370: Introduction,* Order No. GC20-1800.

Appendix C: CP Command Privilege Classes

The CP commands are divided into eight groups, each being represented by a privilege class which indicates the source from which the commands will be accepted. Each user is assigned, as part of his entry in the user directory, one or more privilege classes. Figure 19 shows the function of each class and Figure 20 shows the commands which are accepted from users in each class.

Class	*User*
A	*System Operator:* The function of the class A user is to control the VM/370 system. class A is assigned to the user who is controlling the CPU console typewriter at VM/370 IPL time. *Note:* The class A system operator who is automatically logged in at CP initialization is designated as the *Primary System Operator.*
B	*System Resource Operator:* The function of the class B user is to control the real resources of the System/370 computer.
C	*System Programmer:* The function of the class C user is to maintain and modify certain functions of the VM/370 system.
D	*Spooling Operator:* The function of the class D user is to control the system unit record equipment and the spool data files.
E	*System Analyst:* The function of the class E user is to examine certain data in the VM/370 storage area.
F	*Service Representative:* The function of the class F user is to obtain and examine in detail certain data about input/output devices connected to the System/370 computer.
G	*General User:* The function of the class G user is to control functions associated with the execution of his virtual machine.
H	Reserved for future use.

Figure 19. CP Privilege Class Description

Class	*Command*
ALL	DIAL, DISCONN, LOGIN, LOGOUT, MSG, QUERY LOGMSG, QUERY NAMES, QUERY userid, QUERY USERS, SLEEP, *
A	ACNT, DISABLE, ENABLE, FORCE, HALT, LOCK, QUERY PAGING, QUERY PRIORITY, SET FAVORED, SET PAGING, SET PRIORITY, SET RESERVE, SHUTDOWN, UNLOCK, WNG
B	ATTACH, ATTACH CHANNEL, DETACH, DETACH CHANNEL, DISABLE, ENABLE, MSG ALL, QUERY DASD, QUERY DUMP, QUERY LINES, QUERY raddr, QUERY STORAGE, QUERY SYSTEM, QUERY TAPE, QUERY UR, SET DUMP, SET LOGMSG, VARY, WNG
C	STCP
D	BACKSPAC, CHANGE, DRAIN, FLUSH, FREE, HOLD, LOADBUF ORDER, PURGE, QUERY FILES, QUERY HOLD, QUERY PRINTER, QUERY PUNCH, QUERY READER, REPEAT, SPACE, START, TRANSFER
E	DCP, DMCP, LOCATE, QUERY PAGING, QUERY PRIORITY, SAVESYS
F	SET RECORD
G	ADSTOP, BEGIN, CHANGE, CLOSE, COUPLE, DEFINE, DETACH, DISPLAY, DUMP, ECHO, EXTERNAL, IPL, LINK, NOTREADY, ORDER, PURGE, QUERY FILES, QUERY LINKS, QUERY PRINTER, QUERY PUNCH, QUERY READER, QUERY SET, QUERY TERMINAL, QUERY TIME, QUERY VIRTUAL, READY, RESET, REWIND, SET ACNT, SET EMSG, SET LINEDIT, SET MSG, SET NOTRAN, SET RUN, SET TIMER, SET WNG, SPOOL, STORE, SYSTEM, TERMINAL, TRACE, TRANSFER, LOADVFCB

Figure 20. Commands Accepted From Each User Class

Appendix D: CP Commands

This section contains descriptions of the commands acceptable in the control program environment. Figure 21 presents an alphabetical list of the commands, the privilege classes which may execute the command, and a brief statement about the use of each command.

Command	*Class*	*Usage*
*	ALL	Annotate the console sheet
ACNT	A	Create accounting records for logged on users and reset accounting data
ADSTOP	G	Halt execution at a specific virtual machine instruction address
ATTACH	B	Attach a real device to a virtual machine Attach a DASD device for CP control Dedicate all devices on a particular channel to a virtual machine
BACKSPAC	D	Restart or reposition the output of a spooling device
BEGIN	G	Continue or resume execution of the virtual machine at either a specific storage location or at the address in the current PSW
CHANGE	D,G	Alter one or more external attributes of a closed spool file
CLOSE	G	Terminate spooling operations on a virtual card reader, punch, or printer
COUPLE	G	Connect channel-to-channel adapters
DCP	E	Display real storage at terminal
DEFINE	G	Reconfigure a virtual machine

Figure 21. CP Command Summary (Part 1 of 4)

Command	*Class*	*Usage*
DETACH	B, G	Disconnect a real device from a virtual machine Detach a DASD device from CP Detach a channel from a specific user Detach a virtual device from a virtual machine
DIAL	ALL	Connect a terminal to virtual machine virtual communication line
DISABLE	A, B	Disable communication lines
DISCONN	ALL	Disconnect user's terminal from a virtual machine
DISPLAY	G	Display virtual storage on user's terminal
DMCP	E	Dump real storage on user's virtual printer
DRAIN	D	Halt operations of specified spool devices upon completion of current operation
DUMP	G	Print on virtual printer: virtual PSW, general registers, floating-point registers, storage keys, and contents of specified virtual storage locations
ECHO	G	Test terminal hardware
ENABLE	A, B	Enable communication lines
EXTERNAL	G	Simulate an external interrupt for a virtual machine and return control to that machine
FLUSH	D	Cancel output on a specific real unit record device
FORCE	A	Cause logout of a specific user
FREE	D	Remove spool HOLD status

Figure 21. CP Command Summary (Part 2 of 4)

Command	*Class*	*Usage*
HALT	A	Terminate the active channel program on specified real device
HOLD	D	Defer real spooled output of a particular user
IPL	G	Simulate IPL for a virtual machine
LINK	G	Provide access to a specific DASD device by a virtual machine
LOADBUF	D	Load real USC or FCB printer buffer
LOADVFCB	G	Load virtual forms control buffer
LOCATE	E	Find control blocks
LOCK	A	Bring virtual pages into real storage and lock them thus excluding them from future paging
LOGIN	ALL	Provide access to CP
LOGOUT	ALL	Disable access to CP
MSG	B, ALL	Transmit messages from user to user
NOTREADY	G	Simulate 'not ready' for a device to a virtual machine
ORDER	D, G	Rearrange closed spool files in a specific order
PURGE	D, G	Remove closed spool file from system
QUERY	A, B, D, E, G, ALL	Request information about machine configuration and system status
READY	G	Simulates device end interrupt to specified virtual address
REPEAT	D	Repeat printing or punching of current real spool output file
RESET	G	Clear and reset all pending interrupts for a specified virtual device and reset all error conditions

Figure 21. CP Command Summary (Part 3 of 4)

Command	*Class*	*Usage*
REWIND	G	Perform operation equivalent to rewinding and readying a tape
SAVESYS	E	Save virtual machine memory space, registers, and PSW
SET	A, B, F, G	Operator–establish system parameters User–control various functions within the virtual system
SHUTDOWN	A	Terminate all CP functions and checkpoint CP system for warm start
SLEEP	ALL	Place virtual machine's terminal in dormant state with keyboard locked
SPACE	D	Force single spacing on printer
SPOOL	G	Alter spooling control option
START	D	Start spooling device after draining or changing output classes
STCP	C	Allow alteration of real storage
STORE	G	Alter specified virtual storage and registers
SYSTEM	G	Simulates RESET, CLEAR STORAGE and RESTART buttons on console
TERMINAL	G	Control terminal I/O processing
TRACE	G	Trace specified virtual machine activity at the terminal, spooled printer or both
TRANSFER	D, G	Direct spooled files to specified user's card reader
UNLOCK	A	Unlock previously locked page frames
VARY	B	Mark a device unavailable or available to CP
WNG	A, B	Transmit a high priority message to a specified user or to all users

Figure 21. CP Command Summary (Part 4 of 4)

PROGRAMMING SYSTEMS

Supported Languages and Libraries

A VM/370 System Assembler is distributed as part of VM/370, and is required for installation and maintenance. All necessary macros are provided in CMS libraries.

BASIC is distributed as a part of the CMS component of VM/370. This BASIC language facility consists of the CALL-OS BASIC (Version 1.1) Compiler and Execution package adapted for use with CMS. This BASIC facility will receive Class A maintenance by the VM/370 Central Programming Service.

The following program products may also be ordered for use with CMS:

Program Products	
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 FORTRAN IV (G1)	5734-FO2
OS FORTRAN IV Library (Mod I)	5734-LM1
OS Code and Go FORTRAN	5734-FO1
OS FORTRAN IV (H Extended)	5734-FO3
OS FORTRAN IV Library (Mod II)	5734-LM3
FORTRAN Interactive Debug	5734-FO5
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

Further details on languages supported appear in the publication *IBM Virtual Machine Facility/370: Introduction,* Order No. GC20-1800.

INTEGRATED EMULATORS UNDER VM/370

Emulator-dependent programs (except for DOS emulation under OS or OS/VS) which run on a particular System/370 equipped with the appropriate compatibility features can be run on that System/370 in DOS or OS virtual machines under VM/370.

The chart shows, by System/370 model number, which integrated emulators can run under VM/370 and the compatibility feature numbers (#xxxx) that are required:

		1401			
		1440			709
	1401	1460			7090
	1440	1410	7070		7094
S/370	1460	7010	7074	7080	7094II
135	#4457				
145	#4457	#4458			
158, 155 II		#3950	#7117		
168, 165 II			#7117	#7118	#7119

No changes are required to the emulators, DOS, OS, or to VM/370 itself to allow emulator-dependent programs to run in virtual machines.

Appendix F: CMS Commands

This section contains descriptions of the commands acceptable in the CMS environment. Figure 22 contains an alphabetical list of the commands and functions performed by each of the commands.

Command	*Usage*
ACCESS	define direct access space to a CMS virtual machine, and relate the disk space to a logical directory.
ASSEMBLE	assemble Assembler Language source code
BASIC	compile and execute VM/370 BASIC programs
*COBOL	compile ANS Version 4 COBOL source code
COMPARE	compare all or part of records in two existing files
*CONVERT	convert free form FORTRAN statements to fixed form
COPYFILE	copy files according to specifications
CP	enter CP console functions from CMS environment
DEBUG	enter DEBUG subenvironment
DISK	perform disk-to-card and card-to-disk operations for CMS data sets
DDR	perform disk-to-tape and tape-to-disk operations for CMS data sets
EDIT	enter EDIT subenvironment
ERASE	delete files from user disks

* These commands invoke processors which are IBM Program Products. They are available from IBM for a license fee.

Figure 22. CMS Command Summary (Part 1 of 3)

Command	*Usage*
EXEC	process special procedures made up of frequently-used sequences of commands
FILEDEF	provide simulation of OS JCL data definition (DD) statements
FORMAT	prepare disks in CMS 800-byte block format
*FORTGI	compile FORTRAN source code using GI compiler
*FORTHX	compile FORTRAN source code using H-extended compiler
GENDIRT	create auxiliary module directories
GENMOD	generate absolute non-relocatable file (MODULE files)
GLOBAL	define CMS libraries to be searched for macros and subroutines
*GOFORT	compile FORTRAN source code and execute program just compiled using Code and Go compiler
INCLUDE	bring additional TEXT files into storage
LISTFILE	list information about user CMS files
LOAD	bring TEXT files into storage and establish linkages
LOADMOD	bring a single MODULE file into storage
MACLIB	perform maintenance on macro libraries
MINIDASD	to initialize DOS or OS minidisks
MODMAP	type load map of a MODULE file
MOVEFILE	move data from one device to another device of the same or different type
*PLIOPT	compile PL/I source code (using optimizing compiler)

* These commands invoke processors which are IBM Program Products. They are available from IBM for a license fee.

Figure 22. CMS Command Summary (Part 2 of 3)

Command	*Usage*
PRINT	spool a specified file to the printer
PUNCH	spool a specified file to the punch
QUERY	request information about the virtual machine
READCARD	read data from spooled card input device
RELEASE	make a disk and its directory inaccessible to a virtual machine
RENAME	change the name of a CMS file or files
RUN	initiate series of functions to be performed on a file
SET	establish, set, or reset virtual machine characteristics
SORT	arrange a specified file in ascending order according to specified field in the data record
START	begin execution of programs previously loaded
STATE	verify the existence of a file
SVCTRACE	record information about supervisor calls
SYNONYM	specify alternate names by which certain commands may be invoked
TAPE	performs tape-to-disk and disk-to-tape operations for CMS data sets
TAPPDS	load OS partitioned data set (PDS) files from tape to disk
TXTLIB	perform maintenance on text libraries
TYPE	type all or part of a file at the terminal
UPDATE	make changes in a file as defined by control cards in a record file
VDUMP	to print out a VM/370 system dump

Figure 22. CMS Command Summary (Part 3 of 3)

Any of the commands in Figure 22 may be entered when the user is running CMS in his virtual machine and the terminal is idle and the keyboard is unlocked. If the keyboard is locked (a command is executing) the user may press the ATTN key (or its equivalent) and when the Attention interrupt occurs, enter the command. It will not be executed until the command that is currently executing completes.

In addition to the commands listed in Figure 22, there are four CMS commands called Immediate Commands, which are handled in a different manner from the others. They may be entered while another CMS command is executing by pressing the ATTN key (or its equivalent), and will be executed immediately. The Immediate Commands are:

- HO–Halt tracing
- HT–Halt typing
- HX–Halt execution
- RT–Resume typing

Appendix G: Representative Directory Entries

The following sample user directory entries may be loaded as shown to provide an operating skeletal system with the user entries necessary for operation and maintenance. The indenting shown below is for clarity only, and is not required by the Directory program. Blank cards and cards with an asterisk (*), are ignored. LINK cards are used whenever possible to minimize the number of changes to the user directories whenever a minidisk extent is moved.

FILE: USER DIRECT

```
* USER DIRECTORY DECK
DIRECTORY 330 3330 CPDSK3
  USER OPERATOR PASSWORD 256K 1M ABCDEG
    ACCOUNT NUMBER BIN1
      CONSOLE 01F 3215
      SPOOL C 2540 READER A
      SPOOL D 2540 PUNCH A
      SPOOL E 1403 A
      LINK CMSSYS 190 190 RR
      MDISK 191 2314 1 10 UDISK1 WR RPASS WPASS

  USER OPERATIONS PASSWORD 256K 1M ABC
    ACCOUNT NUMBER BIN2
      CONSOLE 01F 3215
      SPOOL C 2540 READER A
      SPOOL D 2540 PUNCH A
      SPOOL E 1403 A
      LINK CMSSYS 190 190 RR
      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

  USER CPSYS PASSWORD
    ACCOUNT NUMBER BIN3
      CONSOLE 01F 3215
      SPOOL C 2540 READER A
      SPOOL D 2540 PUNCH A
      SPOOL E 1403 A
      LINK CMSSYS 190 190 RR
      MDISK 191 2314 31 10 UDISK1 WR RPASS WPASS
      MDISK 192 2314 102 101 UDISK2 WR RPASS WPASS

  USER CMSSYS PASSWORD
    ACCOUNT NUMBER BIN4
      CONSOLE 01F 3215
      SPOOL C 2540 READER A
      SPOOL D 2540 PUNCH A
      SPOOL E 1403 A
      MDISK 190 2314 1 101 UDISK2 WR RPASS WPASS
      MDISK 191 2314 41 10 UDISK1 WR RPASS WPASS
```

```
USER SRMAIT PASSWORD 256K 1M FG
  ACCOUNT NUMBER BIN5
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    LINK CMSSYS 190 190 RR
    MDISK 191 2314 51 10 UDISK1 WR RPASS WPASS

USER PSRMAIT PASSWORD 256K 1M EG
  ACCOUNT NUMBER BIN6
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    LINK CMSSYS 190 190 WR
    MDISK 191 2314 61 10 UDISK1 WR RPASS WPASS
    LINK CPSYS 192 192 W

USER USER1 PASSWORD
  ACCOUNT NUMBER BIN7
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    LINK CMSSYS 190 190 RR
    MDISK 191 2314 71 10 UDISK1 WR RPASS WPASS

USER USER2 PASSWORD
  ACCOUNT NUMBER BIN8
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    LINK CMSSYS 190 190 RR
    MDISK 191 2314 81 10 UDISK1 WR RPASS WPASS
    MDISK 230 3330 0 100 MFTSYS
    MDISK 231 3330 50 20 MFTUSE W

USER USER3 PASSWORD
  ACCOUNT NUMBER BIN9
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    LINK CMSSYS 190 190 RR
    MDISK 191 2314 91 10 UDISK1 WR RPASS WPASS
```

```
USER BATCH PASSWORD 512K
  ACCOUNT NUMBER BIN10
    OPTION REALTIMER ISAM
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    DEDICATE 230 BATCH1
    DEDICATE 231 BATCH2

USER TESTSYS PASSWORD
  ACCOUNT NUMBER BIN11
  OPTION ECMODE
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    LINK CMSSYS 190 190 R
    MDISK 330 3330 390 15 SYSWRK WR
    MDISK 331 3330 1 20 SYSWRK WR

USER 2780 PASSWORD
  ACCOUNT NUMBER BIN12
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    DEDICATE 30 30

USER JFM PASSWORD 512K
  ACCOUNT NUMBER BIN13
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    MDISK 230 2314 000 050 OSDOS1 WR
    MDISK 231 2314 100 020 OSDOS1 WR
    MDISK 232 2314 031 030 CPVOL1 WR

USER JCE PASSWORD
  ACCOUNT NUMBER BIN14
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    MDISK 130 2314 050 050 OSDOS1 WR
    MDISK 131 2314 001 020 CPVOL1 WR
```

```
USER APLSYS PASSWORD
  ACCOUNT NUMBER BIN15
    CONSOLE 01F 3215
    SPOOL C 2540 READER A
    SPOOL D 2540 PUNCH A
    SPOOL E 1403 A
    DEDICATE 191 SYSAPL
    DEDICATE 190 APLWRK
    SPECIAL 80 2702 IBM
    SPECIAL 81 2702 IBM
    SPECIAL 82 2702 IBM
    SPECIAL 83 2702 IBM
```

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

Index

Trim Along This Line

READER'S COMMENTS

Title: IBM Virtual Machine Facility/370: Planning Guide

Order No. Order No. GC20-1801-0

Please check or fill in the items; adding explanations/comments in the space provided.

Which of the following terms best describes your job?

□ Programmer
□ Manager
□ Operator
□ Instructor
□ Systems Analyst
□ Engineer
□ Mathematician
□ Student/Trainee
□ Customer Engineer
□ Systems Engineer
□ Sales Representative
□ Other (explain below)

Does your installation subscribe to the SL/SS? □ Yes □ No

How did you use this publication?

□ As an introduction
□ As a reference manual
□ For another purpose (explain) ____________________
□ As a text (student)
□ As a text (instructor)

Did you find the material easy to read and understand? □ Yes □ No (explain below)

Did you find the material organized for convenient use? □ Yes □ No (explain below)

Specific criticisms (explain below)

Clarifications on pages ____________________

Additions on pages ____________________

Deletions on pages ____________________

Errors on pages ____________________

Explanations and other comments:

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

Order No. GC20-1801-0

YOUR COMMENTS PLEASE . . .

This manual is one of a series which serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the back of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Trim Along This Line

FOLD FOLD

FIRST CLASS
PERMIT NO. 44308
CAMBRIDGE, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
VM/370 PUBLICATIONS
545 TECHNOLOGY SQUARE
CAMBRIDGE, MASS. 02139

FOLD FOLD

IBM Virtual Machine Facility/370: Planning Guide Printed in U.S.A. GC20-1801-0

IBM

**International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]**

**IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]**