GH20-0857-2

Type III Class A Program

IBM

Control Program-67/Cambridge Monitor System (CP-67/CMS) Version 3.2
Program Number 360D–05.2.005
Installation Guide

The purpose of this document is to provide the installation with instructions on creating a runnable CP-67/CMS system tailored to its configuration. There are also considerations for tuning the system as well as maintaining both CP-67 and CMS.

This manual should be read in its entirety before installing the system, as procedures have changed since Version 3, Modification Level 0.

NOTICE

This Type III Program performs functions which may be fundamental to the operation and maintenance of a system. It has not been subjected to formal test by IBM.

Until program reclassification, IBM will provide

- Central Programming Service, including design error correction and automatic distribution of corrections
- FE Programming Service, including:
  (1) Design error verification
  (2) APAR documentation and submission
  (3) Application of Program Temporary Fixes or development of an emergency bypass when required.

IBM does not guarantee service results or represent or warrant that all errors will be corrected.

The user is expected to make the final evaluation as to the usefulness of this program in his own environment.

THE FOREGOING IS IN LIEU OF ALL WARRANTIES EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Third Edition (May 1973)

This edition is a major revision and makes obsolete GH20-0857-1. Extensive technical changes have been made to this manual; therefore, the user should read it in its entirety.

This edition applies to Version 3, Modification Level 2, of the Control Program-67/Cambridge Monitor System (360D-05.2.005) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters. Information in this publication is subject to change. Therefore, be sure you have the latest edition and any pertinent Technical Newsletters.

This manual should be read in its entirety before installing the system, as procedures have changed since Version 3 Level 1.

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

A form has been provided at the back of this publication for readers' comments. If this form has been removed, address comments to: IBM Corporation, VM/370 Publications, 24 New England Executive Park, Burlington, Massachussetts 01803.

PREFACE

The following publications are referenced in this manual:

CP-67/CMS User's Guide GH20-0859

CP-67 Operator's Guide GH20-0856

Operating System 360: Utilities C28-6586

CP-67: Operating Systems in a Virtual Machine GH20-1029

OS/360 ISAM Program Logic Manual GY28-6618

The following publications provide additional information on CP-67/CMS:

CP-67/CMS System Description Manual GH20-0802

CP-67 Program Logic Manual GY20-0590

CMS Program Logic Manual GY20-0591

CMS SCRIPT User's Manual GH20-0860

CP-67/CMS Hardware Maintainability Guide GH20-0858

CONTENTS

FIGURES

INTRODUCTION

The initial installation of CP-67/CMS involves running CMS on the bare machine. CMS is used to load the CP-67 basic distribution tape onto a disk, read in the RIO configuration and SYSDESCR module for the installation and assemble the REALIO, SYSDESCR and other modules, if necessary, to select the desired options. Once this is done, an EXEC procedure is invoked to punch the CP-67 nucleus, utilities, and loaders. The punched decks are then used on the bare machine to format the CP-67 volumes (sysres, drums, etc.) and to load the nucleus and directory on the residence volume. If CP-67 is already installed, regenerations or new installations can be performed from a CMS virtual machine.

The basic CMS system tape contains an IPLable dump/restore of a System disk. After restoring the tape file, a Primary disk is formatted. Using the restored System disk and formatted Primary disk, the CMS Source tape and the CP basic System tape can be read onto the Primary disk, and SYSGEN of the system can then be accomplished.

The minidisk, or partial volume disk allocation, provides economical use of direct access storage space. Minidisks are available to both CMS and OS users, and can be used for both system residence and user-data volumes. Formatting of OS volumes is performed by the CP utility MINIDASD; the CMS FORMAT command is used for CMS volumes. In addition to formatting the disk for OS use, MINIDASD writes an appropriate VTOC which determines the amount of space available on the volume. Minidisks can be dumped to tape, and restored, using the CPDMPRST program, as a means of periodic backup.

Version 3 Level 1 of CP-67 contained features that made it incompatible with previous versions. Version 3 Level 2 is an extension of Version 3 Level 1. For this reason, utilities, loaders, or nucleus decks from releases prior to Version 3 must not be used. In particular, the real machine configuration description requires additional information to produce a valid RIO and SYSDESCR module. With Version 3 Level 1 virtual machines are dispatched with consideration given their priorities as defined in the directory.

CMS FOR CP-67 SYSGEN

To obtain CP-67, CMS is utilized. A description follows of how to obtain CMS and run it on the real machine (not under CP). This description is intended as a convenient summary, as more detailed information can be found under "Installing CMS".

First, restore the DUMP/RESTORE copy of the CMS System disk. The CMS System tape contains the following files:

file 1--IPL'able DUMP/RESTORE
file 2--a D/R copy of the CMS System disk.
file 3--a tape-dump of the CMS System disk.

Then, IPL the restored disk. Hit REQUEST on the 1052 console if its real address is not X'009' or X'01F'. CMS now allows you to reconfigure the CMS device address table to match the real device configuration of the installation. The following question is asked:

Q1: REDEFINE ADDRESSES? (YES,NO):
Answer YES if the real addresses are different from the standard CMS virtual addresses listed below.

It is then necessary to enter the four-digit addresses for the following devices on the real machine if they differ from the address specified in the standard CMS device table: console, S-disk, P-disk, reader, punch, printer, tape one, and tape two. In looking for the 1052 console when CMS is IPLed on the real machine, CMS looks for device 009 and then 01F before waiting for the first interrupt denoting the console.

Q2: WILL THE CP-SAVE FUNCTION BE USED? (YES,NO):

Answer NO to the CP-SAVE question.

The standard CMS device addresses are as follows:

Device	Virtual Address	Symbolic Name	
1052	009	CON1	console
2311,2314	190	DSK1	system disk (read-only)
2311,2314	191**	DSK2	permanent disk (user files)
*2311,2314	192**	DSK3	temporary disk (workspace)
*2311,2314	000**	DSK4	A disk (user files)
*2311,2314	000**	DSK5	B disk (user files)
*2311,2314	19C**	DSK6	C disk (user files)
1403	00E	PRN1	line printer
2540	00C	RDR1	card reader
2540	00D	PCH1	card punch
*2400,3420	180	TAP1	tape drive
*2400,3420	181	TAP2	tape drive

---

* The 2311 or 2314 for the temporary disk, the A, B, and C disks, and the two tape drives are optional devices; they are not included in the minimum configuration.

** The reference between the virtual address and I/O device may be changed at any time by the CMS LOGIN command.

After entering the real device addresses, the CMS initialization message is typed

```
CMS .. VERSION n   LEVEL  m
```

Issue the CMS command

```
FORMAT P ALL
```

to format the P-disk to be used for CP-67 SYSGEN. The card punch and printer should be placed in READY status.

Place the CP distribution tape on symbolic drive TAP2 and issue

```
TAPE LOAD 4
```

All necessary CP files are loaded onto the P-disk. These files are of the following filetypes for generation of CP-67 and the utilities:

SYSIN files, which are the CP source decks written in Assembler Language;

EXEC files, which contain procedures for assembling and updating;

TEXT files containing the assembled modules of the system;

MACLIB files containing CP-67 macros for assembling the system and the real machine configuration;

ASP360 and COPY files for creating and changing the macro library CPMACS MACLIB, when necessary;

LOADER files containing two relocating loaders;

SAMPLE files containing examples of the various decks required for system setup.

It is from this P-disk (on which the above CP files are loaded) that the CP system is obtained.

Note: While executing CMS on the real machine, the last card of any punched deck must be manually run out from the card punch.

MACHINE CONFIGURATION DEFINITION FOR CP-67

A description of the physical machine must be provided for CP-67. This description is contained in a SYSIN or source deck called RIOxxx, where xxx are any three characters specified by the installation. This SYSIN deck is assembled via the CMS ASSEMBLE command, and the resultant text or object deck placed in the CP-67 system. Some simplex and duplex machine configurations are distributed among the P-disk files with the identifiers of RIOCSC SYSIN, SIMPLEX SAMPLE, and DUPLEX SAMPLE. If the distributed RIOCSC SYSIN does not meet the installation's configuration, a new RIOxxx deck must be constructed from the macros below, assembled, and used in place of the distributed RIOCSC text deck.

To get the real I/O definitions for the target configuration read onto the P-disk, from the card reader, place the RIOxxx deck into the reader, ready it, and issue the CMS command

```
OFFLINE READ RIOxxx SYSIN
```

where xxx are any three characters specified by the installation. If CSC is chosen for xxx, the existing file RIOCSC SYSIN is erased.

To assemble the real I/O configuration deck, produce the text deck, and print the listing, issue the CMS command

```
SYSASM RIOxxx PTFX
```

where xxx are the same three characters specified with the OFFLINE READ command. The TEXT deck is not punched but remains on the P-disk for the generation of CP.

The physical (real) machine is described to CP via control blocks containing information about the input/output devices, channels, and control units. These control blocks are generated by the following CP-67 macros which are described below: REALIO, SIMPLEX, DRCH, DRCU, DRDEV, and DMXDV.

The following conventions are used throughout this description: 1. variable information is indicated in lowercase and system key-words are indicated in uppercase letters, whereas both are written in uppercase when macros are punched in cards; 2. < and > are used to bracket choices and the brackets are not part of the macro definition (for example, RDEVPNT=<0,dlabel>) would be used to indicate that RDEVPNT=0 or RDEVPNT=dlabel could be used; 3. a pair of |'s is used to indicate an optional parameter and the |'s are not a part of the macro definition (for example, SIMPLEX |n| would be used to indicate that SIMPLEX or SIMPLEX n could be used).

MACHINE CONFIGURATION MACROS

REALIO Macro

The REALIO macro is the first card of the deck. Its purpose is to generate the prerequisite entry points for use by the system and to give a CSECT name to the deck. Its use is

```
|label|  REALIO  |TITLE='Listing Header'|
```

where "label" becomes the alphabetic serialization of the text deck and title appears at the top of the pages in the listing.
Note. label must be four characters or less.

SIMPLEX Macro

The SIMPLEX macro follows the unit, control unit, and channel specifications for any given CPU. It provides for the creation of the necessary tables required by CP and is dependent on the configuration specified. If a simplex configuration is described, the SIMPLEX macro must be followed by the END card. If a duplex configuration is being described, two SIMPLEX macros are required: the first SIMPLEX macro is

followed by the I/O definitions for another CPU; the second SIMPLEX macro is followed by the END card.

```
SIMPLEX |n|
```

where 'n' is either null for a simplex system, '1' for the first half of a duplex specification, or '2' for the second half of a duplex configuration.

I/O Block Definition Macros

The following macros are used to describe the physical input/output units available:

Define Channel Macro:

```
|label|  DRCH  |CHANPNT=<0,chlabel>|,
               RCULIST=list,CHANADD=c
```

where label is the symbolic label of this channel; chlabel is the symbolic label of the next channel in the channel list; list is the symbolic label of the first control unit on this channel; and c is the channel address.

Define Control Unit Macro:

```
|label|  DRCU  |RCUPNT=<0,culabel>|,
               DEVLIST=dlabel,RCUADD=c,
               |RCUPATH=<0,path>|,CUTAIL1=tail1
```

where label is the symbolic label of this control unit; culabel is the symbolic label of the next DRCU macro for the next control unit on this channel, if any; dlabel is the symbolic label of the first device defined for this control unit; c is the control unit address; path is the value (hexadecimal) of the logical path for this control unit (each control unit on a channel has a unique bit of the eight bits assigned to the channel and this bit or path defines which devices are accessed through this control unit); tail1 provides a connection to this control unit from the channel--use the symbolic label of the channel on which this control unit resides.

Define Device Macro:

```
|label|  DRDEV  |RDEVPNT=<0,dlabel>|,
                RDEVCU=culabel,RDEVADD=ccu,
                TYPE=type,DECUPTH=path,
```

where label is the symbolic label for this device; dlabel is the symbolic label of the next device on this control unit; culabel is the label of the control unit on which this device resides; ccu is the channel, control unit, and unit address of this device; type is the device type (specified as xxxx where xxxx is 2311, 2314, 2321, 2301, 2303, 2400, 3420, 1403, 2540P, 2540R, 2701, 2702, 2703, or 2250; "path"

is identical to the control unit's path (RCUPATH in DRCU) for this device and defines which control unit this device uses.

Define Multiplexor Device Macro:

```
|label|  DMXDV  RDEVADD=ccu,TYPE=type,|SAD=n|,

                |RDEVPNT=<0,mlabel>|
```

where "label" is the symbolic label of this multiplexor device; ccu is the channel, control unit, and unit address of this device; type is the device type, which is one of the following: 1052, 1403, 2540RDR, 2540PCH, 2701T, 2702T, 2703T, or TT35T (where the 2701T, 2702T, 2703T implies a 1052, 2741-BCD or 2741-correspondence terminal coming into a 2701, 2702, or 2703 and the TT35T is a teletype 33 or 35); n is the SAD address of the 2701, 2702, or 2703 and has a value of 0, 1, 2, 3, or 4 (the SAD address does not indicate terminal type); For a 2701 which does not require a SAD command, specify a value of 4. For a 2703 which ignores SAD commands, a value of 4 can also be specified. For a 2702, the correct SAD value for the particular line must be specified (0, 1, 2, and 3 are valid). Your local IBM CE can supply you with the SAD numbers for each 2702 line; and mlabel is the symbolic label of the next multiplexor device in the chain. There must be one DMXDV macro defined for each 2701, 2702, or 2703 line.

Note: The ordered arrangement of the real I/O macros is required to properly generate the correct entries and tables.

The multiplexor devices must be defined before the selector channel devices. Each selector channel must be completely defined in terms of devices, control unit, and then channel (in that order) before the next selector channel is defined. If a simplex configuration is being described, the SIMPLEX macro must be followed by the END card. If a duplex configuration is being described, two SIMPLEX macros are required: the first SIMPLEX macro is followed by the I/O definitions for another CPU; the second SIMPLEX macro is followed by the END card.

An example of a real I/O source deck for a simplex configuration is as follows:

```
RIOS     REALIO TITLE='SAMPLE SIMPLEX'
OPCONSOL DMXDV RDEVPNT=PRINTER1,RDEVADD=009,TYPE=1052
PRINTER1 DMXDV RDEVPNT=CARDRDR1,RDEVADD=030,TYPE=1403
CARDRDR1 DMXDV RDEVPNT=PUNCH1,RDEVADD=031,TYPE=2540RDR
PUNCH1   DMXDV RDEVPNT=TERM01,RDEVADD=032,TYPE=2540PCH
TERM01   DMXDV RDEVPNT=TERM02,RDEVADD=020,TYPE=2702T,SAD=1
TERM02   DMXDV RDEVPNT=TERM03,RDEVADD=021,TYPE=2702T,SAD=1
TERM03   DMXDV RDEVPNT=TERM04,RDEVADD=023,TYPE=TT35T,SAD=2
         .....
         .....
TERM4E   DMXDV RDEVADD=04E,TYPE=2703T,SAD=4
DRUM2    DRDEV RDEVCU=R2820A,RDEVADD=100,TYPE=2301,DECUPTH=80
R2820A   DRCU  DEVLIST=DRUM2,RCUADD=0,CUTAIL1=CHAN1,RCUPATH=80
CHAN1    DRCH  RCULIST=R2820A,CHANADD=1,CHANPNT=CHAN2
DISK30   DRDEV RDEVCU=R2314,RDEVADD=230,TYPE=2314,DECUPTH=40,       *
               RDEVPNT=DISK31
DISK31   DRDEV RDEVCU=R2314,RDEVADD=231,TYPE=2314,DECUPTH=40,       *
               RDEVPNT=DISK32
         .....
         .....
```

```
DISK37   DRDEV RDEVCU=R2314,RDEVADD=237,TYPE=2314,DECUPTH=40
R2314    DRCU  DEVLIST=DISK30,RCUADD=3,CUTAIL1=CHAN2,RCUPATH=40
CHAN2    DRCH  RCULIST=R2314,CHANADD=2,CHANPNT=CHAN3
DRUM1    DRDEV RDEVCU=R2841B,RDEVADD=393,TYPE=2303,DECUPTH=80,      *
               RDEVPNT=DISK4
DISK4    DRDEV RDEVCU=R2841B,DEVADD=390,TYPE=2311,DECUPTH=80,       *
               RDEVPNT=DISK5
         .....
         .....
R2841B   DRCU  DEVLIST=DRUM1,RCUADD=9,CUTAIL1=CHAN3,RCUPATH=80,    *
               RCUPNT=R2250
SCOPE1   DRDEV RDEVCU=R2250,RDEVADD=306,TYPE=2250,DECUPTH=40
R2250    DRCU  DEVLIST=SCOPE1,RCUADD=0,CUTAIL1=CHAN3,RCUPATH=40
CHAN3    DRCH  RCULIST=R2841B,CHANADD=3,CHANPNT=CHAN0
TAPE1    DRDEV RDEVCU=R2400A,RDEVADD=0C0,TYPE=2400,DECUPTH=80
TAPE2    DRDEV RDEVPNT=TAPE2,RDEVCU=R2400A,RDEVADD=0C1,TYPE=2400,   *
               DECUPTH=80
R2400A   DRCU  DEVLIST=TAPE1,RCUADD=C,CUTAIL1=CHAN0,RCUPATH=80
CHAN0    DRCH  RCULIST=R2400A,CHANADD=0,CHANPNT=CHAN0A
D2701A   DRDEV RDEVCU=R2701A,RDEVADD=010,TYPE=2701,DECUPTH=80
R2701A   DRCU  DEVLIST=D2701A,RCUADD=1,CUTAIL1=CHAN0A,RCUPATH=80
CHAN0A   DRCH  RCULIST=R2701A,CHANADD=0,CHANPNT=CHAN0B
D2701B   DRDEV RDEVCU=R2701B,RDEVADD=012,TYPE=2701,DECUPTH=80
R2701B   DRCU  DEVLIST=D2701B,RCUADD=1,CUTAIL1=CHAN0B,RCUPATH=80
CHAN0B   DRCH  RCULIST=R2701B,CHANADD=0
         SIMPLEX
         END
```

Note: Nonshared multiplexor devices, except for tapes, should be defined as separate channel, control unit, and device blocks (macros); that is, each device has its own set, as shown for D2701A and D2701B above. This avoids lock-out problems on that channel if unusual I/O operations are performed by users on a particular device.

DRCU macros specified for control units which occur on different multiplexor shared subchannels should be generated with separate DRCH macros in order to insure maximum throughput. For example, suppose the real configuration has four tape drives on 0C0 through 0C3, and four more units sharing subchannel D (0D0 through 0D3), then each subchannel should have associated with it a single DRCH specification.

Further examples of REAL I/O configuration decks are available from the CP-67 distributed P-disk files: SIMPLEX SAMPLE, DUPLEX SAMPLE, and RIOCSC SYSIN.

SETUP OF INSTALLATION PARAMETERS

Installation parameters must be provided for CP-67 just as the machine description was provided. Installation parameters are described in a SYSIN deck called SYSDESCR, which is assembled and the resultant TEXT deck placed in CP-67. The SYSDESCR deck contains the following items: the SYSRES macro which describes the system residence volume, the SYSGEN macro which defines installation variables, a constant called SCREDAT which is the creation date and character string printed on the 1052 at system IPL, and a constant called SYSCHAR which is the character to be printed when CP console function mode is entered or a CP console function error occurs. A sample SYSDESCR SYSIN deck is distributed with the P-disk files along with the SYSDESCR TEXT deck. If the distributed SYSDESCR SYSIN does not meet the installation's needs, it should be

altered as needed and reassembled as described for RIOxxx SYSIN in "Machine Configuration Definition for CP-67".

The SYSRES and SYSGEN macros are described below. The SCREDAT and SYSCHAR constants are illustrated in the example below as well as in the distributed sample SYSDESCR SYSIN deck.

SYSRES Macro

The SYSRES macro is used to describe the system residence volume. Its format is

```
SYSRES SYSRES=ccu,SYSTYPE=type,SYSVOL=volid,SYSERR=aaa,
               SYSDNC=bbb,SYSWRM=ccc
```

where ccu is the address of the disk onto which the nucleus is written; type is 2311 or 2314; volid is the CP-67 formatted label of the volume (that is, CPDSK1); aaa is the cylinder allocated as permanent for error recording; bbb is the starting cylinder for nucleus residence; and ccc is the cylinder for warm start control.

SYSGEN Macro

The SYSGEN macro is used to describe certain installation variables used for system operation. Its format is

```
SYSGEN SYSOPER=userid,SYSDUMP=dumpid,SYSEMRG=xxx,
       SYSCNSL=aaa,SYSPRT=bbb,SYSPUN=ccc,SYSCORE=dddK
```

where userid is the operator's id for AUTO LOGIN; dumpid is the user's id for whom system dumps are made available from spooling files; xxx is the 270x line address for the emergency console startup; aaa is the system console address; bbb is the system line printer; ccc is the system punch, and ddd is the storage size of the real system expressed in K.

An example of a SYSDESCR SYSIN follows:

```
SYSD     TITLE 'SYSDESCR     VERSION 3, LEVEL 2'
SYSDESCR START 0
         ENTRY SCREDAT
         DC    AL1(SCREDATF-SCREDAT)    LENGTH OF MESSAGE
SCREDAT  DC    CL10'05/15/73'
         DC    C'RELEASE 3.2'
SCREDATF EQU   *
         SPACE 1
         ENTRY SYSCHAR
SYSCHAR  DC    C'CP'
         SPACE 1
         DS    0D
         SYSRES SYSRES=230,SYSTYPE=2314,SYSVOL=CPDSK1....
         SYSGEN SYSOPER=CPSYS,SYSDUMP=00E,SYSEMRG=02F....
         END
```

In the above example, the SCREDAT field can be set to the date of system creation. This date and the following character string are printed on the 1052 at system IPL time.

The SYSCHAR characters are printed to show the level of the virtual

machine; that is, when the virtual machine is in console function mode, these characters are printed for command errors and verification.

SELECTION OF SYSGEN OPTIONS

Currently there are several options to be chosen during system generation: the real timer, the virtual 67, the statistical gathering/tracing options, and ISAM. To obtain the options, the file LOCAL COPY on the P-disk must be changed by using the CMS command EDIT. For each option there is a corresponding statement in the LOCAL COPY file, for example:

```
columns 1              10    16
        option name    SETA  0
```

A 0 or 'NO' means the option is not selected; a nonzero numeric or 'YES' in the operand field means the option is selected. Therefore, to obtain each option, the appropriate operand field must be chosen. Once the options have been selected, the corresponding CP-67 module(s) must be updated and assembled by issuing the following CMS command:

```
SYSASM name PTFX
```

Note: After the LOCAL COPY file has been edited for the selection of options, the COPY file must be placed in a MACLIB for access by the ASSEMBLER. The following procedure may be used to create a new MACUP maclib with an updated version of LOCAL COPY.

1. Edit the EXEC file MACUP EXEC to add the name LOCAL to the list of macros and copy files contained in MACUP MACLIB. Position the editor line pointer to the bottom of the file and add the following card:

   ```
   &1 &2 &3 LOCAL
   ```

2. Use the SYSMAC EXEC utility to regenerate the MACUP MACLIB file. The following illustrates the use of this command:

   ```
   SYSMAC MACUP PTFX
   ```

The assembled modules distributed with this system contain the following options:

```
&RTIMR     SETA   10
&V67       SETA    1
&ISAM      SETC   'YES'
&TRACE(5)  SETB    1
&TRACE(6)  SETB    1
```

If these options are not desired or are to be changed, update the LOCAL COPY file and reassemble the modules affected, as described for each option below.

Note: Use of these options affects system performance and should be selected with care. The TRACE options involve especially heavy overhead.

Real Timer--&RTIMR

The real timer option &RTIMR provides for updating a virtual machine's timer when the machine is in wait state, if that virtual machine has the REAL TIMER option in the DIRECTORY. This option allows a dormant machine (for example, APL) to be awakened by a timer interrupt. (See "Directory Creation" for details.)

The modules to be assembled for the &RTIMR option are DISPATCH and SCHEDULE.

For the real timer option the SETA symbol is set to the number of concurrent real timers that the installation is willing to support. For instance, the statement

```
&RTIMR   SETA   6
```

produces code in the dispatcher to maintain up to six real timers.

Virtual 360/67 Option--&V67

The virtual 67 option, &V67, provides code in the CP-67 nucleus to support virtual machines in which CP-67 can be run. Those virtual machines with the virtual 67 option specified in the directory have the facility to operate in virtual extended PSW mode with 24-bit addressing. The distributed system has the &V67 option selected. If the option is not desired, change the LOCAL COPY value to

```
&V67 SETA 0
```

and use the EXEC procedure SYSASM to reassemble the following modules:

```
CFSMAIN
DISPATCH
IOINT
LOGON
MVIOEXEC
PAGTRANS
PRIVLGED
PROGINT
QUEVIO
RESINT
PSA
UNSTIO
USEROFF
VIOEXEC
```

ISAM Option--&ISAM

Certain OS/360 ISAM channel programs use a self-modifying operation which under normal CP-67 processing is not allowed. With the &ISAM option selected, CP can detect the specific OS/360 ISAM channel programs and make changes in the translated channel program to handle the self-modifying sequence. With the option selected and with the virtual machine using OS/360 ISAM, CP restricts somewhat the location of the

ISAM CCW's. Four critical double words in certain channel programs cannot cross a page boundary. Refering to the OS/360 ISAM Program Logic Manual (Form GY28-6618) the following table lists the restricted channel program 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, CS80
CP23	CS17, CS18, CS19, CS19A
CP26	CS34, CS35, CS36, CS37

Only those users with the ISAM option in the 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.

The module to be assembled for the &ISAM option is CCWTRAN. The distributed version has the option selected.

If the option is not desired, change the LOCAL COPY value to

```
&ISAM    SETC  'NO'
```

and issue the CMS command

```
SYSASM CCWTRAN PTFX
```

Statistical Gathering / Tracing--&TRACE

This option is a binary array of 25 entries. Each element controls the selection of certain statistical gathering and tracing functions in CP-67. The following 6 elements are currently implemented.

&TRACE(1) SETB 1
used in module PSA to trace CP-67 SVC calls in an SVC trace table in that module. To select the option, enter the statement in the LOCAL COPY file, update the MACLIB and assemble the PSA SYSIN file.

&TRACE(2) SETB 1
&TRACE(3) SETB 1
these two options control the tracing of selector and multiplexor interrupts (real hardware) respectively. The trace table and code selection is in the IOINT module. To select either or both options, enter the statements in the LOCAL COPY file, update the MACLIB, and assemble the IOINT SYSIN file.

&TRACE(4) SETB 1
this option is used for FREE/FRET statistical gathering. With the option selected, such data as number of calls, number of elements, etc. is gathered by the FREE module. To select the option, enter the statement in the LOCAL COPY file, update the MACLIB, and assemble the FREE SYSIN file.

&TRACE(5) SETB 1 (selected in the distributed system)
this option is used to gather counts of user privileged instruction execution. The file STAT COPY defines the counters in low core that are used. To select the option, enter the statement in the LOCAL COPY file, update the MACLIB, and assemble the files PRIVLGED SYSIN, PROGINT SYSIN and VIOEXEC SYSIN.

&TRACE(6) SETB 1 (selected in distributed system)
this option is used to select the virtual machine tracing and address stop functions invoked with the SET TRACE and SET ADSTOP console functions. To select the option, enter the statement in the LOCAL COPY file, update the MACLIB, and assemble the files PSA SYSIN, PROGINT SYSIN, VIOEXEC SYSIN, UNSTIO SYSIN, DISPATCH SYSIN, RESINT SYSIN, TRACER SYSIN, and CFSSET SYSIN.

The &TRACE elements 7 through 25 are reserved for future use.

Note: The above procedure for options can also be applied to code added by an installation. The module can be assembled to selectively contain the installation's code (1) by placing a COPY OPTIONS and a COPY LOCAL instruction before the START card in the module being modified, (2) by adding a SETA instruction in the LOCAL COPY file, that is,

```
&LOCAL1    SETA  1
```

(where &LOCAL1 might mean all local modifications made by programmer number 1), and (3) by checking for the setting of the option in the module.

OBTAINING CP-67

To obtain the tailored CP-67 system, a load deck must be punched and loaded onto a properly formatted volume. To punch the CP load deck, the CP utilities, and card loaders, verify that the card punch is in READY status, then issue the CMS command:

```
CPGEN &1 &2
```

where:

&1 is the address of the printer that the loader uses to print the CP load map. A loader called RELDR punches at the start of the deck; (for example, 00E or 030).

&2 is the full name of the real I/O configuration deck that was specifically assembled for the system in the procedure above (for example, RIOCSC or RIOABC).

The above command punches out the complete CP load deck (about 2500 cards with each text deck uniquely identified in columns 73-75) and each deck is separated by an identifying card with the format

```
OFFLINE READ fname ftype
```

where fname is the object deck filename and ftype is TEXT.

Note: There is a pause after punching the CP-67 LOAD DECK to allow the user to clear the punch and get the last card. The user can continue by typing "ready" or he may stop here by typing "quit". This is clearly stated on the terminal as the CPGEN operation proceeds.

OBTAINING THE UTILITIES

The utilities supplied with CP-67 are as follows:

```
DIRECT
FORMAT
SAVESYS
VDUMP
FDUMP
EDITDUMP
MINIDASD
CPDMPRST
SAVEOS
CPIPL
```

The first three CP utilities are punched out as part of the CPGEN procedure above. The second file output contains all the utilities, each identified by a deck separator of the OFFLINE READ form (as outlined above for the CP load deck).

The CP utilities, FORMAT and SAVESYS, are stand-alone programs and must have the RELDR loader on the front with a "CNTR WTR" control card that reflects the correct printer address. The DIRECT utility runs stand-alone or under CMS. If it is run stand-alone, it requires a loader. The VDUMP utility runs only under CMS. It is used to retrieve CP-67 dumps from disk after an ABEND in the CP supervisor. The VDUMP utility is run by the user specified in the SYSGEN macro of SYSDESCR as the SYSDUMP parameter. In addition to the standard CMS machine configuration, the user must also have a reader printer device in his virtual machine as

```
UNIT   OF1,RPRT
```

in order to run VDUMP. See "Obtaining CP-67 Dumps--VDUMP" for details.

SAVEOS and CPIPL are OS programs, to be run in an OS virtual machine, and are equivalent to SAVESYS for an OS system. The CMS files for implementation are: SAVEOS JOB, SAVEOS SYSIN, and CPIPL SYSIN. Reference Operating Systems in a Virtual Machine for further information.

See the CP-67 Operator's Guide for a description of CPDMPRST, a CMS program that dumps and restores 2311 or 2314 volumes between disk and tape.

Detailed descriptions of each of the other utilities are found in the ensuing sections.

To obtain copies of the loader, issue the CMS command

```
OFFLINE PUNCH RELDR LOADER
```

The default printer address is 010.

If an installation has a printer of any other address, use the RELDR LOADER and immediately after the loader and before the TEXT file(s) to be loaded place the following control card:

```
column  1 2      6
          DEV    PRNT=xxx,TYPW=yyy
```

column 1 - 12, 2, 9 punch

where xxx is the printer address and yyy is the system console address (default console address is X'009').

The CPGEN procedure punches out three loaders for the specified printer address &1 (that is, 00E) to be used with various loading functions and also three loaders which do not print a load map; these are useful when loading the CP utilities.

FORMATTING OF VOLUMES

Once the CP utilities have been punched out, the direct-access volumes used by the system (for paging, spooling, nucleus residence, and directory) must be properly formatted. This is accomplished by means of the FORMAT utility. The detailed instructions for operating FORMAT follow.

After the operator's console has been identified, the following message appears:

```
CP/67 DASD FORMAT PROGRAM
```

Then the program requests the type of device being formatted:

```
DEVICE TYPE
```

where the response may be four digits indicating the type of device (that is, 2301, 2303, 2311, 2314) and an optional fifth character, p, which causes only partial formatting to take place (that is, 2314p).

If the device type is recognized, the system requests the unit address

```
DASD ADDRESS =
```

where the response is the channel and unit address as three hexadecimal digits (for example, 100, 1C0, 230).

If the device is ready, the system asks for the new label to be written by typing

```
VOLUME LABEL =
```

where the response is six characters (uppercase or lowercase) written on the label record of the volume.

If partial formatting was requested when indicating device type, the system now asks the operator to define the area to be formatted by typing

```
START CYL.(HEX) =
```

where the response is null for cylinder 0, "LO" (Label Only) if only the label is to be written, or two hexadecimal digits specifying the starting cylinder. The system then requests the ending cylinder by typing:

```
END CYL.(HEX) =
```

where the response is null for the entire volume (whose actual value varies with the device type) or two hexadecimal digits for a specific cylinder.

The system indicates that the format operation has commenced when its data is complete. At termination, the program will so indicate and may be restarted by pushing REQUEST.

The system's residence volume for CP is usually formatted with the label CPDSK1, as that label is required for IPL'ing by name. If it is desired to change the volume label, the file SYSTEM SYSIN must be modified to reflect the desired system's residence volume label and SYSTEM SYSIN must be reassembled and replaced in the CP load deck.

ALLOCATION AND DIRECTORY CREATION

Allocation

All 2314 and 2311 volumes to be used for CP-67 spooling and/or paging must have space allocated. The system residence volume must have space allocated for the system directory and the nucleus.
Note: 2301 and 2303 drums must not be allocated since they are specially formatted for dynamic paging allocation. The DIRECT utility (ALLOCATE control cards) is used to allocate space for the system residence volume and others, if necessary. For 2314 residence, the following space must be allocated as permanent:

Cyl.	Usage
0	IPL and file directory
'SYSERR'	error recording (1 cyl required)
'SYSWRM'	warm start control (1 cyl required)
'SYSDNC'	nucleus residence (2 cyl required)
to 'SYSDNC+1'	nucleus residence

The values for the symbolic cylinder locations are defined in the SYSDESCR deck with the SYSRES macro.

For 2311 residence, the following space must be allocated as permanent:

Cyl.	Usage
0-1	IPL and file directory
'SYSERR'	error recording (1 cyl required)
'SYSWRM'	warm start control (1 cyl required)
'SYSDNC'	nucleus residence (5 cyl required)
to 'SYSDNC+4'	nucleus residence

The symbolic values are obtained from the SYSRES macro specified in the real machine configuration description.

Cylinder 0 and others may be allocated as DRCT space. In addition, some cylinders may be allocated as permanent for user file space or

named system residence. Other cylinders must be allocated as TEMP (for spooling or paging) or TDSK.

CP-67 requires an allocation record to be written behind the volume label on cylinder 0 head 0 record 3 of each disk that is to contain space for the directory, paging, and spooling. The allocation record specifies which cylinders on the disk are used as permanent space and which are temporary space. Permanent space on a volume consists of the areas reserved for user files and system residence. Temporary space consists of those areas used for paging and spooling by CP-67.

There are five types of allocation control cards: ALLOCATE, DRCT, TEMP, TDSK, and PERM. The format of the ALLOCATE card is

```
ALLOCATE UNIT=ccu,VOLID=xxxxxx
```

ccu is the address of the device to be allocated.

xxxxxx is the six-character label of the volume to be allocated.

ALLOCATE must start in column 1 and UNIT in column 11.

DRCT space is used for CP-67 directory residence.

TEMP space is used for CP-67 spooling and paging.

PERM space is for user direct-access files.

TDSK is for DASD units in the user machine descriptions that are to be allocated at login time from a pool of temporary disk space. The TDSK area is <u>not</u> used by CP for spooling and paging.

The control cards for the allocation of space are

```
DRCT   xxx,yyy
TEMP   xxx,yyy
PERM   xxx,yyy
TDSK   xxx,yyy
```

where DRCT, TEMP, PERM and TDSK start in column 10, and the cylinder designations in decimal form (with leading zeros to make three characters each) start in column 16. xxx is the first cylinder and yyy is the last cylinder.

There must be one ALLOCATE card for the CP residence volume and for each disk that contains <u>temporary space</u> (TEMP and TDSK). Each ALLOCATE card must be followed by the control cards for that volume. When all the allocation for a device has been specified, the following card is used to indicate the end of allocation cards for the volume:

```
*EOA*
```

The *EOA* begins in column 10. The *EOA* card is then followed by another ALLOCATE card or a DIRECTORY card.

Note that the system always makes cylinder 0 permanent for the volume label and allocation table residence. Therefore, <u>do not begin user files on real cylinder 0</u> of those disks that are being allocated. Checks are also made for exceeding device limits on the device address specified.

The allocation of volumes can be a separate run from that of the directory. Also, each volume can be allocated independently of the others.

The directory must be created after the direct-access volumes are properly formatted. A sample directory is contained on the P-disk in the file DIRECT SAMPLE. To obtain a printout of that file, issue the CMS command

```
OFFLINE PRINT DIRECT SAMPLE
```

The directory is created by the utility DIRECT, the instructions will be found in the next paragraphs. The allocation of space on the residence volume (as described above) must precede the directory creation.

Directory Creation

The CP system residence volume must contain a directory that describes the users and their virtual machines. The directory portion of this utility accepts the descriptions of the users and their virtual machines from the card reader specified at the operator's console and writes the directory in the temporary space previously allocated on the system residence volume (initially the disk must be formatted before ALLOCATE and DIRECTORY routines are applied). Depending upon the size of the directory and the device type of the system residence volume (that is, 2311 or 2314), at least the first two or three cylinders should be allocated as DRCT space on the CP system residence volume. Estimates on space requirements for the system directory are as follows:

2314-- 150 USER virtual machine descriptions per cylinder

2311-- 40 USER virtual machine descriptions per cylinder

These space requirements are in addition to cylinder 0 of the residence volume, which is used as the master directory. Thus, the directory allocation control card,

```
DRCT  000,003
```

provides space on three cylinders (1, 2, and 3) for virtual machine descriptions.

Every user who is to be permitted to log in to CP-67 must be described in the directory, including the system operator.

The directory description must begin with the following control card:

```
DIRECTORY UNIT=ccu,VOLID=xxxxxx
```

ccu is the real address of the CP system's residence volume on which the directory is to be written.

xxxxxx is the six-character volid of the disk on which the directory is written.

DIRECTORY must start in column 1 and UNIT in column 11.

After the DIRECTORY control card comes the OWN card. Its format is

```
OWN   aaaaaa,bbbbbb, ... ,hhhhhh
```

where "aaaaaa,bbbbbb, ... ,hhhhhh" are the volume labels of disks that contain allocation tables. OWN must start in column 10 and the volume labels in column 16. A maximum of eight labels can be specified on a single card. To name more than eight, use multiple OWN cards. A maximum of 40 owned volumes is allowed.

Note that the only volumes specified in the owned list are those that CP can use for TDSK allocations, paging and spooling, such as drums, and the system residence volume. Volumes that have no TEMP, DRCT, or TDSK space (for instance, OS, DOS, CMS volumes) should not be in the owned list, as it is obviously not necessary to allocate volumes that are all PERM space.

Next are the cards that describe a user and his virtual machine. These cards are described below. (Note that the label field always starts in column 1, the operand field or card type in column 10, and the arguments in column 16.)

```
userid     USER  password,account,priv-class,priority,options
```

The USER card initiates a new machine description file and creates a user directory entry for the specified user.

"Userid" is the one to eight character external name by which the user and all spooling output are labeled; it must be left-justified and trailing blanks included if userid is not eight characters.

"Password" is the key by which the user must respond when trying to login to the system. The password must be specified as eight characters, left-justified, with trailing blanks if eight characters are not specified. Note that if the terminal has the print inhibit feature, the system turns off the printing mechanism to maintain the security of the password, or if the user issues the LOGIN command with a mask character, one line of overprinting occurs before the user is allowed to enter his password.

"Account" is any eight-character identification for installation accounting uses.

"Priv-class" is the user's privilege class: A for system operator, B for system administrator, C for IBM customer engineering use, and D for normal users. This privilege class determines what console functions in the system the user may exercise. (See Appendix A in the CP-67 Operator's Guide for a table of console functions available to each privilege class.)

Note: Users with privilege class C do not have I/O errors recorded for them by CP-67. Use of this class should therefore be carefully restricted. Privilege class C can also execute certain DIAGNOSE functions to maintain the error-recording cylinder.

"Priority" is used by the dispatching scheme. The decimal number may range from 00 - 99, with 00 being the highest priority.

"Options" is a hexadecimal number that selects specific options available to users at login time and execution time. There are currently three user options available (see "Selection of SYSGEN Options"). They, and their hexadecimal values, are:

```
REAL timer support          X'80'
ISAM support                X'40'
VIRTUAL 360/67 support      X'20'
```

Note: The &ISAM option allows the user to do I/O operations as used by OS/360 ISAM. It is not required for DOS ISAM.

Multiple options can be selected by combining the values. For instance, real timer and virtual 67 would be specified as

```
... USER .....,X'A0'
```

The core card specifies the size of the user's virtual memory.

```
CORE   | ddddK |
       |   ddM |
```

It may be specified by one or more decimal digits followed by K or M, indicating a multiplier of 1024 bytes or 1,048,576 bytes respectively. The core size must be a multiple of 8K, where 8K is the minimum. When a user logs in to CP-67, he receives a virtual machine of this size. Paging space is not selected for this virtual machine until the user references core storage; space on the paging device is then selected one page at a time. If a user does not need a large core space, allocate less--for example, give the system operator 8K core.

```
UNIT  ccu,devtype

      |,<xxxxxx,(TEMP),REM=IDccu,CON=YES,DED=IDccu>|

      |,<relno,zzz>|

      |,last|

      |,RDONLY|

      |,RDSHAR=password|

      |,WRMULT|

      |,WRSHAR=password|
```

The UNIT card is used to specify the input/output units available to a user and where and on which volumes his permanent files reside.

ccu is the virtual address of the device.

devtype is the device type, which may be one of the following:

- 1052
- 2540P
- 2540R
- 1403
- 2250
- 2311 or 2312
- 2314

2400
3420
2701
2702
2703
TIMR
RPRT
RPUN

where a 2312 is used to designate the bottom half of a 2314 drive used as a virtual 2311.

A 2312 device type is treated as a 2311 by CP-67. User seek commands issued to the pseudo 2311 have the head address relocated (by adding decimal 10) to access the bottom half of the 2314 disk. It should be noted that 2311 device types supported on the top half of a 2314 pack may experience difficulties if multitrack operations are attempted; however, multitrack operations on the 2312 (pseudo 2311) device type should work correctly since a physical end-of-cylinder condition is recognizable.

TIMR is a pseudo-timer device which can provide the time of day, date, virtual CPU time, and total CPU time to a virtual machine. It should be specified for every CMS user. Its format is

```
UNIT  0FF,TIMR
```

RPRT and RPUN are special device types that allow the user to access the disk dump created by a CP-67 system abend. For the desired userid to receive the dump (see section on VDUMP) specify

```
UNIT  0F1,RPRT
```

xxxxxx is the volume label of the disk that is being described.

(TEMP) specifies that the disk space is to be obtained at logon time from that allocated as TDSK space and then removed from the user at CP-67 logoff.

REM=IDccu specifies a remote device with real address ccu that is to be used for spooled output from this defined user. It has the same function as the SPOOL console commands for directing spooled output. See CP-67/CMS User's Guide for details. The only device types supported as remote devices are the 2540P and the 1403.

DED=IDccu specifies a device with real address ccu that is dedicated to this user at login time and unavailable to other users. The device types supported are the 1403, 2540R, 2540P, 2250, 2400, 2701, 2702, and the 2703. If the specified device is not available or in use when the user logs in, he is so informed and the device is not attached to him. Note that a 2250 or 2400, if specified, must be defined as a dedicated device.

CON=YES specifies that continuous spooled input is desired for this virtual card reader; that is, if there is more than one spooled input file for the card reader, they are read as one continuous file before end-of-data is given. This has the same function as the SPOOL ccu CONT console function (see CP-67/CMS User's Guide for details.)

relno is a three-digit decimal cylinder relocation factor to be applied to this device; for example, if a user's permanent files started at cylinder 54 of the specified volume, relno would be specified as 054.

zzz is a three-digit decimal number specifying the number of temporary cylinders to be obtained at logon. zzz is used only if (TEMP) was specified.

last is a three-digit decimal number specifying the last cylinder of the user's file space. If relno is 054 and the user is allowed 20 cylinders of space, last would be specified as 073.

RDONLY specifies that the virtual direct-access volume can only be read and not written upon by the user associated with this directory entry. If RDONLY is not specified, the user has read/write access to the volume by himself, and no other user can read or write that volume unless WRMULT is specified for those users. (See WRMULT.) If the volume being described is the CMS system disk, RDONLY should be specified for each user.

RDSHAR=password is a one to eight-character password (left justified with trailing blanks included) that allows this volume to be shared for reading by users knowing the proper password. If ALL is specified as a password, all users have access to the volume for read-sharing. RDSHAR is only used for permanent volumes and not temporary ones. If either WRSHAR or WRMULT is specified, but not RDSHAR, a comma must be used to show that RDSHAR is omitted.

WRMULT allows the user to have access to the volume regardless of other users. If WRMULT and RDONLY are both specified for a user's volume, he will have read-only access to that volume, regardless of other users. If only WRMULT is specified, he has read/write access regardless of other users on that same volume. This parameter is intended for use by users using operating systems which provide an interlocking mechanism to protect files or for systems programmers who know what they are doing. CMS does not have an interlocking mechanism.

WRSHAR=password is a one to eight-character password that allows the volume to be shared for writing as well as reading, but only one user can have access to the volume at a time. The volume is shared among users knowing the proper password. If ALL is specified as a password, all users have write-sharing privileges on the volume.

Note: RDONLY is inconsistent with WRSHAR and consequently they may not be specified together. If RDONLY is specified on a volume for all users but one, and that one user logs in first, he will use the volume in a read/write status and other users will have no access to that volume unless they have the WRMULT parameter. When the RDONLY users log in, they receive the message

```
DEV ccu IN USE BY userid; NOT ATTACHED
```

The RDONLY users should either log out and then log back in or issue LINK console function after userid logs out or userid detaches that device from his virtual machine. If that read/write user logs in after a RDONLY user, he will receive the message

```
DEV ccu IN USE BY userid; SET TO R/O
```

and he will have read-only access to that volume. The userid will be one of the RDONLY users who logged on first.

If a read/write WRMULT user logs in after a RDONLY or a read/write user, the WRMULT user will have read/write access regardless of the other users on that volume, and he will receive the message

DEVICE ccu IN USE BY userid

If a RDONLY, WRMULT user logs in after any other user, he has read-only access to the volume, regardless of other users, and he receives no message.

The parameters specified with the UNIT control card are positional, therefore they must be specified in order and a comma must be used for each parameter omitted.

Each user's machine description is terminated by a card with

EOU

beginning in column 10. At the end of all user directory descriptions, a card with

EOD

beginning in column 10 is used to terminate the directory creation process and to return to the ALLOCATE and DIRECTORY card-scan routine. If there are no additional ALLOCATE or DIRECTORY cards, DIRECT is terminated; if there are additional ALLOCATE or DIRECTORY cards, DIRECT is executed again.

A sample allocation and directory deck are as follows:

```
ALLOCATE  UNIT=230,VOLID=CPDSK1
          DRCT  000,003
          PERM  004,004
          TEMP  005,197
          PERM  198,202
          *EOA*
ALLOCATE  UNIT=235,VOLID=CPDSK5
          PERM  000,000
          PERM  001,075
          TDSK  076,165
          PERM  166,202
          *EOA*
DIRECTORY UNIT=230,VOLID=CPDSK1
          OWN   CPDSK1,CPDR01,CPDSK5
OPERATOR  USER  CSC     ,A1234    ,A,80
          CORE  8K
          UNIT  009,1052
          *EOU*
USER1     USER  PASS1   ,222      ,D,30
          CORE  256K
          UNIT  009,1052
          UNIT  00E,1403
          UNIT  00C,2540R
          UNIT  00D,2540P
          UNIT  0FF,TIMR
          UNIT  190,2314,CMS190,000,053,RDONLY
          UNIT  191,2314,CPDSK5,166,175
          *EOU*
USER2     USER  PASS2   ,A590     ,C,30,X'80'
```

```
                CORE   256K
                UNIT   00E,1403,DED=ID030
                UNIT   009,1052
                UNIT   00C,2540R
                UNIT   00D,2540P
                UNIT   106,2250,DED=ID106
                UNIT   0FF,TIMR
                UNIT   190,2314,CMS190,000,053,RDONLY
                UNIT   191,2312,CPDSK6,148,152,,,WRMULT
                UNIT   193,2311,KVR999,000,202
                *EOU*
USER3           USER   PASS3    ,X3214567,C,30,X'C0'
                CORE   256K
                UNIT   009,1052
                UNIT   00E,1403
                UNIT   00D,2540P
                UNIT   00C,2540R
UNIT            0FF,TIMR
                UNIT   190,2314,CMS190,000,053,RDONLY
                UNIT   191,2314,CPDSK3,050,060,,RDSHAR=RDXX     ,WRMULT
                UNIT   192,2314,CPDSK5,001,075,,,,WRSHAR=MYPASS
                *EOU*
                *EOD*
```

CREATION OF PERMANENT RESIDENCE VOLUME

Once the volumes have been formatted and the directory created, the permanent CP residence nucleus should be created from the CP LOAD DECK. The last module in the CP load deck is the SAVECP module The function of this module is to create, after the load through the card reader, an IPL'able copy of CP on the residence volume from the core image. The volume address, label, and nucleus cylinders are indicated to SAVECP from information contained in the SYSDESCR module, in particular, the information from the SYSRES macro.

To create the permanent CP residence volume, ready the CP load deck into an available reader. The first module in the load deck must be the relocating loader, which prints the load map of the system onto the printer. Perform an IPL sequence on the card reader. The volume must be mounted and ready before the load completes. It is advisable to clear core before loading. If the disk is not ready, the message

```
**** DISK CUU NOT READY ****
```

is printed.

If the label on the disk is incorrect, the message

```
**** VOLID NOT dlabel ****
```

is printed.

If either message is printed, remedy the situation and press the EXTERNAL button to retry, or start from the card load.

At the termination of the load and the creation of the residence volume, the following message appears on the operator's terminal:

```
DISK LOAD OK
```

At this time, the permanent CP residence volume has been created and may

be IPL'ed to initiate system operation.

OBTAINING CP-67 FROM AN EXISTING CMS

For those installations that have an operating CP-67/CMS system, the CMS restore procedure outlined above can be bypassed. The following steps can be used to obtain CP-67:

1. Mount the distribution tape on an available tape drive attached to the desired userid as '181'.

2. Using CMS issue a TAPE LOAD 4 command. The four files will then load in their entirety onto the user's P-disk space.

   Note: About 50 cylinders of 2314 space are adequate to hold all the files and do assemblies.

3. Perform any necessary assemblies for configuration or options as outlined under "Setup of Machine Configuration" and "Selection of SYSGEN Options".

4. Follow the procedure under "Obtaining CP-67" using the CPGEN command.

5. BE SURE TO USE THE VERSION 3 UTILITIES AND LOADERS WITH VERSION 3 OF CP-67.

GENERATING "NAMED" OPERATING SYSTEMS UNDER CP-67

Providing a user with the capability of performing an IPL function by name, rather than by device, requires the installation to save a copy of this system on a DASD device in paging form. The self-loading deck which performs this is entitled SAVESYS. It accepts a control card of the form

```
SAVE VOLID=cccccc,UNIT=ccu,FP=nn,LP=mm,CYL=ppp,DISK=tttt,TRK=hhh
```

where:

SAVE must begin in column 1;

VOLID must begin in column 6;

cccccc is the volume label of the receiving DASD volume, which must have been formatted by the FORMAT utility;

ccu is the channel, control unit, and device address of the receiving volume;

nn is the page number of the first page to be written (in hexadecimal);

mm is the page number of the last page to be written (in hexadecimal).

ppp is the starting cylinder of the image; this is a real address, that is, a minidisk cannot be used;

tttt is the device type (either 2311 or 2314).

hhh is the starting track number of the image. If omitted, zero is assumed. If specified, it must be an even track number.

Note: Be sure that the parameters on the SAVE card correspond with those defined in the SYSTEM and SWPTABLE macros in the SYSTEM module.

The space indicated on the card must have been previously allocated as permanent space on the volume. The information on the control card must match that information provided in the SYSTEM module assembled and loaded with the system. In that module the system name, location, shared pages, if any, and operating conventions are established. In the distributed SYSTEM module, the system name is CMS, and eighteen pages are saved, beginning at page 0, up to and including page X'12'. The saved copy of CMS is written on the 2314 volume labeled CPDSK1 at cylinder 200.

The SYSTEM SYSIN file is constructed using a SYSTEM macro, which has the format

```
SYSTEM  NAME=nnn,FIRSTP=aa,LASTP=bb,TABLE=ttt,
          VOLID=xxx,EXADD=ccc,SYSMASK=sss,RUNKEY=kkk,
          SYSTAB=yyy
```

where:

nnn is the name for the system;
aa is the first page saved;
bb is the last page saved;
ttt is the label of the SWPTABLE macro associated with this system;
xxx is the label of the volume on which the system is saved;
ccc is the virtual machine execution address;
sss is the virtual machine system mask;
kkk is the virtual machine protection key;
yyy is the label of the shared page table in PAGTRANS if this system has shared pages; otherwise it is zero.

The SWPTABLE macro used with the SYSTEM macro has the following format:

```
ttt  SWPTABLE DASDORG=(ccc,hh,r),FIRSTP=aa,LASTP=bb,
       FIRSTSP=xx,LASTSP=yy,DISK=ddd
```

where:

ttt is a label (same as ttt in SYSTEM macro);
ccc,hh,r is the cylinder head and record address of where the system is saved;
aa is the first page saved;
bb is the last page saved;
xx is the first shared page, if any;
yy is the last shared page, if any;
ddd is the disk type where saved, 2311 or 2314.

The SYSTEM macro builds a table to define the name and running attributes of the saved system. The SWPTABLE macro builds a model SWPTABLE that is mapped to the virtual machine SWPTABLE in core to give that machine access (through paging) to the saved system.

Pages defined as shared in SWPTABLE are automatically LOCKed in core once the saved system is IPLed; they remain locked until nobody is

using the saved system.

The number of cylinders required to hold a named system depends upon the number of pages saved and the device type. For a 2314, one cylinder holds 30 pages; a 2311 cylinder holds 8 pages. Thus, CMS with 18 pages requires one 2314 cylinder or three 2311 cylinders.

Models of the SYSTEM and SWPTABLE macros are given below. The named system "OS" will be saved starting at real cylinder address 86 on the 2314 volume CPVOL1. 64 pages will be saved. When ipl-ed, execution will begin at X'88'.

```
      SYSTEM  NAME=OS,FIRSTP=00,LASTP=X'3F',
                    TABLE=OSTBL,VOLID=CPVOL1,
                 EXADD=88,SYSMASK=00,RUNKEY=00
OSTBL SWPTABLE DASDORG=(086,0,1),FIRSTP=00,
                    LASTP=X'3F',DISK=2314
```

When SAVESYS saves a copy of the system, it saves from FIRSTP to LASTP. If LASTP is greater than X'20', the SAVESYS module must be loaded into higher core, as it normally loads into X'20000'. To load SAVESYS into higher core, change the address in the SLC 20000 card at the front of the SAVESYS text deck.

The procedure for creating the page-form copy is as follows:

1. Load the required system into memory, with an address stop set to a location within that system where execution may be resumed without the system assuming any previous state in the machine (that is, registers, lower core locations, etc.). For CMS the address stop location is X'88'.

2. ATTACH the direct access device on which the system is to be saved to the virtual machine being used. The cylinders on which the saved system is to be stored must be formatted for CP file residence by the CP-67 utility program FORMAT. Follow the procedures outlined in this manual under "Formatting of Volumes". For the purpose of formatting an area which will hold a pageable copy of OS, the full volume must be ATTACHed, and the cylinder addresses specified must be real (i.e., it is not possible to use a minidisk). Do not attempt to use a 2301 or 2303 as the FORMAT program initializes those devices for dynamic paging only, and makes them unsuitable for saved system storage. Note that FORMAT will write a CP-67 volume label at cylinder 0, record 3 of the receiving volume.

   If the saved system is to be stored on a CP system disk (one that is available to CP for paging, spooling or CMS Temporary file space), then the CP utility program DIRECT must also be run. In the allocation control cards which are input to DIRECT, specify that the cylinders to be used for the saved system are permanent by supplying an appropriate PERM control card; refer to the section entitled "Allocation and Directory Creation". Note that DIRECT will allocate cylinder 0 of the receiving volume for its allocation tables.

3. When the system has entered manual state (that is, the address stop has been reached), load the SAVESYS deck into the system card reader and load from the reader.

The format of the SAVESYS deck is as follows:

High core loader
SLC 20000 card
SAVESYS program
LDT card
SAVESYS control card

4. If the save was successful, a message appears on the operator's terminal to that effect (see "Saving CMS under CP-67" in this manual for details of preparing CMS for the CP-67 SAVESYS function).

OPERATIONAL PROCEDURES

The device on which the shared systems residence volume exists must be ATTACHed to the CP-67 system to be made available to all potential users. Those users with appropriate directory entries will be given read-only access to the volume. Write access to the volume will be given if the directory entry so indicates, or if the real device containing the shared system volume is ATTACHed to the user's virtual machine. For the latter, the real device must first be DETACHed from the system.

TUNING OF CP-67

To prevent paging overload, CP-67 allows only a subset of the users to be eligible for dispatching at any one time. There are two queues for such users; Q1 and Q2. A user will be dropped from a queue after using an allotted amount of computer time (300 milliseconds for Q1, 3 seconds for Q2) or when CP-67 determines that the user is in wait state with no I/O outstanding for a high-speed device. A user enters Q1 whenever there is an I/O interrupt from the virtual operator's console or a virtual 270x line. If a user uses his allocated time in Q1, he is dropped from Q1 and scheduled for Q2. All Q1 users are serviced in a round-robin fashion before Q2 users. Q2 users are serviced according to their priority and operating characteristics. A user's priority is defined in the CP directory and can be examined by 'QUERY PRIORITY' once the user has logged in. The priority of a logged in user can be changed by 'SET PRIORITY userid nn'. It is recommended that all users of similar characteristics (i.e., running CMS) have the same or similar priority. Only certain users (i.e. high priority processing users or users not requiring average service) should have their priorities set higher or lower (smaller or larger number) respectively.

The number of concurrent Q1 users is limited to a maximum fixed number. In general we have found the maximum to be greater than any number actually achieved.

The number of concurrent Q2 users is limited by the paging activity of the system. A 'paging cost' value is calculated for each user, which is proportional to the paging activity caused by that user divided by the Q2 (paging activity control) value. A Q2 user is allowed to enter the queue if his 'paging cost' value plus the sum of all 'paging cost' values for users currently in both Q1 and Q2 is less than a system maximum. The Q2 value has a default of 14 and can be interrogated and changed from a class A ID via the 'QUERY Q2' and 'SET Q2' commands. A decrease or increase in the value of Q2 causes a corresponding decrease or increase in the paging activity of the system. The maximum range for

the Q2 value is from 5 to 25; the recommended range is from 8 to 16.

The paging algorithm is tied to this dispatching system. Whenever a page of memory is required, PAGTRANS selects a non-referenced page. The intent of this queue structure is to set a limit on the number of tasks that can compete for computer resources (memory and cpu), and to segregate short execution tasks (for example, edit requests) from long execution tasks (for example, FORTRAN compilation). By so segregating the tasks, limiting the number of long execution tasks, and minimizing paging between runnable users, excessive paging can be avoided.

The queue sizes are automatically adjusted at system IPL time based upon real core size. The queue values for various core capacities are

	256K	512K	768K	1024K	1280K	1536K	1792K	2048K
Q1	3	6	9	12	15	18	21	24

Q2 = 16 for all sizes

Note that the Q1 value is the maximum number of users in Q1 at any given time, while the Q2 value is a paging activity tuning value. The number of users in Q2 at any given time is computed dynamically and thus varies as a function of system load.

The operator has a console function to change the amount of system paging activity by altering the Q2 value as described above.

OBTAINING CP-67 DUMPS -- VDUMP, FDUMP, EDITDUMP

Note: The VDUMP EXEC procedure invokes the FDUMP and EDITDUMP modules that produce formatted CP-67 abend dumps from the disk dump for the user specified to receive the dump (SYSDUMP= in the SYSGEN macro).

The VDUMP program runs under the control of the Cambridge Monitor System. It is used to retrieve CP-67 dumps from disk after an ABEND in the CP supervisor. When a system crash occurs, a spool file is created for the virtual machine whose USERID is specified in the SYSGEN macro of SYSDESCR as the SYSDUMP parameter. The spool file then may be printed as a CP-67 system dump by issuing the CMS command VDUMP to the designated virtual machine (see section on AUTODUMP and RE-IPL).

In order to create the CMS command called VDUMP on the appropriate virtual machine, this procedure may be followed:

a. Read the punched card deck for VDUMP EXEC, FDUMP TEXT, and EDITDUMP TEXT into the appropriate virtual machine (attach a USERID card to the front of the deck).

b. IPL CMS

c. offline read *

d. load FDUMP (SINV

e. genmod FDUMP

f. load EDITDUMP (SINV

g. genmod EDITDUMP

The virtual machine which has access to the system dump created by the autodump feature must have the standard CMS virtual device addresses. In addition, this machine must contain the following directory entry for a special virtual device which reads the spooled file for the dump:

```
UNIT 0F1,RPRT
```

To print a dump, type the CMS command VDUMP. The system will respond with:

```
WHEN PROMPTED REPLY YES TO PRINT DUMP, NO TO IGNORE AND ERASE
CAUSE OF SYSTEM DUMP:
'cause' AT LOCATION xxxxxx REPLY YES OR NO
```

where cause is SVC 0, PROGRAM CHECK, MACHINE CHECK or UNKNOWN CAUSE.

The following error messages may be encountered:

I/O ERROR READING WIDE CARD
A permanent I/O error on the CP-67 spooling file has been encountered.

FIRST WIDE CARD NOT LOW CORE RECORD
The first spool file record did not correspond to the format expected.

READER EMPTY
No spool file exists in the "spool file reader" at address 0F1.

I/O ERROR WRITING TO DISK. TRY AGAIN.
A permanent error was encountered in writing the dump to the CMS disk.

When the dump is finished, the message END OF DUMP is printed at the terminal and control returns to CMS.

The VDUMP printed output is preceded by a header page giving the date and time of system crash

```
CP-67 SYSTEM ABEND  DATE mm dd yy  TIME hh mm ss  cause AT LOC xxx
```

The following are printed in the edited part of the dump:

- The general purpose registers, the control registers, the CSW and CAW

- The interrupt code, old and new PSW's

- General register analysis, the contents of all 16 general purpose registers followed by the name and the displacement into the routine if the register contains an address

- A listing of CP-67 nucleus routines and their entry points from CPSYM

- An edited display of the multiplexor real device control blocks. The address, name of the block and its contents are printed.

- An edited display of the real channel, control unit and device blocks, indented to indicate attachment. Should an IOTASK have been active at the time of the system dump, it is printed after the block to which it was pointing.

- Control block associated with each virtual machine. The address and contents of the UTABLE, the MVDEBLOK and the virtual channel, control unit and device blocks attached to the virtual machine are printed.

If the default (CP) is in effect for the SET DUMP command, or if CP is specified as an option, the following is printed in the dump: general-purpose registers 0-15, control registers 0-15, and all of CP nucleus and free storage. If ALL is specified as an option, the dump includes all of real core.

DISTRIBUTED CP-67 MAINTENANCE PROCEDURES

With CP-67 Release 3.2, the distributed maintenance routines are changed. The revised maintenance procedure, whose feasability has long been tested at the development site, has proven itself to be invaluable with respect to insuring system integrity. The basic philosophy of the system is to keep a release-level copy of source online while providing procedures which enable the installation to easily apply IBM-distributed fixes, installation-applied changes, and programmer modifications in any order such that if a regression is detected, appropriate action can be pursued with a minimum of effort. The annotated precedence lists employed in assembling the various modules with particular collections of updates also provide very detailed information on the load map describing the CP nucleus. Furthermore, installations planning to migrate to VM/370 can appreciate adopting a maintenance procedure which is nearly identical to that of the new system.

The facilities described here provide the capability of updating SYSIN or macro (ASP360 or COPY) files with several levels of updates and any number of PTF's. If SYSIN files are used, procedures are supplied for assembling the updated SYSIN to produce a uniquely defined text deck (object module). The text deck identified by a unique name and by some control cards indicates the origin of the updates, maclibs, and sysin. For macro library files, a copy file is produced to identify the origin of the input and any updates applied.

Procedures are provided for generating load decks from various text decks and for generating MACLIB files from various COPY and ASP360 files.

The basic structure involves a file naming convention for update and text files, a set of programs to support the processing and a set of exec procedures to process the files.

The following is a synopsis of the various procedures employed in the new maintenance scheme:

GENUPD — This distributed program converts an easy to generate change deck to a form which is valid input to the UPDATE program.

UPDATE — The CMS version of UPDATE is used to apply modifications to the various files.

SYSUPD — This EXEC procedure is used to perform a multilevel update to a specific file.

SYSASM — The SYSASM EXEC procedure invokes the SYSUPD procedure and assembles the updated source programs.

SYSLOAD This EXEC file employs the SOFPCH program and various other procedures to produce a "punched" deck comprised of a loader and several text files.

SOFPCH The SOFPCH program searches for a desired text file, punches it when found, or prints an error message.

SYSMAC This procedure uses SYSUPD, SYSMUP, and other EXEC files to generate a macro library.

SYSMUP SYSMUP adds individual macros to a temporary macro library called (TEMP) MACLIB.

SYSED This EXEC procedure assists in the editing of CP files.

MULTASM This EXEC file is used to perform several multilevel update and assembly operations.

SYSLOAD1 The SYSLOAD1 EXEC procedure creates a CP load deck comprised of multiple text files and either writes the load deck to tape or IPLs it.

TAPE80 The TAPE80 program writes 80-byte card images on a tape.

CPLIST This EXEC procedure prints files from a CP-67 LISTING tape.

CPSYS The CPSYS EXEC is used by the SYSLOAD function to produce the CP-67 load deck.

PTFX The PTFX EXEC list specifies the level of updates to be applied to the various modules in the base system.

LDRG This EXEC function punches the decks required to generate a loader.

VDUMP The VDUMP EXEC procedure invokes the FDUMP and EDITDUMP modules to produce formatted CP-67 abend dumps.

UPDATE Files

Files used to update another file are given a filetype of UPDGxxx where xxx is a unique update identifier for programmer and system use. The filename of the update file is the same name as the file to be updated. For instance, file PROGRAM SYSIN could be updated by PROGRAM UPDGJA1 and/or file PROGRAM UPDGCM6.

The update control cards in the UPDGxxx files have been extended to provide automatic serialization control. The input files are usually serialized in 1000 increments. Replacements and insertions in the files can be serialized in any sequence so long as the updated version maintains the serialization. The default update sequencing is 100. This default can be automatically invoked or specifically stated with the extended control. The extended control fields are identified with a dollar sign ($) following the normal update control fields. Sequencing starts with the serial of the first record replaced (+100) or the serial of the record after which an insertion is made (+100) if no fields are specified after the $.

The first optional field after the $ specifies the desired starting sequence. Delete control cards do not require the $ control field. The second optional field can specify the increment. The default increment will be 100.

The program used to perform the actual update operation is UPDATE. Since the UPDATE program does not recognize the $ control field, an intermediate program is used to prepare the file for the UPDATE program. This program is called GENUPD. It takes as its input UPDGxxx files with the $ control cards and produces UPDTxxx files properly sequenced for the UPDATE processing with the $ control fields deleted. If no $ control fields are specified in the update control cards, GENUPD does no processing and leaves the serialization, if any, as it appears in the following records up to the next control card. The program is invoked in the following manner:

```
GENUPD fname xxx <NO>
```

where 'fname' is the filename of the file to be processed and 'xxx' is the identifier of the update (UPDGxxx). In normal operation, GENUPD takes the first character of the identifier and places it in column 80 of the UPDTxxx file. The 'NO' is used to suppress this action if desired. The following example will show the GENUPD processing for a UPDGxxx file and assumes that the 'NO' was specified:

PROGRAM UPDGxxx input to GENUPD

```
./ R 1000
*  UPDATE 1
./ R 2000 $
*  UPDATE 2
*  UPDATE 3
./ R 3000 $ 3050
*  UPDATE 4
*  UPDATE 5
./ I 4000 $ 4020 20
*  UPDATE 6
*  UPDATE 7
```

PROGRAM UPDTxxx output from GENUPD

```
                    inserted
                    serial no.
                    (73 - 80)
./ R 1000                       - no change
*  UPDATE 1                     - no change
./ R 2000                       - no $
*  UPDATE 2         0002100     - serialized
*  UPDATE 3         0002200     - serialized
./ R 3000                       - no $
*  UPDATE 4         0003050     - serialized
*  UPDATE 5         0003150     - serialized
./ I 4000                       - no $
*  UPDATE 6         0004020     - serialized
*  UPDATE 7         0004040     - serialized
```

The filename of the UPDTxxx file is the same as the filename of an UPDGxxx file. The UPDTxxx files are in a format suitable for the UPDATE program.

TEXT Files

Text files are produced by the assembler. The filename of the text file is the same as the filename of the SYSIN file. The filetype of the completed text deck produced by these procedures is either TXTnamex, where namex represents a unique level identifier or TEXT. The value of namex is taken from an EXEC control file and corresponds to the highest level of update applied. The TEXT or TXTnamex deck is produced from an auxiliary control file containing data describing the origin of each of the fules used in the assembly process combined with the output deck from the assembler. The auxiliary file is called 'filename UPDATES' and is produced by a program called DTFILE as the updates are applied. The filename is the same as the filename of the UPDTxxx and UPDGxxx files.

CP-67 EXEC PROCEDURES

Several system EXEC procedures are employed to facilitate system update and maintenance functions. Some EXEC procedures invoke others or make use of user supplied EXEC procedures to accomplish various functions such as multi-level updating, text generation, and maclib generation. Each system EXEC procedure is described following.

CPLIST EXEC PROCEDURE

CPLIST &1

This EXEC procedure prints one or all files from a CP-67 LISTING tape. The first file on the listing tape is a CPLIST EXEC file. The CPLIST EXEC file is merely a sequential list of the names of the LISTINGS that follow. If &1 is ALL then all files are printed, otherwise the tape is searched for the desired file (for example, DISPATCH) and that file (DISPATCH LISTING) is printed.

CPSYS EXEC PROCEDURE

CPSYS

This EXEC procedure is used by the SYSLOAD function to produce the CP-67 load deck. It contains control statements and an ordered list of the nucleus components.

CPGEN EXEC PROCEDURE

CPGEN &1 &2

CPGEN is used to punch a self-loading CP-nucleus, the CP utilities, and CP loaders.

&1 = The three characters which indicate the system's printer address.

&2 = The name of the installation's real I/O configuration deck, for example, RIOCSC.

LDRG EXEC PROCEDURE

LDRG

This EXEC procedure punches the LOADER and TEXT decks required to generate a LOADER. The six files form an IPLable program that will produce an IPLable LOADER. The installation may choose to make modifications to the RELDR SYSIN for certain reasons. The assembled module (RELDR TEXT) is then used by the LDRG EXEC to produce a new LOADER.

VDUMP EXEC PROCEDURE

VDUMP

This EXEC procedure invokes the FDUMP and EDITDUMP modules that produce formatted CP-67 abend dumps from the disk dump for the user specified to receive the dump (SYSDUMP= in SYSGEN macro).

Control EXEC Files

Each user may have several control files to specify various combinations of updates and macro libraries to be used. These control files are of the EXEC type and contain records in the following format:

```
1.   &1 &2 nam00 m1  MACS maclib1 maclib2
2.   &1 &2 nam01 up1
3.   &1 &2 nam02 up2
4.   &1 &2 nam03 up3 NO
5.   &1 &2 nam04 up4 AUX
```

The fields (up1 up2 up3 up4) are update identification fields and the fields (nam00 nam01 nam02 nam03 nam04) are level identification fields.

Record 1 is the maclib record and defines the macro libraries (maclib1 maclib2...) to be used in the assembly in the order of search required. Up to five libraries may be specified. The m1 field is an identifier for the maclibs used. (NOTE: Because of restrictions in CMS's EXEC facility, macro library names are limited to a maximum of seven characters.)

Records 2, 3, and 4 are update identification records. They define the UPDGxxx files that exist for updating procedures. Record 2 defines a UPDGup1 file and records 3 and 4 define UPDGup2 and UPDGup3 updates respectively. None, some, or all of the updates may exist to be applied. Note that the 'NO' in record 4 is passed to GENUPD to suppress the character in column 80 as described earlier.

Record 5 defines an auxiliary file that specifies an auxiliary list of PTF's (APAR answers) or "final" versions of updates that are to be applied. This auxiliary file is identified as 'filename AUXup4', where 'filename' is the same as the filename of the input file. Records in the auxiliary file have the following format for PTFS to be applied:

```
PTF  A30246CA   comments
PTF  A21726CA
PTF  A07426CA
*    Any comment
```

The 'PTF' field is an identifier and the second field (e.g. A30246CA) defines a specific PTF or update to be applied. The PTF will have a 'filename A30246CA' identification, where 'filename' is the same as the filename of the file to be updated. Files which are named in "AUX" lists should ALL exist and have proper serialization since they will be applied directly without being processed by GENUPD. The filetype of a format Axxxx6CA is used to indicate an APAR answer or PTF for an APAR number xxxx. The comment field is useful to describe the function of the particular PTF. The '*' field is ignored and is used to provide additional comments on any updates or PTFS.

The updates (PTF's included) are applied in the reverse order in which they appear. In the example above the updates would be applied in the following order:

```
A07426CA
A21726CA
A30246CA
UPDGup3
UPDGup2
UPDGup1
```

There can be any number of UPDGxxx definition and AUX definition records but only one MACS record. The complete exec file can have any filename but typically has the same name as the first specified UPDGxxx control record. In the example, the exec file could be named up1 EXEC.

The underlined fields in each record mark the level identification fields. The highest level (last) update to be applied selects the name that can be used to identify updated files. In the example, if UPDGup3 was the last update applied, then the name selected would be nam03. The value for the identification is usually comprised of a combination of the update identifier (up1, up2, ...) and the maclib identifier (m1). Thus, the level identification field for record 4 might be up3m1. If no updates are applied then the nam00 field is selected to identify the TXTnam00 produced. This name can be used to uniquely identify updated files. The text files described above, for instance, can have a type of TXTup3m1. It is desirable on occasion to have entries in the user exec file that specify a level identification but no update. A record of the following format, as an example, is allowed:

```
&1 &2 nam05
```

This is because the exec file serves a double purpose and is used for loading text decks as well as updating input files. An identifier of TEXT as a name causes special handling in the SYSASM exec procedure whether or not an update is used with it. Essentially a name of TEXT is used without level identification catenation. Thus, TEXT becomes the filetype.

SYSASM EXEC PROCEDURE

The SYSASM procedure accomplishes the multi-level update function by invoking the SYSUPD procedure before assembling the desired file. To update and assemble a sysin file the SYSASM exec procedure is invoked in the following way:

```
SYSASM filename control <options>
```

where 'filename' is the name of the SYSIN file to be processed and 'control' is the name of the user exec file that contains the maclib, update, and any AUX control records. The SYSASM procedure invokes the SYSUPD exec procedure passing the values 'filename', 'SYSIN', and 'control' as well as parameters dealing with the disposition of the intermediate files. NOTE: If options are specified with SYSASM, they are NOT to be delineated by parentheses as they are with the ASSEMBLE command in CMS.

The SYSUPD procedure performs updating according to a control file, creating one record in 'filename UPDATES' for each update applied. It then returns a level identifier and a maclib list from the MACS record of the control file. The SYSASM exec procedure then reads the level identifier returned from the SYSUPD procedure. If the identifier is TEXT then that becomes the filetype of the completed text deck, otherwise the filetype is TXTxxxxx, for example TXTup3m1. The procedure then reads the maclibs list passed by SYSUPD and issues a GLOBAL command to prepare for the assembly using the specified maclibs.

The ASSEMBLE program is invoked with the specified options. If no options are specified, the defaults are PRINT NODIAG LIST DECK NORENT. The options that can be specified are LDISK DIAG NOLIST NODECK and RENT. The DTFILE program is used to construct a record for each maclib used and for the SYSIN file. Each record is placed in the auxiliary file 'filename UPDATES'. The text deck produced by the assembler is combined with the file produced by the DTFILE program and is named 'filename text', where 'filename' is that of the SYSIN file and the filetype (text) is constructed from the value returned by the SYSUPD exec procedure. All intermediate files will be erased leaving only the original SYSIN and UPDGxxx files and the newly created text file unless the 'LDISK' option is specified. The procedure will prompt the user for any missing values or for re-specification of values if the files required cannot be located. The user may enter QUIT to exit from the procedure if prompting cannot satisfy the requirements.

SYSUPD EXEC PROCEDURE

The SYSUPD procedure is used to perform a multi-level update to a specific file. The procedure is invoked in the following way:

```
SYSUPD filename filetype control
```

The SYSUPD procedure uses the 'control' exec file to perform a series of updates to the input file. The original input is not modified during the update procedure. Instead a temporary file is created from the first update which is modified by subsequent updates. Not all UPDGxxx files specified in the control file need be used, only those that exist during the SYSUPD procedure.

The SYSUPD procedure orders the updates to be applied in the correctly specified sequence and invokes the GENUPD program to perform the necessary pre-processing before invoking the UPDATE program. SYSUPD also uses the DTFILE program to construct a record for each UPDGxxx and for each record of any auxiliary (AUX) files used during the update procedure. The intermediate UPDTxxx files are erased after each update.

The update log produced by the UPDATE program (UPDLOG) has its type altered to UPDLxxx, where xxx is the same as the identification field extracted from each record of the control file. The filename is the same as the filename of the input file. Each update log is printed and then erased unless the 'LDISK' option is specified by the user with SYSASM.

The SYSUPD program returns to the invoking procedure passing the name xxxxx, where xxxxx is the level identification field of the last applied UPDGxxx or AUX file. The SYSUPD procedure also passes back to the invoking procedure a maclibs list specifying the maclibs to be used in the assembly. The completely updated file is identified as '.filename filetype' where filename and filetype are as specified in the parameters when the procedure was invoked.

SYSLOAD EXEC PROCEDURE

The SYSLOAD procedure uses SOFPCH program, and two user supplied procedures, a 'loadlist' exec and a 'control' exec, to produce a 'punched' deck comprised of a loader and several text files. The SYSLOAD procedure is invoked in the following way:

```
SYSLOAD loadlist control loader
```

The loadlist is a user supplied exec file consisting of several records of the following format:

```
&TYPE OFF
&1 &2 &3 filename <filetype>
&1 &2 &3 filename <filetype>
.
.
.
```

The 'filename' specifies the name of a text file to be punched. The text files are punched in the order specified. If a filetype is specified, a search is made for that specific file, and if it is found it is punched.

If the filetype is not given, the specified 'control' file is used to search for the highest level text file available and it is punched. See the 'SOFPCH Program' description for details of this searching function. The 'loader' parameter specifies the address (3 HEX digits) of the printer to be used to produce the LOADMAP when the complete deck is ipled.

The SYSLOAD procedure invokes the SOFPCH program and prints a confirmation or error message upon completion. Before invoking the SOFPCH program, a CLOSIO PUNCH OFF is executed to assure that the punched files appear as one deck. A CLOSIO PUNCH ON is executed upon completion. No XFER functions are used in any of the procedures; however, this capability is not precluded if the user so desires. Higher level invoking EXEC procedures can perform any desired extra control. The procedure will prompt for the required values if necessary similar to the prompting described under the SYSASM procedure.

The SOFPCH program is the program that searches for the desired text file, and punches it when found, or prints an error message. The program is invoked in the following way:

```
SOFPCH loadlist control
```

The control field specifies a user supplied control EXEC file with a filename of 'control'. This control file is of the same type and format as the one used to perform multi-level updates. Indeed, most often the file used to produce the updated and assembled text decks is the one used to load the text decks. The loadlist, as previously described, supplies the list of filenames and filetypes (if desired) to be searched for and punched.

The SOFPCH program uses the control file to search for the desired text deck in the order in which the identifiers are specified in the file. As an example, consider a control file of the following format:

```
1.   &1 &2 000m1 m1  MACS maclib1 maclib2
2.   &1 &2 up1m1 up1
3.   &1 &2 up2m1 up2
4.   &1 &2 up3m1 up3 NO
5.   &1 &2 up4m1 up4 AUX
6.   &1 &2 name
7.   &1 &2 TEXT
```

The underlined fields define the level identification used to identify the text deck.

The SOFPCH program will search for the following text decks in the order shown below.

```
1.   filename TXTup1m1
2.   filename TXTup2m1
3.   filename TXTup3m1
4.   filename TXTup4m1
5.   filename TXTname
6.   filename TEXT
7.   filename TXT000m1
```

The first file located is punched and all lower files are ignored. If the end is reached before finding a text file, the program will search for 'filename TEXT' and punch it if it exists. If the TEXT file is not found, SOFPCH will search for 'filename TXT000m1' and punch it if found.

It is quite possible to have a completed load deck comprised of different levels of text decks. The user is responsible for determining the validity of the completed deck. As an example, the following could be a valid load deck:

```
PSA       TXTup1m1
ACNTON    TXTup4m1
.
.
.
PROGINT   TEXT
.
.
.
LDT       SAVENUC
```

SYSLOAD1 EXEC PROCEDURE

The SYSLOAD1 procedure uses the SYSLOAD exec to 'punch' a deck comprised of multiple TXT files. The differences are that SYSLOAD1 issues an XFER to the user and has additional parameters. The procedure is invoked int the following manner:

```
SYSLOAD1 loadlist control loader <TAPE/IPL> mapid
```

where 'loadlist', 'control', and 'loader' are the same as for SYSLOAD. The fourth parameter MUST be either 'TAPE' or 'IPL'. If the 'TAPE' option is specified, the procedure reads the resulting card file onto the user's disk giving it the name 'TAPE LOADmapid' and then uses the TAPE80 program to write that file on a tape which, when later IPL'ed, will write a copy of CP on disk.

If the 'IPL' option is specified, the SYSLOAD1 procedure creates 'READMAP EXEC' on the user's P-disk, XFER's the printer specified when invoking the procedure to the user, and IPL's the card file created writing a copy of CP on disk. The user is prompted to IPL CMS and use READMAP to read the resulting loadmap onto his disk. READMAP EXEC will read the XFER-ed printer output onto the user's disk giving it the name 'mapid LOADMAP' which the user can then print off at his discretion.

SYSMAC EXEC PROCEDURE

The SYSMAC procedure is used to generate a maclib using the system SYSUPD and SYSMUP exec procedures and two user supplied exec procedures, a 'control' exec file used to specify levels of update, and a 'macrolist' exec file that specifies the contents of the maclib to be generated. The SYSMAC procedure is invoked in the following way:

```
SYSMAC maclib control
```

where 'maclib' specifies the name of the maclib to be generated, which is also the name of the macrolist EXEC file defining the contents of the maclib, and control specifies the name of a control EXEC file for updating procedures. The macrolist is a user supplied exec file consisting of several records of the following format:

```
&TYPE OFF
&1 &2 &3 filename
&1 &2 &3 filename
.
.
.
```

filename specifies the name of a COPY or ASP360 file to be included in the maclib.

The SYSMAC procedure invokes the SYSMUP procedure by executing the macrolist exec in the following way:

```
EXEC macrolist EXEC SYSMUP control
```

The SYSMUP procedure, described following, adds each COPY or ASP360 file to a temporary maclib file after being updated by the SYSUPD procedure.

The SYSMAC procedure then creates a 'maclib COPY' file as a record of updates and source files used to create the maclib, and adds the COPY

file to the maclib. The temporary maclib is then altered to the name of the specified maclib. A confirming message is printed. The procedure will prompt for specification of the required values if necessary similar to the prompting described under the SYSASM procedure.

SYSMUP EXEC PROCEDURE

This procedure is used to add individual macro files to a temporary maclib named (TEMP) MACLIB. The procedure is invoked from each record of the macrolist procedure in the following way:

```
SYSMUP control filename
```

The control field specifies a user supplied exec file as described under 'Control EXEC Files'.

Before adding a macro to the maclib, the SYSUPD procedure is invoked by SYSMUP to perform a multi-level update as described in the 'SYSUPD EXEC Procedure'. The values returned by the SYSUPD procedures are read and ignored by the SYSMUP procedure.

The SYSMUP procedure uses the DTFILE program to create a record of the COPY and ASP360 files used to build the maclib. A confirming message is printed at the completion of the operation.

Note: Because of the way CMS handles macro files, updated macro files are permanently changed to retain the original filename. For this reason, it is expected that COPY and ASP360 files to be processed reside on read-only disks, with the updated versions being created on the read-write disk. Updated macro files are erased after processing. Macro files that are not updated may reside on a read/write disk since they will not be erased after processing.

SYSED EXEC PROCEDURE

To assist in the editing of some of the files used in system control, an exec file named SYSED is provided. The SYSED exec procedure allows for the editing of files with a filetype of SYSIN, ASP360, COPY or UPDGxxx. The normal CMS editor (EDIT) is invoked with the SYSED procedure with serialization set to 1000 and tab settings at positions 1, 10, 16, 31, 36, 41, 46, 51, 56, and 72. The SYSED procedure is invoked in the following manner:

```
SYSED filename ident
```

where 'filename' is the desired filename and 'ident' is the xxx value for the UPDGxxx files or it is the filetype of SYSIN, ASP360, or COPY. The 'ident' should not be specified as UPDGxxx. SYSIN, ASP360, and COPY files are normally only edited during initial creation; thereafter they should be updated using the update procedures described. The UPDGxxx files are the files most commonly edited to furnish updates to the system.

Since the CMS editor does not recognize the UPDGxxx filetype, non-standard tab settings would have been set. The SYSED procedure avoid this problem by forcing the correct tab settings through use of the editor's TABSET function. The procedure will prompt for the specification of the file to be edited if it was improperly given.

In addition to the above mentioned EXECs and programs, the following are used in the process of creating and maintaining a CP system.

MULTASM EXEC PROCEDURE

The MULTASM procedure is used to perform multiple multilevel update and assembly operations. It uses MULTASM1 EXEC, a 'control' exec, and a user supplied list of files to be assembled. The MULTASM1 procedure reads the list of files and passes the names and any options specified on to the SYSASM procedure. MULTASM is invoked by typing

```
MULTASM loadlist control <options>
```

The loadlist is an EXEC file in the same format as the 'loadlist' described for the SYSLOAD procedure. The control file is also an EXEC file in the same format as that described in the section on 'Control EXEC Files'. The options available to MULTASM are the same as are available for the SYSASM procedure.

TAPE80 PROGRAM

The TAPE80 program writes 80 byte card images on a tape attached as 181. The program writes one CMS file at a time and is invoked by typing

```
TAPE80 fname ftype
```

The program is used mainly to create IPL-able system load tapes.

CP-67 LOAD DECK FORMAT

The following text files are contained in the CP load deck:

Module	
Relocating Loader (RELDR)	
SLC 000080	loader control card--with PSA TEXT
PSA	
ACCTON	
ACNTOFF	
ACNTIME	
CCWTRAN	
CFSCOM	
CFSDBG	
CFSIPL	
CFSMAIN	
CFSPRV	
CFSQRY	
CFSSET	
CFSSPL	
CFSTACH	
CHKCUACT	
CONSINT	
CONVRT	
CPCORE	

```
CPFILE
CPSTACK
CPSYM
DEDICATE
DIAGDSK
DIAL
DISPATCH
DSKDUMP
EXTEND
FREE
IOINT
IOERROR
LINK
LOGON
LOGFILES
MRIOEXEC
MVIOEXEC
PACK
PAGEGET
PAGTR
PAGTRANS
PRIVLGED
PROGINT
QUEVIO
RDCONS
RDSCAN
RECFREE
RESINT
RIOCSC
SCANUNIT
SCHEDULE
STCONS
SYSDESCR
TMPSPACE
TRACER
UNSTIO
UNTRANS
USERLKP
USEROFF
VIOEXEC
VSERCH
WRTCONS
ICS CPEND       loader control card--with IPL TEXT
SLC 020000      loader control card--with IPL TEXT
IPL
SLC 020800      loader control card--with CPLOCS TEXT
CPLOCS
SLC 021000      loader control card--with SYSTEM TEXT
SYSTEM
SLC 022000      loader control card--with CHKPT TEXT
CHKPT
SLC 023000      loader control card--with CPINIT TEXT
CPINIT
SLC 025000      loader control card--with SAVECP TEXT
SAVECP
LDT SAVENUC     loader control card
```

CP-67 MAINTENANCE PROCEDURE

When corrections have to be made to a CP-67 module, a PTF is made available to the installation through the normal channels. The PTF is in a form suitable for a CMS OFFLINE READ or TAPE LOAD function depending upon the distribution medium. A description of the procedures to apply PTFs is given below.

1. Load the PTF onto the CP maintenance P-disk by use of the CMS OFFLINE READ or TAPE LOAD commands. All PTFs distributed by the development group have the naming convention of module-name as the filename and xxxxx6CA as the filetype where xxxxx is the original APAR number the PTF fixes. One PTF may involve changes to several modules. For example, PTF number A32486CA may create (through loading onto the P-disk) the following files:

```
DISPATCH  A32486CA
PAGTRANS  A32486CA
UNSTIO    A32486CA
CALL      A32486CA
```

2. Insert a PTF number entry in the auxiliary PTF files (AUXPTFS files) for each module for which an update was supplied. This entry must appear as the first entry in the AUXPTFS file as the PTFs are applied in the inverse order as they appear in the file. In the example used above four PTF auxiliary files must be modified (or created if there have been no prior PTFs updating this module). The resultant files might look like the following:

```
DISPATCH  AUXPTFS
     PTF A32486CA
     PTF A31206CA

PAGTRANS  AUXPTFS
     PTF A32486CA
     PTF A21426CA
     PTF A20316CA

UNSTIO  AUXPTFS
     PTF A32486CA

CALL      AUXPTFS
     PTF A32486CA
```

   Note that in this example the modules DISPATCH and PAGTRANS had one and two prior PTFs respectively, and that the module UNSTIO, and ASP360 file CALL had had no prior PTFs applied. Note also that the new PTF entry appears as the first entry in the DISPATCH and PAGTRANS AUXPTFS files.

3. Determine if any of the PTF updates changes an ASP360 or COPY file. If a PTF is supplied which changes either of these filetypes a new MACUP maclib must be generated. The following steps should be executed to generate a new MACUP maclib with an updated version of the ASP360 or COPY file.

   a. Add the ASP360 or COPY file name to the MACUP EXEC file on the CP maintenance P-disk if it is not already in the file.

   b. Use the SYSMAC EXEC utility to regenerate the MACUP maclib. The command used to regenerate the MACUP maclib file is

```
SYSMAC MACUP PTFX
```

4. Use the SYSASM EXEC utility to reassemble all SYSIN files changed by the PTF. The text decks that are created will have a filetype of TXTPTFS indicating that at least one PTF has been applied to this module. Using the above example, the modules DISPATCH, PAGTRANS, and UNSTIO would be assembled with the following commands:

```
SYSASM DISPATCH PTFX
SYSASM PAGTRANS PTFX
SYSASM UNSTIO PTFX
```

The resultant text decks will be named

```
DISPATCH TXTPTFS
PAGTRANS TXTPTFS
UNSTIO TXTPTFS
```

5. Use the SYSLOAD or SYSLOAD1 utility to construct a new CP nucleus. It is suggested that the nucleus be written as the first file of a tape followed by a dump of the maintenance disk used to build the CP system. This procedure insures that a previous system could be restored to the CP system disk by IPLing the tape on the bare machine in case any errors are introduced in the new system. In addition it provides a backup of the CP maintenance disk.

All SYSIN, updates, and programs needed to generate a CP-67 system can be contained in a 50 cylinder (2314) area. In addition, all the necessary text files required to produce the "floor" system can be placed on the same disk. The PTFX EXEC allows a user update level to be applied to the floor system. User update files must have a filetype of UPDGUS in order to be applied via the distributed PTFX EXEC control file.

In creating an IPL-able system tape or deck for CP, the "loadlist" that is used is CPSYS EXEC. This file contains the filenames of all the required modules for CP in the proper order.

CP-67 TRACE FACILITIES

The SET TRACE console function initiates and terminates the following tracing functions: user interrupts, instructions including interrupts and successful branches, I/O operations issued to a selector or dedicated device, and virtual and real CCW's involved in a selector or dedicated device I/O operation. Output can be directed to either the virtual console or printer, or both. Tracing of successful branches and/or all instructions involves considerable overhead and should, therefore, be used wisely, especially when output only to the virtual printer. (Note: The TRACE options BR (to trace all successful branches) and ALL (to trace all instructions) can be issued only by a class A or B user.) The virtual printer will enter console function mode after any output message directed to it; it is necessary to issue a 'begin' to continue execution. When tracing on the virtual printer, the printer must be closed (i.e., 'close 00e') to get output after turning trace off.

Note: The TRACE(6) option must be chosen at system generation time in order to issue SET TRACE.

SET ADSTOP enables the user to specify a virtual instruction address at which execution is to be stopped. Only an instruction execution address stop can be detected--not a storage reference. The

user is placed in console function mode; a 'begin' will resume execution from the address stop location. The address stop function is removed by a) reaching the address stop location, b) 'ipl' or 'reset' of the virtual machine, or c) the set adstop off command. Note: the user must have the TRACE(6) option at system generation time in order to issue SET ADSTOP (see the CP-67/CMS User's Guide and CP-67 Operator's Guide for further information).

DUMP/RESTORE UTILITY

To create a CMS System disk, place the distribution tape on a tape drive, address 0CUU. Clear core and IPL the tape by pressing the LOAD button. When the wait light remains on, and the system and load lights go off, hit REQUEST on the online console. DUMP/RESTORE is given control.

CMS may be restored under CP-67, if CP already exists. Merely, attach the CMS System Disk tape to a virtual machine and issue IPL cuu. Produce an ATTENTION interrupt.

The console address is dynamically set. If replies are incorrect while the DUMP/RESTORE is executing, DMPRST reinitializes itself and repeats Question 1. If an I/O error is sensed on the disk or tape, it tries to rectify the error condition. If successful, it continues. If not, it prints out a message and terminates.

Termination, either after a successful operation or because of an error, places the CPU in the wait state.

Q1: TASK? ("DUMP" , "REST"):
DUMP--disk to tape
RESTORE--tape to disk

Q2: DEVICE TYPE? ("2311" , "2314"):
specify the disk device type

Q3: TAPE ADDRESS? ("0CUU"):
self-explanatory. Must enter a four digit hexadecimal reply (for example, 0181, 0275)

Q4: DISK ADDRESS? ("0CUU"):
self-explanatory. The reply must be four hexadecimal digits (for example, 0191,0290).

Q5: REWIND TAPE? ("YES" , "NO"):
This applies to before and after the operation. Obviously, since the tape after IPL is positioned at its second file, the answer must be NO.

Q6: NUMBER OF CYLINDERS? ("0NNN"):
self-explanatory. The reply must be four decimal digits (for example, 0203, 0100). For restoring the distribution tape, specify 203 cylinders for a 2311 or 54 cylinders on a 2314.

Q7: STARTING CYLINDER? ("0NNN"):
self-explanatory. (For the system disk, reply: "0000".)

Q8: BEGINNING OF TAPE? ("YES" , "NO"):
The answer determines if a tape mark is to be written ahead of the file. If YES, a tape mark is written; if NO, the EOF at the end of the previous file serves as the tape mark for the beginning of this file. (When restoring the distribution tape, Q8 is not asked.)

The DUMP/RESTORE operation begins after the last question is answered. If either device is not ready, the program waits for its ready state. When restoring, the disk need not be formatted or DASDI'ed. DMPRST formats the disk as it is being restored.

Upon encountering an EOF, satisfying the amount of cylinders specified in Q6, or executing a SEEK beyond the disk limits, the operation terminates with the message

```
'DUMP/RESTORE MOVED nnn CYLINDERS
THERE WERE nnn RECOVERABLE TAPE ERRORS.'
```

```
TASK? ("DUMP","REST"): restore
DEVICE TYPE? ("2311","2314"): 2314
TAPE ADDRESS? ("0CUU"): 0181
DISK ADDRESS? ("0CUU"): 0190
REWIND TAPE? ("YES","NO"): no
NUMBER OF CYLINDERS? ("0NNN"): 0054
STARTING CYLINDER? ("0NNN"): 0000
BEGINNING OF TAPE? ("YES","NO"): no
DUMP/RESTORE MOVED 054 CYLINDERS.
THERE WERE 001 RECOVERABLE TAPE ERRORS.
```

Figure 1. Restore Procedure for CMS Distribution Tape

CMS SYSTEM DISK

Once restored, the CMS System disk assumes the following appearance:

Block Information

0-2 IPL pointers, CCWs, etc.

3 label = CMS190

4 Master File Directory

5-7979

1. Nucleus Text decks in one file called NUCLEUS 3.1, in which each individual text deck is preceded by an OFFLINE READ filename TEXT card. The nucleus text decks are also present individually.
2. Disk Command MODULE files.
3. Disk Command TEXT files.
4. EXEC Procedures
5. MACLIB and TXTLIB files

7980-8100 IPL'able Nucleus (0-12000x and loader tables).

The nucleus routine, NUCON, contains a device table. It has been initialized as follows:

```
CONSOLE (CON1)   =   0009
CREADR (RDR1)    =   000C
PRINTER (PRN1)   =   000E
CPUNCH (PCH1)    =   000D
S-Disk (DSK1)    =   0190
P-Disk (DSK2)    =   0191
T-Disk (DSK3)    =   0192
A-Disk (DSK4)    =   0000
B-Disk (DSK5)    =   0000
C-Disk (DSK6)    =   019C
TAP1 (TAP1)      =   0180
TAP2 (TAP2)      =   0181
CRT1 (CRT1)      =   0106
```

These values are the standard CMS machine device addresses. If it is necessary to redefine these device addresses when running CMS on the physical machine, refer to "Bare Machine-CMS".

The second file on the Source Disk Tape contains an alphabetized TAPE DUMP of the 'standard' CMS System Disk. This is included so that existing users may selectively acquire TEXT and MODULES by using their current system.

BARE MACHINE-CMS

Having RESTORE'd the CMS System disk onto either a 2311 or a 2314 disk pack, CMS may now be executed. Set the address of the System pack into the LOAD unit dials and depress the LOAD button. When the CPU enters the wait state, hit the REQUEST button on the operator's console. The INITB67 routine is given control. Note that in looking for the 1052 online console on the real machine, CMS looks for device 009 or 01F before waiting on first interrupt denoting the console; on the virtual machine CMS only looks for device 009 as the console and does not wait for an interrupt.

Since the physical device addresses of your particular installation might not be identical to CMS standard virtual device addresses, CMS allows redefinition of the device table, so that bare machine operation of CMS is possible.

```
Q1:  REDEFINE DEVICE ADDRESSES? ("YES","NO") . . .
     NO--CMS assumes the standard addresses
     YES--address modification is made
C1:  ALL DEVICE ADDRESSES MUST BE ENTERED AS 4 HEXADECIMAL DIGITS:
     "0cuu".
Q3:  OPERATOR'S TERMINAL . . .
                .
                .
                .
```

After redefining addresses, CMS types the following identification message:

```
CMS .. VERSION n   LEVEL  m
```

where n is the version and m the modification number.

CMS SOURCE DISK

On the CMS Source Disk tape, which is one of the three Basic Program Material tapes, is a TAPE DUMP of the file SOURCE EXEC which is an alphabetized log of the source files distributed. After restoring the CMS System disk, issue TAPE LOAD to obtain the SOURCE.1 EXEC file; a second TAPE LOAD command will load the Source files (SYSIN files for all of CMS, excluding OS/360 language processors and libraries, plus EXEC files). However, TAPE LOAD is a CMS command. Therefore, the loading must be executed under CMS control. Once CMS is operational, obtain a P-disk of 203 2311 cylinders or 54 2314 cylinders, format the disk (FORMAT P ALL), and load the tape (TAPE LOAD). The tape must be mounted on symbolic device TAP2.

Files 3 through 9 on the Basic Source Tape contain an alphabetized TAPE DUMP of all OS language processor object decks which are contained in CMS Version 3 Level 1: ASSEMBLER-F, Rel. 20, Component IEU; FORTRAN-G, Rel. 20, Component IEY; FORTRAN-G, Rel. 20, Component IHC; PL/I, Rel. 20, Component IEM; PL/I, Rel. 20, Component IHE.

ALTERING THE CMS SYSTEM DISK

NUCLEUS

After restoring the CMS System Disk, IPL 190. Use the CMS System Disk as a P-Disk by issuing LOGIN 190 P.

The LOADER deck supplied for Nucleus Generation is used to IPL the nucleus under CP (that is, it recognizes the standard CMS virtual device addresses, PRINTER = X'00E'). If the nucleus is to be IPL'ed on incompatible device addresses, the loader deck (first deck) must be replaced with the appropriately addressed loader, located on the CP distribution tape. Several loaders for different printer addresses are available from the CP distribution tape.

The CMS nucleus may be created by using the two EXEC procedures on the system disk--PUNCH and NUCLEUS. Issue the command NUCLEUS EXEC PUNCH userid/OFF. Each component text deck of the nucleus is punched. If a userid was specified, the deck is XFER'ed to that userid. If OFF was specified, the physical card deck is punched.

IPL'ABLE SYSTEM DISK

To rewrite the IPL'able nucleus onto the CMS System disk, place the nucleus in the card reader and press the LOAD button. Under CP, read the nucleus deck into the CP card reader with the appropriate ID card, or use the EXEC procedure NUCLEUS and XFER the deck into the card reader, then type the CP command IPL 00C. The nucleus is loaded into core, linkages and V-cons are resolved, and a load map is produced. An IPL'able core-image copy of the nucleus (locations 0-13000) may now be written onto disk.

After the IPL sequence has completed, the INITIPL routine is given control. It expects the standard console (terminal) address (009) or (01F) on the real machine. If it is other than 009 or 01F, the wait state is entered, and INITIPL accepts an ATTENTION interrupt in order to define the console address. The following information is requested:

"REWRITE NUCLEUS? (YES,NO)":
YES--a new IPL'able copy of the CMS nucleus is to be written onto disk
NO --no writing takes place. Control returns to INIT (CMS initialization routine)

"SYSTEM DISK? (0CUU,SYS)":
The S-Disk slot in the NUCON Device Table must contain the address of the device to be used as the CMS System Disk. If the reply is "0CUU", the S-disk address will be set to "0CUU". If "SYS" is entered, the default value (X'190') will be used.

"IPL DEVICE? (0CUU,SYS)":
A core-image copy is written onto device 0CUU (if specified) or

onto SYS, which is the System disk address specified in the nucleus device table.

"STARTING CYLINDER? (0NNN,SYS)":
The core-image copy is placed at cylinder 0NNN--decimal representation--or on the SYS cylinder. The SYS cylinder is

	cc	hh	rr	block
2314:	53	04	00	7980
2311:	199	05	00	7980

"VERSION IDENTIFICATION (30 characters)":
Thirty bytes of information, including blanks, are used as the version title and are printed out when IPL'ing the CMS nucleus.

"INSTALLATION HEADING...(40 CHARACTERS)":
Forty bytes of information, including blanks, are used as an installation heading at the top of printer output.

An IPL'able copy of the CMS nucleus (locations 0-13000) is written onto the specified disk. A numerically ordered load map showing all deck names and reference points for the nucleus is written to the system printer device. There are three unresolved symbols at the end of the load map, as follows:

BATCH, SYSCTL--external references for the Batch Nucleus.

CNTRL--unresolved simulation routine.

SAVING CMS UNDER CP

In place of IPL'ing a device each time CMS is to be invoked, CP allows the system programmer to SAVE a copy of the CMS nucleus, so that a user may invoke CMS by issuing the CP command IPL CMS.

To save CMS, IPL the CMS System disk on the bare machine (that is, not under CP). Cause an ATTENTION interrupt on the console.

The following is typed:

Q1: REDEFINE DEVICE ADDRESSES? ("YES","NO")... (reply NO). This question is asked when running CMS on the bare machine to allow the user to dynamically change the standard device addresses.

Q2: WILL THE CP-SAVE FUNCTION FOR CMS BE EXECUTED? ("YES","NO")... (reply YES).

C1: ALL DEVICE ADDRESSES MUST BE ENTERED AS 4 HEXADECIMAL DIGITS: "0CUU".

Q3: ENTER THE "REAL" address of the CMS SYSTEM DISK... reply "0CUU", where CUU is the real online address of the CMS system disk--as opposed to virtual address 190.

Q4: SET THE "ADDRESS COMPARE" SWITCH FOR LOC X'88'; REPLY "GO" After setting the instruction counter switches to reflect hexadecimal location '88', and setting the "address compare" key "on", reply "GO".

C2: WHEN "MANUAL" STATE IS ENTERED, IPL THE CP "SAVESYS" UTILITY DECK FROM THE CARD READER

When the CPU stops at location X'88', place the CP SAVESYS utility deck with loader and control card into the real card reader, reset the load address on the CPU console, and IPL the card reader. The message NORMAL TERMINATION OF SAVE FUNCTION is typed.

CMS may also be saved on a virtual machine. However, it is advised that only a system programmer well-versed in the concepts of CP/CMS attempt the following:

- Logon as any virtual user. While still in the CP environment issue these two commands:
  1. DETACH 0FF
     First the software timer device must be detached from the virtual machine because CMS must 'think' that it is executing on the real machine.
  2. LINK sysuser cuu 230 w
     (a password probably is also necessary). Write status must be acquired for the CP system disk (cuu) which will contain the 'saved' CMS nucleus. It must be virtually attached at address X'230', which matches the address specified in the data card for the SAVESYS program.
  3. Verify that an IPLable copy of the CP SAVESYS program is in the virtual card reader. A copy of that program resides on the CMS S-Disk as SAVECMS LOAD P1 and can be obtained when the S-Disk is accessed as a P-Disk.

- IPL 190 This loads into core the CMS nucleus. The machine will enter wait state.

- Produce an attention interrupt to define the console address. Answer questions 1 and 2 as above. Question 3 must be answered with the virtual address of the CMS system disk, 190.

- Now the tricky part: for question 4, reply as follows--
  (Note: The following procedure can be bypassed by using CP's SET ADSTOP xxxxx command, where xxxxx equals 88, if the &TRACE option is selected for that virtual machine.)

  GO (type a blank,then hit ATTN, entering CP environment)

  When CP environment is entered, issue the following commands:

  1. d 88
     88 = 070058C0
  2. ST 88 47F00088
  3. begin

- CMS will type C2 as above. Note, no CPU panel manipulation is necessary or for that matter possible. The hard address stop is achieved by causing a single instruction loop at location X'88'. Wait about 10 real seconds.

- Hit ATTN and re-enter CP environment.

  1. d PSW
     PSW = xx xx xxxx xx000088

     (if the PSW is another value, enter begin and wait another 10 seconds before hitting ATTN)

  2. ST 88 070058C0

  3. IPLSAVE 00C

     NORMAL TERMINATION OF SAVE FUNCTION

- Logout of CP and login again to reacquire the timer device, 0FF, and IPL CMS. Note the form of the IPLSAVE console function issued in step 3. IPLSAVE preserves virtual memory; whereas, IPL clears virtual memory prior to executing.

CMS EXEC PROCEDURES

The following EXEC procedures can be used in the building and maintaining of the CMS system.

ASMGEND, FORGEND, PLIGEND

ASMGEND/FORGEND/PLIGEND

Used to create overlay structures for the OS language processors, using the object decks and producing modules. These -GEND EXEC procedures are used only when creating the overlay module structure. Do not try to use them when the text decks of the processor components are not present. For re-genmoding the interface modules, use CMSGEND EXEC. No arguments are passed to these -GEND procedures. ASMGEND, FORGEND, and PLIGEND reside on the source disk.

CMSGEND

CMSGEND &1

Creates any or all CMS disk-resident modules. The name of the disk module to be created must be passed to CMSGEND (for example, to create the TAPE module, issue the command CMSGEND TAPE). CMSGEND resides on the CMS system disk and can only be accessed when the S-disk is used as the P-disk. An example of CMSGEND EXEC follows:

```
CMSGEND &1 &2
```

where &1 = module name (= START card name)

&2 = TEXT name (if &2 is present)

If the START card name and TEXT name are the same, the command is CMSGEND &1 00. If the START card name and TEXT deck name are different, the command line is:

CMSGEND (START name) (TEXT name)

Example:

CMSGEND COMBINE 00

will produce a COMBINE module with a mode of 'P2'.

CMSPTF

CMSPTF &1 &2

Will permanently apply a 'PTF' to a source deck, print a LISTING to the offline printer, and punch off a TEXT deck. The command line is: CMSPTF &1 &2 , where &1=filename of SYSIN and &2=filetype of PTF deck, i.e. 'AxxxxyCA'.

CMSSORT

CMSSORT 'fn' 'ft' P

will produce a sorted file of the P-disk.

Will list and sort in alphabetical order into a file the names of the files on the disk for which the MODE letter was given. The file will be printed on the offline printer.

CMSUPASM

CMSUPASM &1 &2

where &1 = SYSIN name, and &2 = UPDATE name.

CMSUPASM is used to update a source deck and get a copy of the new object deck in a one-step procedure. This procedure will take the original source deck and the update deck and will create the new .SOURCE deck, assemble it producing a .TEXT deck, print a listing of the update changes, print an assembly listing on the offline printer, offline punch the .TEXT deck, and erase the .SOURCE and .TEXT decks.

The options that can be used with CMSUPASM are: SP, NP, NPR, NDG, BLIP, and ORG, where SP = the .TEXT deck will be saved on the P-disk, NP = no offline punch of the .TEXT deck, NPR = no assembly listing on the offline printer, NDG = no online diagnostics, BLIP = use BLIP (BLIP character '?'), ORG = alter .XXX TEXT to original text name. Consider the following example of CMSUPASM EXEC:

```
SOURCE DECK : TEST SYSIN
UPDATE DECK : TEST UPDATE
```

where the command is: CMSUPASM TEST SP, and the file results are: TEST SYSIN (original source deck), TEST UPDATE (update deck), and .TEST TEXT (new updated object deck). TEST UPDLOG and the Assembly listing are printed on the offline printer.

NUCLEUS/PUNCH

NUCLEUS EXEC PUNCH (userid, OFF)

Used to create a physical CMS nucleus deck by punching each individual

TEXT deck. Issue the command: NUCLEUS EXEC PUNCH userid, where userid is the identity of the virtual machine to which the nucleus deck is XFER'd. PUNCH EXEC will first search for an updated TEXT deck; i.e., when looking for INTSVC TEXT, it will first search for .INTSVC TEXT, then INTSVC TEXT. This will utilize a newly updated deck.

MAINTAINING THE CMS SYSTEM DISK

To add or change commands in the nucleus or on the system disk, proceed as described below.

CMS MAINTENANCE PROCEDURES

To update the CMS system the procedure described below must be applied.

When a PTF is received, the user should apply it to his SYSIN deck. The PTF is an update deck identified as "filename AxxxxyCA", where filename is the name of the routine to which the update is to be applied and AxxxxyCA is the filetype. xxxx is the APAR number, and y is the APAR code that the PTF corrects.

Example:

```
V3 deck     EDIT SYSIN
Update      EDIT AxxxxyCA
```

The command to issue is

```
UPDATE EDIT SYSIN EDIT AxxxxyCA (seq8 inc P)
```

The "seq8" allows sequencing of up to eight digits. The "inc" includes the sequence number of the update card into the new updated sysin file. The P causes a newly updated SYSIN file to be created.

The updated file created is EDIT SYSIN.

Note: The original EDIT SYSIN is replaced with the updated copy.

CMSPTF EXEC

The updating process described above can be executed with the aid of a CMS EXEC file called CMSPTF EXEC.

This EXEC procedure takes the original SOURCE deck and the PTF deck and creates the new SOURCE deck, assembles it producing a TEXT deck, prints an UPDATE LOG of the update changes, and an assembly LISTING onto the offline printer, and offline punches the TEXT deck. The new SYSIN and TEXT decks remain on the user's read/write disk.

This enables the user to update and get a copy of the new object deck in a one-step procedure.

CMSPTF Arguments and Options

The format of the CMSPTF command is as follows:

CMSPTF filename filetype options

where filename is the name of the SYSIN file to which the APAR is to be applied.
filetype is the APAR identification

Options Defined:

NP = No offline punch of the TEXT deck.

NPR = No assembly listing on the offline printer.

NDG = No diagnostic messages on the terminal.

BLIP = Use BLIP, (BLIP Char. ' ? ')

An example of CMSPTF EXEC follows:

```
SOURCE DECK : EDIT SYSIN
UPDATE DECK ; EDIT AxxxxyCA
```

Command:

CMSPTF EDIT AxxxxyCA

Results:

The status of the files is as follows:

EDIT SYSIN (updated source deck), EDIT AxxxxyCA (original PTF update deck), EDIT TEXT (new updated object deck); EDIT UPDLOG, and the Assembly listing are printed on the offline printer.

CHANGING OR ADDING NUCLEUS RESIDENT COMMANDS

Assemble the source of the new or changed command to obtain a loadable TEXT deck (object deck).

If a new command, update and assemble the system routine FUNCTAB, making an entry into FUNCTAB for the new command.

If a new command, replace the old FUNCTAB text deck within the nucleus deck with the newly assembled copy. Also add the text deck of the new command to the nucleus.

If changing an existing command, replace the old TEXT deck with the newly assembled copy.

If using the NUCLEUS/PUNCH procedure of creating a CMS nucleus, the newly updated TEXT decks need only be placed with the remaining TEXT decks. They will automatically be incorporated into the nucleus when it is constructed.

Rewrite the CMS nucleus onto the system disk.

CHANGING DISK-RESIDENT COMMANDS

As above, obtain the object deck of the new command.

Using the system disk as a read/write P-disk, offline read the text deck of the new command.

The CMSGEND EXEC procedure may be used to generate any CMS disk-resident command distributed with the system. Details for generating each command can be determined by examining the CMSGEND EXEC file.

MANAGEMENT OF TEXT FILES FOR LANGUAGE PROCESSORS

On the Source Disk tape are the TEXT decks of all the processors distributed with the system. The following files are included in this tape, in the order indicated below and with the specified disk space requirements for loading:

ASSEMBLER	211	CMS records
FORTRAN Compiler	166	CMS records
FORTRAN Library	213	CMS records
PLI Compiler	1999	CMS records
PLI Library	709	CMS records

MODIFICATIONS TO PID TEXT DECKS FOR CMS

For Assembler, Release 20, all IEUF7* TEXT decks are represented by one IEUF7 TEXT deck, and all IEUF8* TEXT decks by one IEUF8 deck, as below:

CMS	Original PID TEXT decks
IEUF7 =	IEUF7I
	IEUF7E
	IEUF7D
	IEUF7X
	IEUF7N
	IEUF7V
	IEUF7G
	IEUF7C
IEUF8 =	IEUF8I
	IEUF8C
	IEUF8M
	IEUF8A
	IEUF8P
	IEUF8D
	IEUF8S
	IEUF8V
	IEUF8L
	IEUF8N

For FORTRAN, Release 20, the following decks have had their names changed:

PID Released Name	CMS Renamed Deck
IEYFORT	IEYFORT0
IEYROL	IEYROL3
IEYINT	IEYINT4

The CMS distributed IEYFORT TEXT deck contains the following PID released text decks:

```
      CMS Name -->  IEYFORT0 IEYFORT1 IEYFORT2 IEYROL3 IEYINT4
IEYFORT = ------
      PID Name -->  IEYFORT  IEYFORT1 IEYFORT2 IEYROL  IEYINT
```

For PL/I Release 20, the following decks have had REP cards inserted:

PID Released Name	Purpose
IEMAB	To cause compiler to use dictionary blocksize as SYSUT1 file blocksize.
IEMAS	To provide an OS/360 save area for the %INCLUDE function.

Code has been added to module IEMAD to get the correct length of compiler modules for the DUMP option.

C-DISK AS A READ-ONLY EXTENSION OF S-DISK

The use of an extra disk as a read-only extension of the standard S-Disk is generally useful for one of several reasons:

1. If the system customarily has a large number of users, it can provide a more efficient utilization of system resources to place some of the system disk programs and libraries on one disk pack, and others on another pack. If some of the programs and/or libraries currently on the S-Disk are placed on another disk instead, this can be accomplished.

2. If a system disk gets very full from addition of new programs, it may be advisable to put new added programs on an extra disk instead of adding more cylinders to the S-Disk.

3. It may be that certain new or restricted programs are to be made available to a limited number of users. In this case, such programs could be put on the extra disk, which would only be in the directory of those permitted to use such programs.

CMS has the capability to provide such an extra disk as a read-only extension of the S-Disk. This extra disk is the C-Disk, and has a default disk-address (in the NUCON table) of 019C. Be sure to place the C-Disk as a read-only disk in the virtual machine description in the CP directory, with virtual address 19C, for all users who are to have the C-Disk available to them.

If the C-Disk is attached and ready at CMS Initialization time, while the version identification message is being typed, CMS automatically

logs in the P2 files from the C-Disk, as a read-only extension of the S-Disk, in a manner similar to the SYSGEN function for obtaining the SSTAT table from the S-Disk.

The C-Disk is then a read-only extension of the S-Disk, transparent to most CMS commands. STAT C gives the disk-statistics on the C-Disk (as STAT S does the standard S-Disk), and LISTF filename filetype C lists the C-Disk files just as LISTF filename filetype S does the standard S-Disk files.

CMS CONSIDERATIONS

FORMATTING THE P-DISK

Once the CMS System disk is restored and all device addresses are assured correct in matching either the bare machine configuration or the CP allocation directory, issue an IPL to the System disk.

Initially, each user must issue as his first command FORMAT P ALL. This command formats his P-Disk for CMS usage. If the disk is not formatted, I/O errors occur when accessing the P-disk.

Note: All disks that are to be accessed by users must be formatted in the same manner as above.

CMS VERSION 3-VERSION 2 COMPATIBILITY

The disk formats for Version 2 and Version 3 are compatible; however, the internal core tables are incompatible between these two versions of CMS.

Do not use old commands from Version 2 Levels 0 and 1 which manipulate files with Version 3 (such as LOGIN, LISTF, OFFLINE, ERASE, TAPE, DISK, FORMAT, and START) as they do not work.

If a previous version of CMS is to be examined, log it in as a C-disk so that it will be searched last and its routines will not be used in place of the system routines.

OBTAINING A NUCLEUS LOAD MAP

A load map of the nucleus, giving the name and core location of each nucleus routine, may be obtained by

```
IPL 190,  (CMS system disk)

login on any formatted P-disk

MAPPRT C OFF    (see other options for this
                 command in CP-67/CMS User's Guide)
```

The file CMS-NUC ALPHANUM P1 is created on the P-disk and is also printed offline.

P-DISK CONSIDERATIONS FOR THE BARE MACHINE

When operating on the bare machine, the P-disk is used from real cylinder 0. Therefore, if a user wishes to use his own files on the bare machine, his files must be physically located on real cylinder 0.

The user must also be aware of his P-disk size when he is operating on the bare machine. CMS is not aware of the user's virtual boundaries and assumes a full physical disk pack; it will allow the user to format all 203 cylinders of the disk pack.

COMMAND ABBREVIATIONS

When a CMS command is issued from the terminal or from a routine, several tables are searched for the command name. If no command is defined for the name, CMS assumes that a command abbreviation was used. The nucleus-resident routine ABBREV is then searched to verify and find the full-name equivalent of the command. Command abbreviation is defined as the minimum number of characters necessary to recognize the full command name.

If standard system abbreviations are not to be allowed in CMS, the ABBREV TEXT deck should be removed from the nucleus deck.

To change, add, or delete an abbreviation, the ABBREV routine must be reassembled, altering the abbreviation table as desired. The table is constructed using the macro ABRV as follows:

```
ABRV name, number
```

where name is the full name of the command, and number is the minimum number of characters which are accepted as an abbreviation.

The macro must be used for each desired command abbreviation. For example:

```
ABRV   OFFLINE, 1
```

signifies that the acceptable abbreviation for the command OFFLINE is O (also, OF, OFF, OFFL, ...).

```
ABRV   ALTER, 2
```

states that the abbreviation for ALTER is AL (also, ALT, ALTE, ALTER).

INSTALLATION OF THE CMS BATCH MONITOR

Using the CMS System disk as a P-disk, obtain a copy of the Batch Monitor Nucleus, BATNUC 3.1, by issuing the command OFFLINE PUNCH BATNUC 3.1. A copy of the Batch nucleus may also be placed in the virtual card reader of USERA by issuing the command: NUCLEUS EXEC PUNCH USERA BATCH

Set up a disk area that is used only by the virtual batch machine. This disk area, address cuu, contains the IPL'able copy of Batch nucleus and need be only two-2314 cylinders, or four-2311 cylinders. Also, set up a P-disk, 191, for the Batch machine.

Write an IPL'able copy of the Batch nucleus onto disk cuu in the same manner as the CMS nucleus is reloaded onto the System disk.

After writing the Batch nucleus onto cuu, the message READY is typed. The Batch Nucleus disk, cuu, must be IPL'ed before executing any Batch Job Streams.

To run a CMS Batch job stream, first it is necessary to IPL CUU. The Batch nucleus may also be SAVE'd by CP, using the name BATCH. If the name BATCH is undesirable, a different name may be used by changing the CP routine SYSTEM, which contains the names and descriptions of all IPL'able named systems. Batch nucleus has set a time limit per job (five minutes) and an output limit per job (5000 pointer lines). These limits may be modified by reassembling the routine BATJCB and placing the new copy into the BATCH nucleus.

SAMPLE PROBLEM

A sample problem is distributed as File 3 on the Source Disk tape. It exists in TAPE DUMP form.

To load the sample problem onto the P-disk, position the tape after the second tape mark and load it from tape. This is done by issuing

```
TAPE REWIND
TAPE SKIP 2
TAPE LOAD
```

Then the program is ready to be compiled and executed.

The complete procedure for logging in, IPL'ing 190 for CMS, FORMAT'ing the P-disk, loading the FORTRAN source deck on the P-disk, and then compiling and executing the sample problem as shown in Figure 2. System responses are printed in uppercase; user-entered information is typed in lowercase. Figure 3 contains a listing of the file SAMPLE FORTRAN.

```
login user1
ENTER PASSWORD:

READY AT 15.30.22 ON 2/15/73

DEV 181 ATTACHED

ipl 190
CMS .. VERSION 3   LEVEL  2

format p all
** "FORMAT P" WILL ERASE ALL YOUR P-DISK (191) FILES **
**DO YOU WISH TO CONTINUE? ENTER "YES" OR "NO":
yes
ENTER 6-BYTE LABEL (IF WANTED), OR NULL LINE (IF NOT):

FORMATTING P-DISK (2314)...
P(191): 007 CYL
R; T=0.10/2.20 15.33.55

tape rewind
R; T=0.02/0.05 15.34.15

tape skip 2
R; T=0.20/0.57 15.35.00

tape load
LOADING:
SAMPLE FORTRAN P1 LOADED
R; T=0.15/0.32 15.35.25

fortran sample
R; T=0.58/0.92 15.35.59

$ sample
EXECUTION BEGINS. . .
    PLEASE TYPE IN THE NUMBER OF VALUES TO AVERAGE
03
   FIRST NUMBER? (FORMAT is F10.4)
45.6
   NEXT NUMBER?
56.7
  NEXT NUMBER?
49.09
                  AVERAGE =  50.4633
     PLEASE TYPE IN THE NUMBER OF VALUES TO AVERAGE
00
     *** TEST COMPLETE *** WELCOME TO CP-67/CMS!
R; T=0.36/1.06 15.37.15
```

Figure 2. Terminal Session With the Sample Problem

```
C       SAMPLE PROBLEM FOR TESTING CP-67/CMS DISTRIBUTION
C
1       PRINT 100
100     FORMAT ('     PLEASE TYPE IN THE NUMBER OF VALUES TO AVERAGE')
        READ 101, N
101     FORMAT (I2)
        IF (N) 2,99,2
2       PRINT 105
105     FORMAT ('   FIRST NUMBER? (FORMAT IS F10.4)')
        READ 103, SUM
        M=N-1
        DO 3 I=1,M
        PRINT 102
102     FORMAT ('  NEXT NUMBER?')
        READ 103, X
103     FORMAT (F10.4)
        SUM = SUM + X
3       CONTINUE
        Y = N
        AVG = SUM/Y
        PRINT 104, AVG
104     FORMAT (15X, 'AVERAGE = ',F10.4)
        GO TO 1
99      PRINT 106
106     FORMAT ('    *** TEST COMPLETE *** WELCOME TO CP-67/CMS! ')
        STOP
        END
```

Figure 3. Listing of Sample FORTRAN

INDEX

GH20-0857-2

Control Program-67/Cambridge Monitor System CP-67/CMS V 3.2, Installation Guide

Printed in U.S.A.

GH20-0857-2

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)

READER'S COMMENTS

Title: Control Program-67/Cambridge System (CP-67/CMS) Version 3.2
Program Number 360D–05.2.005
Installation Guide

Order No. GH20-0857-2

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.

Trim Along This Line

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.

FOLD FOLD

FIRST CLASS
PERMIT NO. 172
BURLINGTON, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
VM/370 Publications
24 New England Executive Park
Burlington, Massachusetts 01803

FOLD FOLD

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]