Education Update Document

CP-67
and
Cambridge Monitor System
Version 3 Level 1
Program Number 360D-05.2.005

9/1/71

This document contains descriptions of modifications to the CP-67/CMS time-sharing system for Version 3 level 1. It augments and refers to the information already available in the following manuals:

CP-67/CMS System Description Manual (GH20-0802-2)
CP-67/CMS User's Guide (GH20-0859-1)
CP-67 Operator's Guide (GH20-0856-1)
CP-67 Program Logic Manual (GY20-0590-1)
CMS Program Logic Manual (GY20-0591-1)
CP-67/CMS Installation Guide (GH20-0857-1)
CMS: SCRIPT User's Manual (GH20-0860-0)
CP-67/CMS Hardware Maintainability Guide (GH20-0858-1)
CP-67: Operating Systems in a Virtual Machine (GH20-1029-0) -- NEW--

** NOTE ** All manuals have been revised for this release except GH20-0860-0. Also, manual GH20-1029-0 is new with this release.

TABLE OF CONTENTS

INTRODUCTION

CMS Version 3.1 includes the following system refinements:

Installation of Release 20 of the OS language processors--Assembler, FORTRAN, PL/1.

Upgrading and expansion of the simulation of OS functions to insure proper operation of the language processors and the correct execution of the resultant object code.

New data error recovery routines according to Type I Specifications.

Design changes

Commands

New naming convention for source routines whereby name connotes operation and relationship to other routines.

CP-67 Version 3 Level 1 contains the following maintenance and performance improvements.

Improved sub-routine linkages for heavily used routines via BALR

Modify page selection algorithm to use reference bit.

Re-coding of scheduler (DISPATCH) to reduce inefficiencies.

Elimination of SCAN macro that generates SLT RPQ instruction.

Support of CMS file I/O by using DIAGNOSE.

Support of CMS dynamic page release by using DIAGNOSE.

Upgrading of system and user statistical gathering.

Re-coding of PROGINT for a better scheduler interface.

Multiple page read/writes on the 2301 drum where possible.

Miscellaneous changes.

Serviceability functions.

1. LANGUAGE PROCESSORS

In Version 3.0 CMS supported three OS/360 language processors: ASSEMBLER (Rel.9), FORTRAN (Compiler-Rel.14, Library-Rel.18), PL/I (Rel. 1b). For Version 3.1, all processors and supporting libraries are upgraded to OS Release 20. The object (TEXT) decks of all components of each processor were obtained by assembling the source code acquired from PID. Then, an EXEC procedure (-GEND) was written to create a modular overlay structure for each processor, e.g., FORGEND for FORTRAN. Type I component code has been modified only where necessary to interface with CMS or to correct a severe error in the type I code. Such modifications are documented in the Installation Guide, GH20-0857.

The CMS language-processor interface routines determine the read/write disk with the most available space, and use that disk to maintain its utility data sets during processing. By correcting the problem with the FILEDEF command, the user has an OS, JCL-type capability to define the characteristics of a data file: blocksize, record length, record format, resident device type. For example, a user may request an output file on tape, disk, console, punch, or printer.

The PL/I compiler (release 20) uses a different linkage to Library Subroutines than did previous releases. Therefore, if the release 20 Subroutine Library is to be used, user programs must be re-compiled.

2. OS SIMULATION UNDER CMS

In order to support the OS language processors, and the generated object code, simulation under CMS has been expanded. When a processor or a user-written program is executing and utilizing OS-type functions, OS is not overseeing this action, CMS is in control. Consequently, it is not OS Type 1 code that is in CMS, but routines to simulate in terms of CMS certain essential OS functions.

These functions are simulated to yield the same results as seen from the processing program, as specified by OS program logic manuals. However, they are supported only to the extent stated in CMS documentation and to the extent necessary to successfully execute OS language processors. The user should be aware that restrictions to OS functions as viewed from OS exist in CMS.

The OS functions that CMS will simulate are:

MACRO Title	SVC Number	Comments
*XDAP	00	called when a compiler uses XDAP
WAIT	01	wait for an I/O completion
POST	02	post the I/O completion
RETURN	03	return from a LINK-to routine
GETMAIN	04	conditionally acquire user free storage
FREEMAIN	05	release user-acquired free storage
LINK	06	link control to another load phase
XCTL	07	delete, then link control to another load phase
LOAD	08	read into core another load phase
DELETE	09	delete a loaded phase
GETMAIN/ FREEMAIN	10	manipulate user free storage
*TIME	11	get the time of day
ABEND	13	abort processing, and enter DEBUG
*SPIE	14	allow problem program to intercept program interrupts
*BLDL/FIND	18	manipulate simulated partitioned data files
OPEN	19	activate a data file
CLOSE	20	deactivate a data file
*STOW	21	manipulate partitioned directories
OPENJ	22	activate a data file
TCLOSE	23	temporarily deactivate a data file
*DEVTYPE	24	obtain device-type physical characteristics
*WTO/WTOR	35	communicate with the console
*EXTRACT	40	effective NOP
*IDENTIFY	41	effective NOP
*ATTACH	42	effective NOP
*CHAP	44	effective NOP
*TTIMER	46	effective NOP
*STIMER	47	effective NOP
*DEQ	48	effective NOP

ABDUMP	51	(same as ABEND)
*ENQ	56	effective NOP
FREEDBUF	57	release a free storage buffer
*STAE	60	allow processing program to decipher abort condition
*DETACH	62	effective NOP
*CHKPT	63	effective NOP
*RDJFCB	64	obtain information from FILEDEF command
*SYNAD	68	handle data set error conditions
*BACKSPACE	69	backup a record on a tape or disk
GET/PUT	-	access system-blocked data
READ/WRITE	-	access system-record data

*-simulated in the transient routine "SOSVCTR".

The manipulation of data is governed by an access method. To facilitate the execution of "OS Code" under CMS, the processing program must "see" data as OS would present it. For instance, when the processors expect an access method to sequentially acquire input source cards, CMS invokes its sequential access method and passes data to the processors in the format that the OS access methods would have produced. Therefore, data appears in core as if it had been manipulated using an OS access method. For example, block descriptor words (BDW), buffer pool management, and variable records are maintained in core as if an OS access method had processed the data. The actual writing onto and reading from the I/O device is handled by CMS file management.

The work of the Volume Table of Contents (VTOC) and the Data Set Control Block (DSCB) is done by a Master File Directory (MFD) to maintain disk contents and a File Status Table (FST) for each data file, respectively. All disks are formatted in physical blocks of 829 bytes.

If the filemode number of a specific data set is set to four (:), CMS will maintain the OS format within its own format onto the auxiliary device. That is, the block and record descriptor words (BDW and RDW) are written along with the data; if a data set consists of blocked records, blocks are written at a time, not records. If the filemode number is not 4, CMS will deblock or enblock records so that the CMS format will be maintained on disk - regardless of how the problem program has written or read the data.

CMS will also simulate the specific methods of manipulating data sets. A data set may be accessed:

directly -- identifying a record by a key or by its relative position within the data set

partitioned -- seeking a named member within an entire data set

sequentially -- accessing a record in a sequence relative to preceding or following items.

CMS will also simulate portions of control blocks used by OS to sustain a program during execution. Each block will be documented in detail as to which slots are maintained by CMS and which are used under OS. The control blocks are as follows:

CMSCVT simulates the Communication Vector Table. Location 16 will contain the address of the CVT control section.

CMSCB is carved out of system free storage whenever a FILEDEF command or an OPEN (SVC 19) is issued for a data set. The CMS Control Block consists of a File Control Block (FCB) for the data file and simulation of the Job File Control Block (JFCB), Input/Output Block (IOB), and Data Extent Block (DEB). The name of the data set is contained in the FCB, and is obtained from the FILEDEF argument list, or from a predetermined file name supplied by the processing problem program.

3. CENTRALIZED ERROR RECOVERY

Error recovery for disk and printer I/O operations has been consolidated into a common error handling routine. The routines affected are described below.

IOERR

PURPOSE: IOERR is a table processing program providing CMS with centralized I/O error recovery. The devices for which recovery is provided are the 2311 and 2314 direct access storage devices and the 1403 printer.

MODULES: IOERRSUP, IOERREC, IOERRMES, IONUTAB

CALLED BY: DISKIO, PRINTIO

GENERAL OPERATION: When either PRINTIO or DISKIO (i.e., the system routines that perform the I/O operations on the devices supported by error recovery) encounters an I/O error, a parameter list is built in free storage and IOERRSUP (the error recovery interface) is called. IOERRSUP initializes its free storage (containing a trace table for debugging, a WAIT parameter list to synchronize the error recovery I/O operations, and a read list for bringing a copy of IONUTAB into free storage if necessary). IOERRSUP then replaces the device's standard interrupt processing routine with ERPRINTD (the error recovery interrupt processing routine).

If the error occurred during the initialization of the operation, SIOERRT is entered. If the error occurred during the completion of the operation, CIOERRT is entered.

SIOERRT contains three routines corresponding to the three possible errors during I/O initiation. SIOBUSY waits for the channel or subchannel to become available and returns to either PRINTIO or DISKIO. SIONOTOP (not operational) calls IOERRMES to type an error message, and returns to either PRINTIO or DISKIO with a not-operational return code in register 15. SICCSW (channel status word stored) does one of four things depending on the condition causing the CSW to be stored. If the device is busy, SIOBUSY is entered. If the device contains a pending interrupt from a previously completed operation, SICCSW initializes the return code to indicate to PRINTIO or DISKIO that the device is now available and enters CLEANUP. If the started operation caused an error, CIOERRT is entered. If the started operation is an unchained immediate command and the immediate operation is error free, normal processing continues. However, if the immediate command causes an error, the routine containing the I/O operation enters ERRSUP.

CIOERRT contains four routines. CIOQUIES (quiesce I/O) analyzes the machine's state on entry into error recovery. CIONUCK (unit check) performs a sense operation if a unit check has occurred.

RTLOADER initializes the device type table (IONUTAB for direct access, the copy of IONUTAB in free storage called IOFRTAB for the printer), and passes control to IOERREC. IOERREC searches the device type table for the error and executes the routine for the recovery action. When completed, IOERREC exits to RTLOADER. When a permanent I/O error has occurred, RTLOADER calls IOERRMES. With a successful recovery, RTLOADER exits to CLEANUP. CLEANUP releases free storage and returns to either PRINTIO or DISKIO with register 15 containing the status of the operation and/or device.

During error recovery, interrupts from the device in error are processed by ERPINTD.

DISKIO and PRINTIO

PURPOSE: I/O operations directed to the 2311 or 2314 direct access storage device are processed by DISKIO. Upon a start or completion error, DISKIO builds a parameter list for IOERR to attempt error recovery. On return from IOERR, DISKIO analyzes the return code. PRINTIO handles error I/O operations which were directed to the 1403 printer in the same way.

ENTRY CONDITIONS: (when calling IOERR)

r1: Address of PLIST

r14 Return address in routine containing the I/O operation in error

r15: Address of IOERRSUP

EXIT CONDITIONS: (on return from error recovery)

r15: return code

OPERATION: When either a start or completion I/O error occurs, DISKIO (or PRINTIO) issues the EIOPL macro to (1) obtain free storage for the parameter list, (2) build the parameter list by using the CSW, I/O old PSW, and information from the device table in NUCON, (3) set a flag indicating if this is a start or completion error, and (4) enter IOERRSUP.

On return from IOERRSUP, DISKIO (or PRINTIO) analyzes the return code. If the recovery was successful, DISKIO (or PRINTIO) releases free storage and returns to the calling program. If register 15 contains X'AF' (device now available), DISKIO (or PRINTIO) releases free storage and retries the operation. If register 15 indicates a permanent I/O error, DISKIO (or PRINTIO) releases free storage and returns to the calling program with register 15 indicating a permanent I/O error.

IONUTAB and IOFRTAB

PURPOSE: IONUTAB is a table containing the recovery procedures and error messages by device type and error.

OPERATION: There are two copies of IONUTAB. One copy, IOFRTAB, is disk resident and is brought into free storage when errors are encountered on devices other than DASD devices. IONUTAB is nucleus resident, and is used for DASD errors.

4. DESIGN CHANGES

- Logical system improvements and increased execution efficiency were achieved by altering several key nucleus routines.

INISVC - Supervisor Call interrupt handler now performs faster for less code.

CMSCARE -Three minor commands have been taken out of the nucleus, thus saving core, and grouped together with five new commands in a transient module called CMSCARE. The commands become arguments to one general command - VSET. These commands are:

BLIP - sets blip character

CHARDEF - defines logical function characters (tabset, backspace, hex representation)

IMPEX - sets the order of search for commands

LDRTBLS - sets the number of reserved core pages for loader tables

LINEND - defines the logical end-of-data return character

RDYMSG - determines the length of the ready message

REDTYPE - if so equipped, the user's terminal will type all error messages in red

RELPAG - releases (or retains) contents of core

- File mode numbers can now range from 0 to 6:

0 - file is private to user of read/write but is not available when the disk is read-only.

1 or 5 - file may be read or written.
2 or 6 - file is read only. This is not currently implemented.
3 - file may be read or written, but is erased when the file is closed.
4 - reserved for OS simulation routines

- It is not necessary to login a disk to function within CMS. The "LOGIN (NO-DISK)" will pass control to the initialization routine (INIT) without benefit of a Primary disk.

- All console routines have been made reentrant code, allowing them to become a part of the locked-shared pages.

- Before entering the WAIT state and forcing the user to effect an attention interrupt, INITB67 and INITIPL will assume the terminal address is X'009' or X'01F'.

- Use will be made of lowcore locations: (all hex addresses)

Core Location	Number of Bytes	
10	4	A (Communication Vector Table)
14	4	A (SYSREF)
84	12	CP's SAVESYS vectors
90	8	name of command being executed
98	8	real date (mm/dd/yy)
A0	8	real time (hh.mm.ss)
A8	4	latest reading of virtual CPU time
AC	4	latest reading of real CPU time
B0	15	version identification (15 chars.)
D0	40	installation heading (40 chars.)
F8	8	named system that was IPL'D

- Rearrangement of nucleus routines to maintain a more orderly and efficient design to the system flow of control. By removing work areas and placing them into pages 0 and 1, more routines are made shareable. Also, when analyzing a core dump, the current system status may be determined by data centrally located in pages 0 and 1 in the working sections NUCON, INTSECT, NUSECT.

- The CMS modules DISKIO and PRINTIO now employ the technique of requesting I/O operations via the DIAGNOSE command -- code X'18'. This is used for records of 800 bytes or less. Otherwise, a normal SIO command is used. This technique allows CP-67 to use a standard CCW sequence, thereby eliminating the overhead of CCW translation, SIO simulation, etc. CP-67 will return a condition code indicating successful or unsuccessful operation. CMS will perform error recovery.
I/O operations requested via the DIAGNOSE command are not overlapped with virtual machine processing. Control does not return to CMS until the operation is completed or an error is detected.

- Cyl 0 track 0 record 3 of a user's disk may contain an identifying label. When formatting the disk, or anytime thereafter, by using "FORMAT d L", the user may write a 6-byte label. The command "STAT?" types the disk label to the console and "OFFLINE PUNCHDT filename filetype" places the label in the punched OFFLINE READ control card.

5. COMMANDS

COMBINE - New Parameters of = and *
New Options

Purpose:

The COMBINE command joins two or more disk files into a single file, moves files between disks, and changes file designations.

Format:

COMBINE	nfn nft nfm ofn1 oft1 ofm1 ... ofnN oftN ofmN (OPTIONS)

nfn nft nfm are the filename, filetype, and filemode of the file to be created.

ofn oft ofm are the filename, filetype, filemode of existing file(s) to be included in the new file.

= for nfn will adopt ofn1 (same filename)

= for nft will adopt oft1 (same filetype)

= for nfm will adopt ofm1 (same filemode)

* for ofn1 means all filenames

* for oft1 means all filetypes

* for ofm1 (not accepted)

Options:

TYPE - will type ofn1 or oft1 if an asterisk is in either field.

NODATE - makes the new creation date and time the same as the old file.

Usage:

The file to be created and each of the included files must specify filename, filetype, and filemode or one of the above options. Input files must have the same record format (fixed or variable-length). Input files of fixed length records must have the same record length. Any number of input files can be included in the new file, in the order named, but the command must not exceed a single input line.

The output file is created on the specified disk according to the mode letter of the new file. Input files may be on any read-write or read-only disk that is logged in.

If the new filename, filetype, and filemode are those of an existing file, the old file will be erased when the new file is created. The old file may be among the input files.

Notes:

a. Files may not be copied to the system (SY) disk.

b. As the input files are processed, a temporary work file is created with the identifiers (TEMP) (FILE) mm, where mm is the specified mode of the output file. When processing is completed, this file is given the designation specified for the output file. If an error occurs such that input files are destroyed, records can be retrieved from this work file.

Responses:

None.

Error Messages:

1. INVALID PARAMETER LIST.
   Invalid set of parameters entered.

2. X - DISK NOT LOGGED IN.
   Disk is not logged in for mode specified in output file.

3. X - DISK NOT IN WRITE STATUS.
   Disk is not in write status for mode specified in output file.

4. INPUT FILE DOES NOT EXIST.
   Input file specified not found.

5. ERROR WRITING OUTPUT FILE ON DISK.
An I/O error occurred or the user's allotted disk space is filled.

6. ERROR TRYING TO ALTER OUTPUT FILE.
An error was returned from 'ALTER' trying to alter the output file.

7. X - DISK NOT LOGGED IN.
Disk is not logged in for mode specified in input file.

8. X - INVALID MODE FOR OUTPUT FILE
Mode specified in output file not supported.

9. ATTEMPT TO COMBINE FIXED LENGTH AND VARIABLE LENGTH FILES.

10. X - INVALID MODE FOR INPUT FILE
Mode specified in input file not supported.

FILEDEF

Purpose:

The FILEDEF command provides device independence by allowing the user to specify the input/output devices as well as certain file characteristics to be used by a program at execution time. FILEDEF may also be used to modify, delete, and list current file definitions.

Format:

To list or clear data definitions:

```
-----------------------------
| FILEDEF | <fileid CLEAR> |
|         | *              |
-----------------------------
```

fileid the name by which the file is referred in the user's program. For FORTRAN, this is the data set reference number; for the Assembler, the ddname of the DCB; for PL1, the ddname of the DCB; and for the language processors, the assumed filetype -- e.g., TEXT or LISTING.

* means all fileids (see the PERM option for an exception).

CLEAR the file definition for the fileid specified is to be cleared.

With no operands, the FILEDEF command types a list of all current file definitions.

Once a file definition has been set up, it may be deleted with the CLEAR operand. CLEAR deletes the definition for the fileid specified. If an asterisk (*) is specified, all definitions are deleted except those defined with the PERM option (see the description below). Files defined with the PERM option must be cleared individually.

To relate a file to a device:

```
-----------------------------------------------
| FILEDEF | fileid device <parms> <(options> |
-----------------------------------------------
```

device specifies the input/output device for the file. Device must be one of the following:

CON	User's terminal
DSK	Disk
DSK-mm	Disk with specific mode
DUMMY	Dummy device for which no actual I/O is to occur
PRT	Printer

PUN	Card punch
RDR	Card reader
TAPn	Magnetic tape (n must be blank or 1 or 2)

parms are the data and file characteristics. The valid parms depend upon the device named. These are defined under the specific devices.

Options:

PERM specifies that this definition is to remain in effect until cleared by fileid CLEAR.

NOCHANGE specifies that the file definition is to be set up unless a definition already exists for the fileid. If a definition exists, the old one is retained.

Usage:

To set up a file definition, both fileid and device must be specified.

A DUMMY device may be used at times when no I/O is wanted -- e.g., for program testing.

Depending on the device specified, parameters may be necessary. The file characteristics that may be called for are described below with the devices to which they are appropriate.

PRT, PUN, RDR

These devices require the parameters RECFM, LRECL, and BLKSIZ.

device	RECFM	LRECL	BLKSIZ
PRT	F	133	133
PUN	F	80	80
RDR	F	80	80

The method of changing these is described under TAP.

CON

The only valid parameters for CON are CASE, RECFM, LRECL, and BLKSIZ. If none are specified, the defaults are used.

CASE LOWER specifies that console data is not to be translated to uppercase. If CASE LOWER is not specified, console I/O is translated to uppercase.

TAP

The only valid parameters for TAP are RECFM, LRECL, and BLKSIZ. For seven-track tapes, MODE may be specified.

To enter user values for RECFM, LRECL, and BLKSIZ, quantities are specified in pairs: name of parameter, then specific value.

RECFM format specifies the record format for the file. format must be one of the following:

F	Fixed unblocked
FB	Fixed blocked
V	Variable unblocked
VB	Variable blocked
U	Undefined

LRECL nn specifies the logical record length in bytes.

BLKSIZ nn specifies the block size in bytes.

For seven-track tapes, a MODE may be defined.

MODE nn specifies the mode number desired from the chart below. Note that 1 - 5 are for 800 BPI, 6 - 10 are for 556 BPI, and 11-15 are for 200 BPI.

MODE NUMBER	800 BPI	556 BPI	200 BPI	PARITY	CONVERTER	TRANSLATOR
1	1	6	11	ODD	ON	OFF
2	2	7	12	ODD	OFF	ON
3	3	8	13	ODD	OFF	OFF
4	4	9	14	EVEN	OFF	ON
5	5	10	15	EVEN	OFF	OFF

DSK and DSK-mm

The format for a device of either DSK or DSK-mm differs from the other devices in that a filename and filetype for the file on disk must be specified -- i.e.,

FILEDEF	fileid device filename filetype <parms> <(options>

filename filetype are the CMS file identifiers for the disk file.

The filemode of the disk file may be specified with the DSK-mm form of the DSK device, where mm is the mode letter and mode number desired.

The valid parameters for DSK and DSK-mm are RECFM, LRECL, and BLKSIZ. (There are additional parameters for BDAM disk files described later in this section.)

The method by which these may be defined is described under TAP.

DSK files for BDAM processing

The same rules apply for disk files for BDAM processing as apply to DSK files in general. In addition, KEYLEN, XTENT, LIMCT, and OPTCD may be specified.

KEYLEN nn specifies the size of the record key in bytes.

XTENT nn specifies the number of records in the extent for the file. A default value of 50 is assumed.

LIMCT nn specifies the number of extra tracks or blocks to be searched.

OPTCD code specifies the direct access processing desired. Code may be one of the following:

A Actual device addressing
E Extended search
F Feedback addressing
R Relative block addressing

Responses:

xxxxxxxx yyy
When FILEDEF is issued with no parameters, a list of current file definitions is typed. xxxxxxxx is the fileid defined, and yyy is the device for which it is defined.

Examples:

a. filedef
FT04F001 DSK filename filetype
TEXT DSK
A list of all current file definitions is typed.

b. filedef * clear

All file definitions are cleared except any that may have been defined with the PERM option.

c. filedef text pun
A file definition for fileid TEXT is defined to the card punch.

d. filedef 04 DSK new file
A file definition is created for FT04F001 on disk with the CMS file identifier NEW FILE.

e. filedef outfile dsk output test recfm fb blksiz 320 (perm
A permanent definition for OUTFILE is set up. The file is defined to disk, where it's filename and type are OUTPUT TEST. It is fixed-blocked, 320 bytes per block. This definition holds until a FILEDEF OUTFILE CLEAR is issued, or the user re-IPL's CMS.

Error Messages:

E(00001) FIL001: PARM1 INVALID
The first parameter is something other than an *, a one or two-digit number, or a DDNAME.

E(00002) FIL002: BAD RTN CODE
The FILEDEF command got a bad return code from its call to SVCFREE or SVCRET. This is a system problem not correctable by the user. Try the command again.

E(00003) PARAMETERS MISSING AFTER OPERAND xxxxxxxx
A parameter required by the FILEDEF command has been left out of the option list by the user. Retype the command with the proper options.

E(00004) SUPERFLUOUS OR INVALID PARAMETERS AFTER xxxxxxxx
One or more parameters after xxxxxxxx is superfluous or invalid. Retype the command with the proper option list.

E(00005) FIL005: BAD LRECL OR BLKSIZ VALUES
One of the following conditions exists:

Value exceeded the maximum allowable size.

LRECL or BLKSIZ values contained nonnumerics.

E(00006) ILLEGAL xxxxx REQUEST
xxxxx is either DUMMY or CLEAR.
ILLEGAL CLEAR REQUEST is generated when a CLEAR request is made for a file definition not created by the user. ILLEGAL DUMMY REQUEST is generated when a DUMMY request is made for DDNAME or DSRN which has already been defined by FILEDEF.

E(00007) FIL007: DS-NAME/-TYPE MUST BE SPECIFIED
These items are required when DSK is specified. Retype the FILEDEF command with the proper parameters.

E(00008) FIL008: PARM1 MUST BE DDNAME
The user has left out an *, DDNAME or DSRN as his first parameter. Retype the FILEDEF command with the proper parameters.

E(00009) FIL009: BAD OPTCD PARAMETER
The user has specified an unsupported option code. The option code must be either E, F, A, or R.

WRTAPE - New Parameters of Blocking Factor and EOF

Purpose:

WRTAPE copies files, fixed and variable length, from disk to tape. If the filetype is 'LISTING', Assembler and compiler carriage - control codes are translated to machine codes.

Format:

WRTAPE	fname	ftype	<fmode> P	<Blocking Factor> default 10	<EOF> default 'NO EOF'

fname ftype fmode Specifies the file to be copied

Usage:

WRTAPE copies files from a specified logged-in disk to TAP2. Records can be fixed length or variable length. The tape is not rewound when the command completes. Blocking of records can be specified by the user.

WRTAPE handles any file of the format previously described. In the case of a LISTING file created either by the ASSEMBLE, PL/I, or FORTRAN commands, it translates the first byte of each record, which is the carriage control code, into machine code for the printer.

Notes:

a. WRTAPE format of writing files to tape is acceptable to the command 'TAPRINT' if a blocking factor of ten is specified.

b. Use the facilities of the TAPE command to position the tape past any existing files. To mark an end of file for the command 'TAPRINT', two end of file marks must be written.

c. Under CP, the tape must be attached by the operator. 'TAP2', addressed at 181, should be specified when the operator is requested to attach the tape.

d. Tape files written with WRTAPE are not suitable for re-reading with TAPE LOAD.

e. Maximum block size accepted = 32,760 bytes

Responses:

None.

Example:

```
WRTAPE PROG SCRIPT A1 100 EOF
```

The file PROG SCRIPT has variable length records and when copied to tape, the records are grouped in blocks of 100 and end-of-file is written.

Error Messages:

E(00002) PARAMETER ERROR
Invalid parameter list was entered

E(00003) BLOCK SIZE TOO LARGE
Block size exceeds maximum 32,766 bytes

E(00004) FILE NOT FOUND
File specified not found

E(00005) ERROR READING FILE
Error occurred reading file from disk

E(xxxxx) TAPE ERROR
A tape error occurred which could not be recovered. The error code is unpredictable.

COMPARE - New option

Format:

COMPARE filename1 filetype1 filemode1 filename2 filetype2 filemode2 (NOSEQ)

(NOSEQ) = Do not check last eight bytes of record.

Additional Responses:

FIRST FILE NOT FOUND

SECOND FILE NOT FOUND

FIRST AND SECOND FILE(S) ARE THE SAME FILE

New Error Messages
(Replacing Current Error Messages):

E(00001) Parameter error

E(00002) First and second file(s) are the same file.

E(00003) At least one record differs.

E(00004) FATAL ERROR

OFFLINE - New parameter of * * and * * fm

Format:

OFFLINE: command filename filetype <filemode>
* * fm

Asterisks: Valid only when command = READ

* Specifies that filename, filetype and filemode are found in OFFLINE READ control cards in the input card stream.

* * Specifies that the filename, filetype are found in OFFLINE READ control cards in the input card stream, mode defaults to 'P' disk.

* * fm Specifies that the filename, filetype are found in OFFLINE READ control cards in the input card stream; mode accepted is the one specified in the command line.

VSET

Purpose:

VSET allows the user to control different aspects of his environment at his console. VSET BLIP controls the character designated to notify the user of every two CPU seconds of execution time; VSET CHARDEF controls the definitions for logical symbols, such as line delete, character delete, backspace, and tab characters, and the hexadecimal representation of defined characters; VSET IMPEX controls the order of search for commands; VSET LDRTBLS controls the number of pages of core used for loader tables; VSET LINEND controls the definition for the logical line-end character; VSET RDYMSG controls the length of the error and ready messages typed by CMS; VSET REDTYPE controls the color of the CMS error messages; and VSET RELPAG controls the releasing of pages of core upon command completion.

Format:

```
-----------------------------
| VSET | function <operands>|
-----------------------------
```

function may be one of the following:

BLIP
CHARDEF
IMPEX
LDRTBLS
LINEND
RDYMSG
REDTYPE
RELPAG

operands are described under "Usage" for each particular function.

Usage:

VSET has multiple uses depending upon the function specified. Each is described below.

VSET BLIP

```
-----------------------------
| VSET BLIP <char <count>> |
|           (OFF)    1     |
-----------------------------
```

char is the character(s) to be typed. The default is a sequence of nonprinting characters -- uppercase shift, lowercase.

(OFF) sets BLIP to nothing, so there is no indication of CPU execution time being used.

count is a number from 1-8 indicating the number of characters in char. The default is 1 character.

The character(s) specified are typed every two seconds of CPU execution time to give the user a printed recording of the execution time of his program. If the default character is used, the typing element shifts to uppercase, then lowercase, to indicate the CPU execution time. Nothing prints. The entire execution time indication of BLIP can be turned off with the (OFF) operand. A user may specify noncharacter BLIP indicators by specifying them in hexadecimal in an Assembly Language Program.

VSET CHARDEF

VSET	CHARDEF logsym <char>
	transtype <char <hexcode>>

logsym specifies the logical symbol for which a character is being defined. The valid logical symbols and their meanings are described below.

- B logical backspace symbol in EDIT. The default symbol is %.
- C logical character-delete symbol. The default symbol is @.
- L logical line-delete symbol. The default symbol is ¢.
- T logical tab symbol in EDIT. The default symbol is #.

transtype specifies the type of translation being defined for the character represented. The valid translation types and their meanings are described below.

- IN terminal input translation to hexadecimal
- OU terminal output translation to hexadecimal
- IO terminal input and output translation to hexadecimal

char specifies either the character to be defined as a logical symbol, or the character for which translation to hexadecimal is to be made. If a character is not specified, there is no character for that logical symbol, or all translations for that translation type are cancelled.

hexcode specifies the specific hexadecimal representation to which char is to be translated.

There are two general ways in which CHARDEF can be used - to define logical symbols, and to define character translations to hexadecimal.

For logical symbols, char becomes the symbol for the logical function specified. If no char is specified, there is no symbol for that logical function. The definitions for Backspace and Tab are effective only in EDIT and CEDIT. The definitions for Character and Line are effective for CMS and all its commands (including EDIT, CEDIT, and DEBUG), but not for CP. By redefining the default logical symbol characters (i.e. & for backspace, @ for character-delete, ¢ for line delete, # for tab), these characters may be typed as normal input characters.

For character translations, char becomes the character that is to be translated to the hexadecimal representation specified. The translation occurs at the time specified by the translation symbol (i.e., on terminal input by IN, on terminal output by OU, or on both terminal input and output by IO). In this way, characters not represented on a specific keyboard may be entered by specifying the char defined to the hexadecimal representation of the missing character. For example, a terminal not having a < sign, might use the (to enter a < by specifying VSET CHARDEF IN (4C. If no char is specified with a translation type (i.e., IN, OU, or IO), all translations of that type are cancelled.

The characters defined by VSET CHARDEF remain in effect until (1) another VSET CHARDEF either changes or cancels it, (2) CMS is reIPL'ed, or (3) the user logs out of CP.

VSET IMPEX

VSET	IMPEX ON OFF

ON specifies that when a command is entered from the terminal, the search is first for a file with a filetype of EXEC, then for a filetype of MODULE. This is the default.

OFF specifies that the search for filetype EXEC is to be bypassed.

The order of search for commands specified at the terminal may be altered by the user with the VSET IMPEX command. Normally, when a command is typed, there is an implied search first for a filetype of EXEC. For example, if SAMPLE is typed at the terminal, CMS first searches for a file SAMPLE EXEC. If this is found, it is executed as if EXEC SAMPLE had been typed. If this file is not found, there is a check of user, then system, abbreviations -- and, if SAMPLE is an abbreviation, it is expanded and a search is made for the extended fname EXEC. If extfname EXEC is found, it is executed as if EXEC extfname had been typed.

If the search for an implied EXEC filetype is not successful, the search is then made for SAMPLE MODULE. If found, it is executed as if LOADMOD SAMPLE then START were typed. If it is not found and SAMPLE is a user or system abbreviation, a search is made for extname MODULE. If this is found, it is executed; if it is not found the ?CMS message is typed indicating an invalid command.

With VSET IMPEX OFF, the first series of searches for the EXEC file are bypassed -- and the search is only made for filetype MODULE. In this mode, EXEC files may be executed only by explicitly typing EXEC fname.

VSET LDRTBLS

VSET	LDRTBLS <nn>

nn is the number of pages of core the user would like to have set aside for loader tables. If omitted, the number of pages of core currently set aside for loader tables is typed.

The user may examine or change the number of pages of core used for loader tables with VSET LDRTBLS.

Normally, a virtual machine of 256K has two (2) pages of loader tables a larger virtual machine has three (3). This number may be changed with VSET LDRTBLS nnn provided that (1) nnn is a decimal number less than 128, and (2) the user has enough core available to allow nnn pages to be used for loader tables. If both of these conditions are met, nnn pages are set aside for loader tables.

If a VSET LDRTBLS request fails because there is not enough core, a smaller number of pages should be requested. If the request fails because core is in use, the user may re-IPL CMS, issue LOGIN (NODISK, then request the number of pages again.

A user may find out the number of pages of loader tables in his virtual machine at any time by VSET LDRTBLS with no operand.

Responses:

NUMBER OF PAGES FOR LOADER TABLES: nnn
This response occurs when VSET LDRTBLS has been specified with no operands. nnn is the current number of loader table pages the user has.

VSET LINEND

```
-------------------------
| VSET | LINEND <char> |
-------------------------
```

char specifies the character that is to become the logical line-end symbol. The default symbol is #. If no symbol is specified, no character is defined as the logical line-end, and the only way to end a line is with carriage return.

The logical line-end character permits a number of logical input lines (each separated by the line-end symbol) to be typed on a single physical input line. The physical input line is terminated by a carriage return. Logical input lines are terminated by the line-end character or by the carriage return. Each call to read a line from the terminal returns the logical input line. Subsequent calls to read a line from the terminal return the next logical input line.

The line-end symbol can be used to input multiple logical lines whenever a physical line is input from the typewriter, whether to CMS or to a program. In addition, logical lines can be input and stacked by use of attention to CMS (double attention if from CMS and running under CP). See Terminal Usage-"Attention Interrupt".

If no VSET LINEND has been issued, the # is the logical line-end symbol. The pound sign (#) is also the logical tab symbol, but the line-end symbol takes precedence over the tab symbol.

If VSET LINEND has no operand, there is no logical line-end symbol; logical lines must be terminated as physical lines with the carriage return.

VSET LINEND does not redefine the logical line-end symbol for CP.

VSET RDYMSG

```
----------------------
| VSET | RDYMSG ON   |
|      |        OFF  |
----------------------
```

ON specifies that CMS is to type standard error and ready messages (eg. R; T=nn.nn/nn.nn hh.mm.ss)

OFF specifies that abbreviated error and ready messages (eg. R;) are to be typed.

Abbreviated error and ready messages (i.e., E(nnnnn); and R;) may be requested with VSET RDYMSG OFF. Besides the shorter message (which eliminates the printout of CMS and CMS + CP execution time and the clock time), there is one carriage return rather than the usual two.

VSET REDTYPE

VSET	REDTYPE OFF
	ON

OFF specifies that all CMS error messages are to be typed in black. This is the default value.

ON specifies that error messages are to be typed in red.

Provided a terminal is equipped with the red ribbon control feature (RPQ #868019) and a black-red ribbon, REDTYPE ON allows a user to have CMS error messages typed in red. A KE command (to truncate terminal output lines) has no effect on lines typed in red.

If a user specifies REDTYPE ON, and the terminal is not equipped as described above, extraneous characters may be typed.

VSET RELPAG

VSET	RELPAG ON
	OFF

ON specifies that after certain commands have executed, core is to be released and zeroed. This is the default.

OFF specifies that the contents of core are to remain after the completion of certain commands.

When CMS is running in a virtual machine, pages of core are released and zeroed after the following commands complete execution: ASSEMBLE, CEDIT, COMBINE, COMPARE, EDIT, FORTRAN, MACLIB, MAPPRT, PLI, SORT, SPLIT, TAPE, TXTLIB, and UPDATE. This is the normal mode of operation - that obtained by default or when VSET RELPAG ON is specified.

However, if a user wishes to examine core after any of these commands has finished - e.g., for debugging or analyzing a problem - it is desirable to inhibit this releasing feature. This is done with VSET RELPAG OFF. The commands affected are given in the above paragraph.

Examples:

a. vset blip ?
This causes the question mark (?) to print after every two seconds of CPU execution time, giving the user a record of CPU time execution.

b. vset charder c !
The character-delete symbol is defined to the exclamation point (!). The @ symbol can be used as normal input.

c. vset charder |
The line-delete symbol no longer exists. The @ can be used as a normal input character.

d. vset charder in (4C
The (is translated to a hex 4C (the < symbol) whenever it is inputted. This enables a < to be inputted from a terminal that does not have a < symbol.

e. vset charder in
All previously defined input character translations are cancelled.

f. vset impex off
The normal implied search for a filetype of EXEC is bypassed. To execute an EXEC file, EXEC filename must be entered.

g. vset ldrtbls
NUMBER OF PAGES FOR LOADER TABLES: 2
The current number of pages of core set aside for loader tables is 2.

h. vset ldrtbls 4
There are now 4 pages of core set aside for loader tables.

i. vset linend !
The line-end symbol is set to the exclamation mark (!).

j. vset rdymsg off
R;
erase no file
E(00002)
CMS ready and error messages are abbreviated as shown.

k. vset rdymsg on
R; T=0.00/0.01 13.30.06

erase no file
E(00002); T=0.00/0.01 13.30.14 Standard CMS error and ready messages are typed to the terminal.

Error Messages:

E(00001) VST001: BAD CHARDEF ARGUMENT
Either no or invalid operands were specified with VSET CHARDEF. Reissue the command.

E(00002) VST002: BAD VSET FUNCTION
Either no or an invalid function was specified with VSET.

E(00003) VST003: BAD LDRTBLS ARGUMENT
Either no or an invalid operand was specified with VSET LDRTELS. If an operand is specified, it must be a decimal number less than 128.

E(00004) VST004: CANNOT MODIFY LOADER TABLES
The number of pages requested with VSET LDRTBLS is greater than the number of pages available, or the core is already in use. Either (1) request a smaller number of pages, or (2) re-ipl CMS, issue LOGIN (NODISK, then try VSET LDRTBLS again.

E(00005) VST005: BAD RDYMSG ARGUMENT
The only valid operands for VSET RDYMSG are ON and OFF, one of which must be specified. The ready and error message lengths are not affected. Reissue the command.

E(00006) VST006: BAD IMPEX ARGUMENT
The only valid operands for VSET IMPEX are ON and OFF, one of which must be specified. Reissue the command.

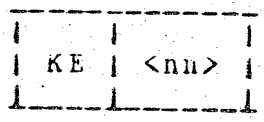
E(00007) VST007: BAD RDTYPE ARGUMENT
The only valid operands for VSET RDTYPE are ON and OFF, one of which must be specified. Reissue the command.

E(00008) VST008: BAD RELPAG ARGUMENT
The only valid operands for VSET RELPAG are ON and OFF, one of which must be specified. Reissue the command.

KE

PURPOSE: The KE command controls the length of the line being typed at the terminal.

FORMAT:



USAGE: KE is activated from the CP-67 environment (entered from CMS by hitting the attn key) by typing in KE nn or KE. IOINT gives control to CONSI (see CMS PLM GY20-0591) which determines whether or not an operand was entered. If not, a default value of 72 is assumed. The column limit is set to the specified value, the KE flag bit is set, the ATTN buffer is released, and control is returned to IOINT.

EXAMPLES

a. Ke
The line length is set to the default value of 72 characters.

b. Ke 80
the line length is set to 80 characters.

New EXEC Key Words

&PUNCH	LINE

&PUNCH punches line to the virtual punch. All keywords, symbolic arguments, etc., are substituted into the line. Any word or words that exceed eight (8) characters are left justified and truncated on the right.

&UPUNCH	LINE

&UPUNCH will punch the unscanned line that was entered up to 72 characters. This command cannot be nested within another EXEC command.

&UPRINT	LINE

&UPRINT will print the unscanned line that was entered up to 72 characters. This command cannot be nested within another EXEC command.

&TYPEOUT	ALL ON ERROR OFF NOEXEC KILL RESUME	TIME NOTIME	PACK NOPACK

KILL: suppresses all typing to the terminal.

RESUME: returns terminal typing to the status prior to the KILL command.

New ALTER Option

Options:

TYPE will type altered files

NOUP directory is not updated

Usage:

The TYPE option will type the name and/or type and/or mode of the altered file depending on which fields contain an asterisk in the old filename, old filetype, or old filemode.

New ERASE Option

TYPE in parameter list after filetype or mode will type the files that are being erased.

New Page Release Facility

ASSEMBLE	EDIT	SORT
CEDIT	FORTRAN	SPLIT
COMBINE	MACLIB	TAPE
COMPARE	MAPPRT	TXTLIB
	PL/I	UPDATE

These routines now include the user page release function described in the following paragraph.

After successful completion and prior to returning to the user or caller the current routine references NUCON and turns the page release flag on. When the program returns to INIT, this flag is checked and if it is on, INIT issues a diagnose X'10' to CP to release the user pages from 12000 Hex up to the value of LOWEXT.

If the user wishes to prevent the release of pages, he must issue

CMS command VSET HELPAG OFF (see VSET).

6. NAMING CONVENTION

To make the filenames of CMS source routines more meaningful to the system programmer responsible for maintaining CMS, a naming convention has been established. A routine's function and relationship to other routines will consequently be identified by prefixes and suffixes. Some naming conventions:

- device dependent routines prefixed by a code denoting the physcial device type: e.g., DISK, CONsole, etc.
- device interrupt handlers suffixed by "INT": (CONINT)
- software interrupts prefixed by "INT": (INTSVC)
- simulators of OS functions prefixed by "SO": (SOQSAM)
- miscellaneous routines of similar functions prefixed by the same code: (FREESYS, FREEXTND)
- device input/output executors suffixed by "IO": (TAPEIO)

The following chart will show:

1. the filename of the source SYSIN deck
2. the internal entry point(s) or START card label if dissimilar from the filename
3. how the routine is used:

   N - nucleus resident
   NI - nucleus resident during the IPL procedure only - then no longer needed
   NS - nucleus resident, sharable code
   D - disk resident module
   DC - component of a disk resident module
   T - transient module

4. relationship between versions 3.0 and 3.1 with respect to the individual routine:

   S - same, routine is unchanged
   U - updated, the update deck that was used to generate the current version of the routine will be supplied.
   UN - updated and renamed. The newly named update deck will be supplied. Note: the version 3.0 source and the version 3.1 source - prior to applying the update - are identical.
   R - replacement, complete substitute for the mentioned version 3 source routines
   RN - rename, the filename of the version 3.0 source deck

was changed.
N - new

5. a brief comment about each function

External Filename	Internal Entry Point	Usage	Relation vis a vis Ver 3.0	Functions
$	-	T	U	execution initiator
ABBREV	-	N	S	abbreviation processor
ACTLKP	-	NS	U	determine active files
ADTLKP	-	NS	U	determine available disks
ALTER	-	T	U	change file identification
ASMDIRT	-	DC	R-ASMDIRT	ASSEMBLER auxiliary directory
ASSEMBLE	-	D	R-ASSEMBLE,ASMREAL ASMA01,ASMA02 ASMA03,ASMAFIND	ASSEMBLER interface
BATBOMB	-	N	S	batch monitor component
BATCH	-	N	U	batch monitor component
BATDECC	-	N	S	batch monitor component
BATJCB	-	N	S	batch monitor component
BATLIST	-	N	S	batch monitor component
BATPRES	-	N	S	batch monitor component
BATSCTL	-	N	S	batch monitor component
CARDIO	CARDRDPH	N	S	reader/punch executor
CEDIT	-	D	U	editor for large files
CLOSIO	-	N	S	close executor for Unit Record Equipment
CMSCARE	-	T	N	auxiliary command module
CMSCONF	CPFUNCTN	T	U	virtual CP console function executor
CMSFORM	FORMAT	D	U	disk formattor
CMSIPL	IPL	T	S	CMS IPL invoker
CMSTIME	-	NS	R-CMSTIME	virtual time accounter
CNVTFIV	CVTFV	D	RN-CVTFV	convert fixed/variable
CNVT26	-	D	S	convert 026/029
COMBINE	-	D	R-COMBINE	file manipulator
COMPARE	-	D	U	file matchmaker
CONATTN	ATTN	NS	UN-ATTN	attention handler
CONINT	CONSI	NS	UN-CONINT	console interrupts
CONREAD	WAITRD	NS	UN-WAITRD	console input
CONWAIT	-	NS	U	console wait
CONWRITE	TYPE	NS	UN-TYPLIN	console write
DEBDUMP	-	N	N	debug dump executor
DEBUG	-	N	U	problem determination aide
DISK	-	D	U	disk utility
DISKINT	-	N	UN-DIOSECT	disk interrupts
DISKIO	RDTK/WRTK	NS	UN-DIO	disk I/O executor
DUMPRST	DUMPREST	D	S	dump/restore utility
DUMPD	-	D	S	dump disk utility

DUMPF	-	D	S	dump file utility
ECHO	-	D	S	terminal tester
EDIT	-	D	U	editor
EDITDUAL	-	NS	U	file eradicator
ERASE	-	NS	U	file eradicator
EXEC	-	NS	U	exec bootstrap
EXECTOR	-	D	U	exec work module
FILEDEF	-	T	R-FILEDEF	define file routine
FINIS	-	NS	U	file deactivator
FORDIRT	-	DC	R-FORTDIRT	FORTRAN auxiliary directory
FORTRAN	-	D	R-FORTRAN, FORTIO,	FORTRAN interface
FREESYS	FREE/FRET	NS	UN-CMSFR...	system free storage
FREEXTN	EXTEND	N	UN-CMSEXTND	system free storage
FSTLKP	-	NS	U	file lookup
FUNCTAB	-	NS	R-FUNCTAB	internal function table
GENDIRT	-	T	S	auxiliary direct generator
GENMOD	LOADMOD/ GENMOD	NS	U	module manipulator
GLOBAL	-	T	R-GLOBAL	library governess
HNDINT	-	T	S	handle I/O interrupts
HNDSVC	-	T	S	handle SVC interrupts
IADT	-	NS	U	init Active Disk Tables
INIT	-	N	U	initializer
INITIPL	TRANSAR, IPLDISK	NI	R-LAST, IPLDISK	reads CMS from disk
INITB67	BARE67	NI	R-BARE67	bare CPU initial processor
INITSUB	-	NI	U	sub-initializer
INITSYS	SYSGEN	NI	RN-SYSGEN	initializes S-disk
INTEXT	EXTINT	NS	UN-CMSEXTIT	external interrupt
INTIO	IOINT	NS	UN-CMSIOIT	I/O interrupt
INTMACH	MCHINT	NS	N	machine interupt
INTPROG	PRGINT	NS	R-CMSPRGIT	program interrupt
INTSECT	-	N	N	interrupts work area
INTSVC	SVCINT	NS	R-CMSSVCIT	SVC interrupt
IOFBTAB	-	N	N	disk tables
IOERR	-	N	N	I/O error processor
IONUTAB	-	D	N	I/O error tables
IXCBLTP	-	DC	S	Fortran library
IXCCMS	-	DC	U	Fortran library
IXCDEF	-	DC	S	Fortran library
IXCDEFTB	-	DC	S	Fortran library
IXCDSD	-	DC	S	Fortran library
IXCFREM	-	DC	S	Fortran library
IXCGETP	-	DC	S	Fortran library
IXCRENM	-	DC	S	Fortran library
IXCRERD	-	DC	S	Fortran library
IXCTAP	-	DC	S	Fortran library.
IXECMS	-	DC	R-IXECMS	PL/I library
IXECLOK	-	DC	R-IXECLOK	PL/I library
IXEFILE	-	DC	R-IXEFILE	PL/I library
LDR	-	NS	U	relocatable loader
LDRIO	-	N	R-LDRIO	loader I/O executor
LDRLIBE	-	N	R-LDRLIBE	loader library processor

LDRSUBS	-	N	R-LDRSUBS	loader subroutine
LDRSYM	GETSYM	N	RN-GETSYM	loader symbols
LISTF	-	T	S	list files
LOAD	-	N	U	load initiator
LOG	-	N	U	maintain disk direct
LOGIN	-	T	U	login a disk
MACLIB	-	D	U	macro library
MAPPRT	-	D	U	nucleus map
MODMAP	-	T	S	module map
NUCON	-	N	R-NUCONTS	nucleus constants
NUDVEXT	-	N	N	I/O device tables
NUSECT	-	N	N	nucleus work section
OFFLINE	-	T	R-OFFLINE	file utility
OSTAPE	-	D	S	tape-file utility
OVERRIDE	-	D	U	override executor
OVERSUB	-	DC	UN-JASOVER	override executor
PLI	-	D	R-PL/I	PL/I interface
PLIDIRT	-	DC	U	PL/I auxiliary directory
POINT	-	NS	S	sys file manipulator
PRINTF	-	T	S	print files
PRINTIO	PRINTR	NS	UN-PRINTR	printer executor
QQTRK	-	NS	S	track manager
RDBUF	-	NS	U	basic read executor
READFST	-	DC	U	login module
READMFD	-	DC	U	login module
RELEASE	-	T	U	release user disk
RLUFD	-	DC	U	login module
SCAN	-	N	U	decipher input lines
SCSFOR	-	DC	S	script module
SCSLNK	-	DC	S	script module
SCSPRT	SCRIPT	D	S	script processor
SOABEND	-	N		R-ABSTD *sim of OS ABEND
SOBDAM	-	NS	N	*sim of BDAM
SOBSAM	-	NS	R-RDWR	*sim of BSAM
SOCNTRL	-	NS	R-NTPT,CHECK	*sim of NOTE/POINT/CHECK
SOEOB	-	NS	N	*sim of end-of-block
SOLINKS	-	NS	UN-LINKAGE	*sim of control transfers
SOMAIN	-	NS	RN-STORAGE	*sim of SVC 4,5,10
SOOPCL	-	NS	R-OPEN	*sim of SVC 19,20,22,23
SOQSAM	-	NS	R-GET	*sim of GET,PUT
SORT	-	D	U	sort data within files
SORTREE	-	DC	S	sort module
SORTSRCH	-	DC	S	sort module
SOSVCNU	-	NS	N	*sim of misc SVC in nucleus
SOSVCTR	-	T	R-SVCCARE	*sim of misc SVC in transient
SOSVCT2	-	T	R-SVCCARE	*sim of misc SVC in transient(con
SPLIT	-	D	U	file utility
START	-	N	S	commence execution
STATDSK	STAT	T	U	disk statistician
STATE	-	NS	U	file lookup
SYN	-	T	U	synonym processor
TAPE	-	D	U	tape utility
TAPEIO	-	T	S	tape I/O executor
TAPRINT	-	D	S	listing tape utility

TPCOPY	-	D	S	copy tapes
TRAP	-	N	S	interrupt trap
TRKLKP	-	NS	S	track management
TXTLIB	-	D	U	library processor
UPDATE	-	D	U	file utility
UPDISK	-	NS	U	disk management
USE	-	N	U	load initiator
WAIT	-	NS	S	I/O wait
WRBUF	-	NS	U	basic write executor
WRTAPE	-	D	U	tape write utility

* "sim of" means "CMS simulation of the OS function ... "

7. Improved subroutine linkage.

Some routines in the CP-67 nucleus are called for processing and exit later with no intervening calls to other modules. These routines could thus use a common save area and linkage could thus be reduced to a simple BALR instead of the standard SVC call. The common save area used is named BALRSAVE and is located in module PSA in page zero. The CALL macro has been updated to select the correct linkage according to the called routine. The following routines have been modified to use the BALR linkage:

CHKCUACT
CPSTACK
BINDEC, BINHEX, DATETIME, DECBIN
FPCONV, HEXBIN
DISACT
FREE, FRET, FRETR
IOISTVDE, IOISTVCU
PAGUNLCK
QUEVIO, QUERIO, CHFREE
RUNITSCN, VUNITSCN
UNTRANS

8. Reference page algorithm.

The algorithm in PAGTRANS to select a page for a user has been modified. Previously, the loop searched for an available page or a page belonging to a user not in a dispatchable queue. The modified loop searches for a page that has not been referenced. Referenced pages found in the loop have the reference bit turned off. If a non-referenced page cannot be found, then the scan continues to find the first non-referenced page. The reference bit will have been reset during the first scan. If a changed page is selected it must first be written out to auxilliary storage before the requested page can be read in.

9. Improved DISPATCH

The DISPATCH module has been completely re-written to give a cleaner interface from other modules, to improve 'real' timer maintenance using a binary clock (BINCLOCK) and to use finite queues for scheduling and dispatching users based upon priorities adjusted by user behaviour to conform to system restraints.

Several new entry points have been defined in DISPATCH so that the dispatcher can determine whether-or-not a user's status has changed, based upon the entry and not upon detailed analysis. Certain entry conditions merely account for user time and then continue to run the interrupted user,

provided his quantum has not been exceeded. Other entry conditions account for user time and, thus, search for another user to run, knowing that the running user is now not runnable.

The maintenance of 'real' timers has been changed to reduce the overhead involved. Essentially, a chained list of all users with 'real' timers is maintained. The list is ordered by the users virtual timer value. Thus, the user at the top of the list expects a timer interrupt first. A binary clock (BINCLOCK) value is maintained by CP-67 to account for all time, CPU and wait. This value is compared to the 'real' timer value at the top of the list. When an interrupt is indicated, CP processes it by the usual virtual PSW swap. The routines used to maintain the time-of-day clock, the binary clock and to schedule 'real' timers are placed in a new module called SCHEDULE that is referenced by DISPATCH.

DISPATCH now maintains two finite lists of runnable users; users runnable and eligible, and users runnable but not eligible. A runnable user is a user not in I/O pagewait or console function wait, and not in virtual wait state, and whose running parameters (CPU time, privileged ratio, page demands, etc.) combined, conform to fit within certain system restraints. These system restraints are imposed depending upon the system configuration, certain operating parameters and the user activity. Interactive users are primarily limited by a maximum number allowed to be eligible. Compute and I/O users are limited by a paging activity index. Each user has a paging activity index calculated from such values as number of pages in core, number of page reads performed and the ratio of CP to user CPU time used. This paging activity index is then used in conjunction with a system paging activity threshold to determine if the limit has been reached. If the system load will accommodate the increased paging activity according to the user index, then the user is considered runnable. An eligible user is a user in pagewait, I/O wait or virtual machine wait state enabled for an operating channel and whose running characteristics (interactive or paging activity index) do not exceed system restraints, when made runnable. Some users will occasionally be neither runnable nor eligible, for instance, a user in console function wait. The movement of users through the various states is a function of CPU time used and other operating characteristics of the virtual machine.

10. SCAN elimination

Previous versions of CP-67 used a SCAN macro in several places to generate code used to search chained lists for particular conditions. The macro generated an SLT instruction which is available as a hardware RPQ. For models without the RPQ, execution caused a program interrupt that

invoked a software simulation of the SLT instruction. It was determined that in many cases a programmed instruction loop was more efficient than the SLT instruction execution, and certainly more efficient than the simulation. Version 3.0 of CP-67 eliminated much of the SCAN macro usage. Version 3.1 will eliminate all usage of the macro. Since the SLT instruction is non-privileged, those machines equipped with it will still function for the users that wish to use it in problem programs.

11. DIAGNOSE disk I/O

Since a large percentage of CMS disk I/O activity consists of a simple CCW string to read or write 829 bytes, a more compact method was chosen to support this operation to reduce I/O overhead. CMS builds its disk CCW string in the normal manner supplying all arguments and buffer addresses and then issues a "diagnose" op-code instead of a SIO. CP-67 intercepts the diagnose, builds a CCW string from a prototype and performs the I/O operation. Upon completion of the disk I/O, CP returns a condition code to CMS and then CMS may continue to run. CMS avoids the overhead of CCWTRANS, IPSW and I/O interrupt handling. Error recovery is performed by CMS in the normal (SIO) fashion.

12. Dynamic page release.

Version 3.0 of CP-67 uses dynamic page allocation for the assignment of auxiliary swapping space on drums and disks. Under many conditions (such as editing a file or a FORTRAN compilation), CMS uses a large portion of its user space. This space is always changed and hence swapped to auxiliary storage. It is possible that those pages may not be referenced for some period of time. CMS has thus implemented a "diagnose" function to interface to CP-67 for the purpose of releasing those pages no longer required, usually from X'12000' to X'3D000' inclusive. This diagnose frees pages in core (if any) and any pages assigned to swapping space. It has the advantage of shrinking the swapping space in use at the completion of CMS functions rather than at re-ipl or user LOGOUT. Certain CMS commands (LOAD and execution) cannot, of course, release the pages used, but these commands are in the minority. For the most part, CMS communicating to CP-67 via diagnose which pages can be dynamically released should improve the utilization of dynamic DASD paging space.

13. System and user statistics

Version 3.0 of CP-67 provides for some basic statistical gathering to determine system activity and provide basic user accounting information. Version 3.1 is extending these

statistics to provide some more meaningful data regarding system utilization and more consistent user statistics. For system measurement, the following data is being added to version 3.1:

- virtual SIO counts
- virtual "spooling" SIO count
- "diagnose" I/O requests
- max pages allocated to DASD
- spool buffers allocated
- user privileged instruction execution

For user measurement, the following data is being added:

- virtual SIO counts
- spool "cards" punched
- spool "lines" printed
- spool "cards" read
- pages read
- pages written
- T-disk allocation

14. Improved PROGINT

Since much of the code in PROGINT used to check for fast re-dispatch was duplicated in DISPATCH, it has been removed and replaced by a direct transfer to DISPATCH since the improved DISPATCH module can now quickly determine a machine's eligibility to run.

The code in PROGINT that dealt with the handling of privileged instruction simulation has been split into a new module for addressability purposes. This module, PRIVLGED, contains the code to perform detection of virtual machine privileged operations and to either perform the required simulation or invoke the module (for example, VIOMXEC) to perform the simulation.

15. Multiple page read/writes.

The allocation table in core for 2301 paging devices has been extended by nine words. Each word acts as an anchor point for a chain of paging tasks; one word for each of nine records per pair of drum tracks. Read or write requests for a particular record are formed into a paging task (IOTASK, control and CCW's) and chained on the appropriate allocation word. If the specified drum is running, no further action is taken until an interrupt is presented. If the specified drum is available or if paging tasks are already chained, the new paging task is chained with others if that record is not already scheduled for an I/O operation. Up to nine paging tasks can be chained together for one I/O operation. Paging

tasks are de-queued and chained together if possible after processing an interrupt from an operating drum. Figures 1 and 2 show how paging tasks are queued.

16. Miscellaneous changes

In version 3.0 of CP-67, CPINIT built a chained list of 200 save areas each 24 words in size. This number was increased dynamically after a certain number of users logged into the system. Since the number was arbitrary, a large amount of storage was unnecessarily used for save areas. Release 3.1 starts with only 35 save areas and increases the number upon demand (not number of users). The additional save areas are obtained from the FREE storage sub-pool for the desired size. The save areas are also maintained in a "pushdown" list instead of a "wrap-around" chain.

A new module called SCREDAT resides in low core and contains 10 bytes of system creation date. These 10 bytes (for example, 03/24/71) are appended to the initial CP-67 message 'CP-67 VERSION x LEVEL y'.

The code in PAGTRANS for PAGSHARE, PAGFREE and PAGOUT has been split into a separate module, PAGTR, for addressability reasons.

17. Serviceability functions

Several functions have been revised or added to improve serviceability to the system. The module CFSDBC has been rewritten to improve the formats and functions of DISPLAY, DUMP, DCP, DMCP, STORE and STCP. DISPLAY can now be performed with translation and storage keys can be displayed. The "CP-67/CMS User's Guide" contains a complete command description.

A new function called "QUERY VIRTUAL" has been added to interrogate the virtual machine configuration. The information provided gives device status and allocation data.

In order to assist in virtual machine problem determination, a trace and address stop function has been provided. The trace function can trap various interrupts and optionally trace all instructions and virtual and real CCW's on a virtual SIO. The address stop can trap instruction execution.