Cambridge Scientific Center

320-2032
September 1968

IBM
Data Processing Division

An Introduction to CP-67/CMS

L. H. Seawright, J. A. Kelch

320-2032

AN INTRODUCTION TO CP-67/CMS

L. H. Seawright

J. A. Kelch

International Business Machines Corporation

Cambridge Scientific Center

Cambridge, Massachusetts

September, 1968

320-2032
September, 1968
Scientific Center Report

AN INTRODUCTION TO CP-67/CMS

L. H. Seawright and J. A. Kelch

International Business Machines Corporation
Cambridge Scientific Center
Cambridge, Massachusetts

Abstract

CP-67/CMS is a time-sharing system that provides easy, conversational use of a terminal-oriented IBM System/360 data-processing configuration. The system has two independent components: the Control Program (CP-67) which manages the resources of a System/360 Model 67 such that remote users appear to have a dedicated System/360 at their disposal; and the Cambridge Monitor System (CMS), a conversational operating system designed to provide a wide range of capabilities through relatively simple commands at a terminal. CP-67/CMS was developed at the IBM Cambridge Scientific Center.

Users communicate with the system through commands that cause compilation, file creation, and numerous other operations. Familiarity with the entire command set is not necessary in order to use the system; typically, if the system is to be used for an occasional Fortran compilation and execution, a knowledge of only two or three commands will be sufficient. Should the user's needs be more varied, however, numerous commands are provided which support many complex operations.

Index Terms for the IBM Subject Index

Computer Systems
IBM 360-67
Time-Sharing
Virtual Systems
Conversational Computing
Interactive Computing
Remote Computing
On-Line Debugging

07-Computers
21-Programming

The following is a complete list of Cambridge Scientific Center Technical Reports. Individual copies are available upon request from IBM Corporation, Cambridge Scientific Center, 545 Technology Square, Cambridge, Massachusetts, 02139.

320-2000 Production Sequencing by Combinatorial Programming
by J. F. Pierce and D. J. Hatfield

320-2001 On the Solution of Integer Cutting Stock Problems by Combinatorial Programming - Part I
by J. F. Pierce

320-2002 Application of Combinatorial Programming to a Class of All-Zero-One Integer Programming Problems
by J. F. Pierce

320-2003 A Multi-Item Version of the Economic Lot Size Model
by J. F. Pierce

320-2004 String Processing on the System/360: Techniques and Example
by S. E. Madnick

320-2005 SPL/1: A String Processing Language
by S. E. Madnick

320-2006 An Approach to the Two Dimensional, Irregular Cutting Stock Problem
by R. C. Art, Jr.

320-2007 A Virtual Machine System for the 360/40
by R. J. Adair, R. U. Bayles, L. W. Comeau, R. J. Creasy

320-2008 Applications of Time-Shared Computers in a Statistics Curriculum
by M. Schatzoff

320-2009 Efficient Calculation of All Possible Regressions
by S. Fienberg, M. Schatzoff, R. Tsao

320-2010 An Experimental Comparison of Time-Sharing and Batch-Processing
by M. Schatzoff, R. Tsao, R. Wiig

Cambridge Scientific Center Technical Reports (continued)

320-2011 Computer Diagnosis: A Review and Discussion
by E. Hoffer

320-2012 URBAN5: An On-Line Urban Design Partner*
by N. Negroponte, L. Groisser

320-2013 A Study of the Effect of User Program Optimization in a Paging System
by L. W. Comeau

320-2014 A Second Order Exponential Model for Multidimensional Dichotomous Contingency Tables, with Applications in Medical Diagnosis
by R. F. Tsao

320-2015 CP/CMS User's Guide

320-2016 COMB User's Guide
by M. Schatzoff

320-2017 Design and Implementation of COSMOS
by M. Schatzoff, R. Tsao, T. Burhoe

320-2018 On the Truck Dispatching Problem, Part I
by J. F. Pierce

320-2019 Computational Probability
by U. Grenander, R. F. Tsao

320-2020 A Conversational Partition Monitor for OS/360 MFT
by C. Johnson, R. Mitchell

320-2022 Linguistic Tendencies in Pattern Analysis
by U. Grenander

320-2023 SCRIPT: An Online Manuscript Processing System
by S. E. Madnick, A. Moulton

*Ekistics, Vol. 24, #142, Sept. 1967, pp. 289-291

Cambridge Scientific Center Technical Reports (continued)

320-2024 Mass Storage Software Simulated Associative Memory for PL/I Graphics
by A. J. Symonds

320-2025 Experimental System for Graphics in PL/I
by C. I. Johnson

320-2026 A Feature Logic for Clusters
by U. Grenander

320-2027 Multi-Processor Software Lockout
by S. E. Madnick

320-2028 The Scattering of Sound by a Gas Bubble In an Elastic, Viscous Medium
by J. W. Horton

320-2029 A Formulation of the Carotid-Artery Baroceptor Transducer Problem
by J. W. Horton

320-2030 On the Solution of Integer Cutting Stock Problems by Combinatorial Programming - Part 2
by J. F. Pierce

320-2031 Pipe Network Analysis in Integrated Civil Engineering Systems (ICES)
by K. T. H. Liu

320-2032 An Introduction to CP-67/CMS
by L. H. Seawright and J. A. Kelch

320-2033 Multi Access Systems - The Virtual Machine Approach
by M. S. Field

TABLE of CONTENTS

I. Components of the System

The CP-67/CMS time-sharing system consists of two independent components: the Control Program (CP-67) and the Cambridge Monitor System (CMS). The Control Program creates the time-sharing part of the system to allow many users to simultaneously perform work. The Cambridge Monitor System provides the conversational part of the system to allow a user to monitor his work in a conversational manner.

Both components are independent of each other. CP-67 can be used on an appropriate configuration without CMS and CMS can be run on a properly configured System/360 as a single-user system without CP-67 (see Appendix A and B for appropriate configurations). If CP-67 is used without CMS, an operating system or systems must be chosen to provide the conversational or production aspect of the system as CP only provides the time-sharing capability.

CP-67 is capable of running any System/360 operating system (including OS/360 and DOS) as long as that system does not include any timing dependencies or self-modifying I/O sequences. CP-67 is also capable of running any System/360 operating system along with CMS in a multi-programming mode concurrent with its usual time-shared, multi-access operation. If the System /360 operating system contains telecommunication facilities, CP will allow that system to control the 2702 transmission control unit and the user to DIAL into that system.

II. System Environment

The environment of CP-67 is one of "virtual machines". A virtual machine is a functional simulation of a real computer and its associated I/O devices. CP-67 builds and maintains for each user a virtual System/360 machine

from a predescribed configuration. The virtual 360 is indistinguishable to the user and his programs from a real System/360, but it is really one of many that CP-67 is managing. CP allocates the resources of the real machine to each virtual machine, in turn, for a short "slice" of time, then moves on to the next virtual machine - thus, time-sharing.

Since the virtual machines are simulated, their configurations may differ from each other and from the real machine. For instance, the real machine may have 512K and eight disk drives and the virtual machine can have 768K and two disk drives. One virtual machine may have a virtual 2702 and run an OS-based system and another virtual machine that does not have a virtual 2702 may run CMS. One virtual machine may have a remote printer and a remote card punch while another virtual 360 may have a dedicated printer and 2250. Regardless of the configuration, each user controls his virtual machine from his terminal, which is, effectively, his console keyboard.

Like real machines, virtual machines will operate most efficiently under an operating system. The Cambridge Monitor System (CMS) is designed to allow full use of a System/360 through a simple command language entered at the console (in the case of a virtual machine, at the terminal). CMS gives the user a full range of capabilities -- creating and managing files, compiling and executing problem programs, and debugging -- using only his remote terminal. Since each user has his own virtual machine with his own copy of CMS residing "in it", nothing he does can affect any other user; if he destroys the CMS nucleus or abends the CMS system, he can re-IPL his virtual machine and continue without disturbing other users. In addition, since users cannot get "outside" their virtual machines, CP-67 is protected from any user error. CMS also provides a batch monitor for compile, load, and go type jobs coming from

tape and cards. The batch monitor can be run from a virtual machine as a background system with conversational CMS users on other virtual machines.

III. The Control Program - 67

Before a user is authorized to use CP, he must be assigned a USERID, which identifies him to the system, and a password, which is checked when he "logs in". Associated with each USERID is a table describing the virtual machine assigned to that user. Whenever he logs in, CP sets up this virtual 360 machine for him. Although all the virtual machines may be different, most will be set up with the configuration expected by CMS, the most commonly used operating system. They will include at least 256K bytes of core storage, two disk drives, a console (the terminal), a card-read-punch unit, and a printer. The real system will usually have a larger number of disk drives and/or a drum, tape drives, and perhaps more core storage.

Because there is not room in real core for all users' virtual core, a technique called "paging" is used by the system. Virtual core is divided into 4096-byte blocks of storage called "pages". All but currently active pages are kept, by the system, on direct access secondary storage; as active and inactive pages change status they are "paged" in and out of real core on a demand basis. While the paging operation is being perfomed for one virtual machine, another virtual machine can be operating. The paging operation, and resultant allocation of real core to a given user's pages, appear random to the system. Special hardware is provided on the System/360 Model 67 that translates, at execution time, the user's (or user program's) addresses into the current real addresses of the randomly located pages. This is called "dynamic address translation"; it is transparent to the user.

Because of the virtual machine concept employed by this system, only the Control Program (CP) may operate in the supervisor state on the real machine. All programs other than CP - that is, all programs executing on virtual machines — operate in the problem state on the real machine. By a special interrupt-handling procedure, however, CP supports what amounts to a virtual supervisor state on the virtual machine. All user interrupts, including those caused by attempted privileged operations, are handled by CP, which then reflects to the user program only those the user program would expect from a real machine. The user may expect his programs to execute on his virtual machine in a manner identical to their execution on a real System/360.

All virtual machine I/O operations are handled by CP, which must translate them into real machine I/O operations. This requires two translations, accomplished as follows: CP intercepts all user I/O when Start I/O is issued. It translates virtual device addresses into real device addresses, translates virtual core storage addresses into real core storage addresses, ensures that all necessary pages are in real core storage, builds a CCW string for the user, and issues SIO when the channel is free. When CP receives an interrupt indicating I/O completion, it sets a "ready-to-run" flag in the user's virtual machine status table; when control is returned to the virtual machine, the proper I/O interrupt is simulated. The virtual machine is not given control from the time it issues an SIO until CP delivers the simulated I/O interrupt. In the meantime, another virtual machine(s) may be operating.

All virtual machine unit record I/O is spooled onto disk by CP. Thus, any card deck to be "read" by a virtual machine must have been read by CP prior to the user's call for it on his virtual machine or transferred to that user from another user's files via the XFER console function in CP; the physical deck must have been preceded by a card containing the USERID, so that CP can

know who the card-image file is for. Later, when the virtual machine has "read" the card deck, a card reader end-of-file is simulated. Card and printer output, similarly spooled, is not queued for physical output until CP is notified of end-of-file in one of three ways: the user logs off the system (end-of-file is assumed); the CLOSE console function specifies the (virtual) address of the device to be closed; or CP detects an invalid CCW addresses to the device (end-of-file is assumed). Further output for a closed device is assumed to start a new file. So that the system operator can separate physical output, printed and punched output files are always preceded by a record (supplied by CP) that contains the USERID.

The CP console functions allow the user to control his virtual machine from the terminal much as an operator controls a real machine. To perform an IPL, for instance, the user types "IPL" and a device address or the name of a "named" operating system, such as CMS. The user can stop his virtual machine at any time (by depressing the ATTN key) and request display of any portion of his storage and registers. He can modify the contents, if desired, and restart his machine. CP also recognizes a few special purpose commands, such as the XFER function mentioned above, the QUERY function to obtain the number of users on the system and their USERID's, the MSG function to communicate with other users, the TIMER function to control the interval timer, the DIAL function to connect the terminal to a telecommunications-based operating system, and the ATTACH and DETACH functions to add or remove I/O devices from a virtual machine configuration (ATTACH can only be issued by the OPERATOR). See Section V for a brief description of the CP-67 console functions.

IV. The Cambridge Monitor System

The Cambridge Monitor System (CMS) is a single-user, conversational operating system, capable of running on a real machine as well as on a virtual machine. It interprets a simple command language typed in at the operator's console (in this case, the user's remote terminal).

Whether running on a real or a virtual machine, CMS expects the following machine configuration:

device	virtual address	symbolic name	
1052	009	CON1	console
2311	190	DSK1	system disk (read-only)
2311	191	DSK2	permanent disk (user files)
* 2311	192	DSK3	temporary disk (work space)
1403	00E	PRN1	line printer
2540	00C	RDR1	card reader
2540	00D	PCH1	card punch
* 2400	180	TAP1	tape drive
* 2400	181	TAP2	tape drive
At least 256K bytes of core storage.			

* The 2311 for the temporary disk and the two 2400 tape drives are optional devices; they are not included in the minimum configuration.

Under CP, of course, these devices are simulated and remapped to different addresses and/or different devices. For instance, CMS expects a 1052 printer-keyboard operator's console, but most remote terminals are 2741's; CP handles all channel program modifications necessary for this simulation.

Under CP, all CMS users share the read-only system disk; on it reside the CMS nucleus routines, which the user IPL's, and the other routines and libraries that CMS calls as needed. The CMS nucleus is also shared in core among users under CP, but users may modify the CMS nucleus as they wish without affecting either the system disk version or the copies in other virtual machines. If a user modifies the CMS nucleus, he will no longer share the nucleus with other users but will obtain his own copy of the nucleus.

File Management Under CMS:

Each CMS user is assigned two disks (a third disk is optional), one of which is shared with other CMS users. These disks, under CP, are seldom complete disk packs. At the time a user is authorized to use CP-67/CMS, the size of each disk area is set by the system administrator, according to the needs of the user and the total amount of disk space available.

The shared disk contains the CMS nucleus, which is loaded into the virtual machine by the IPL console function. Also on this disk, referred to as the "system (SY) disk" are CMS commands which are not core-resident, and system libraries of routines and macros. No user may write on this disk as it is read-only. Any attempt to modify any of the system files results in an error message.

The two other disks are known as the "permanent" and "temporary" disks. The user does not normally share these disks with any other user, as they are accessible only to him after he has logged in with the correct USERID and password. The permanent disk is used for files which are to be saved from one terminal session to the next. The temporary disk, which is

optional, provides space for work files which need not be retained between sessions. This disk is erased whenever the user logs out.

If a user knows the correct password, additional permanent disks can be concatenated or "linked" to his permanent disk to allow file-sharing among users.*

CMS uses a three-field file identification to catalog both system and user files. The fields are referred to as the filename, filetype, and filemode of the file. Uniqueness of any one of the fields is sufficient to differentiate a file from other files. CMS maintains a directory (the "User File Directory") of each user's files, which includes information on the file format, size, and location. This allows the user to specify files by using only the file identification or a portion of it.

All CMS disk files are written in 800-byte physical records. The system I/O routines handle packing of logical records into this format. The record blocks are written onto the user's disk area in random order. CMS maintains chains of disk addresses to keep track of the files. These chains are linked to the User File Directory, which has an entry for each user file. The directory is brought into storage when the user logs in, and is updated whenever files are used. Periodically, and at least as often as once per command, the updated file directory in core is written out onto disk, so that the permanent copy is as current as possible. This insures an accurate directory if it is necessary to re-IPL CMS during a terminal session.

The directory handles files up to 12.8 million bytes in length, which is 56 cylinders of a 2314 disk pack and is beyond the capacity of an entire 2311 disk pack. In practice, the user's disk will not normally require files of that length, since the typical user uses less than 25 cylinders. Whenever CMS de-

*File-sharing is currently being debugged.

tects that only a few tracks are left on the user's disk, a warning message is typed, and the files currently open are closed. A program or command in execution is halted, so the user may create more free space on the disk by erasing some files, or copying them from disk to other media.

Although most of the CMS commands operate on disk-resident files, the user also has access to the card-read-punch, printer, and tape drives. The commands, in general, create sequential files of fixed-length records; however, the programmer using the CMS I/O support routines is able to use any record format with either fixed-length or variable-length records.

Files are automatically "opened" for reading or writing when the first read or write is issued. Only eight files may be open at the same time. CMS routines automatically close files after every command, and after user programs that complete normally. If a user program needs to access more than eight files during execution, the FINIS command must be called to close some files. Files must also be closed between writing and reading.

CMS Commands:

CMS commands fall naturally into four categories: file manipulation, compilation, execution control, and debugging aids.

The file handling commands allow the user to create, copy, move, combine, update, print, compare, and erase disk files. Other commands provide access to the tape units, printer, and card-read-punch. Under the CMS linkage scheme, all of these commands are available to executing programs as well as to the user at the terminal.

The CMS language processors are the same ones used under Operating System/360 (OS); these include Assembler (F), FORTRAN IV (G), and PL/I(F). The Assembler produces object programs that may be executed under either CMS or OS, depending on the macros used in the source program. Special file-

handling routines for macro libraries are included. The FORTRAN and PL/I compilers also produce OS-compatible object programs. (The FORTRAN execution – time support programs have been modified for CMS.) The SNOBOL compiler and assembler-interpreter were adapted from programs designed to execute under OS.

The execution control commands allow the user to load his programs from single object decks called TEXT files (the filetype TEXT is reserved for relocatable object programs) or from a library of programs. He can pass a list of parameters to his program from the terminal, and specify the point at which execution is to begin. To avoid relocation (bypass the relocating loader) he can create a file consisting of an image of the portion of core storage containing his program, and load that non-relocatable copy back at any time. Since the loading commands can be accessed by executing programs, overlay structures may be set up. The user can also create a file which is a series of commands, and then execute these commands by typing a single line; logic statements can also be placed in the file with the commands such that if errors occur from a command, no more commands will execute from that file.

The debugging command in CMS is called DEBUG. It allows the user to stop his programs at pre-determined points and examine his registers, PSW, and storage, and modify these if he so desires. This information may be typed out at his terminal or printed offline. A program interrupt gives control to DEBUG, as does the external interrupt caused by the EXTERNAL console function. The user may also employ the program tracing routines, which record all SVC transfers, or just those SVC's in which an error return is made.

There are other miscellaneous commands which give the user the facility to terminate the typeout at his terminal, to shorten the length of the line typing at the terminal, and to kill program execution. The user can also obtain statistics on his file space and share additional permanent disks with other users.

Each of the CMS commands is described briefly in Section VI.

A sample terminal session for CMS running under CP-67 is shown and described in Section VII.

CMS Batch Monitor:

As well as being a conversational monitor, CMS provides a batch facility for running compile, load, and go type jobs. The CMS batch monitor accepts a job stream from a tape unit or the card-reader and writes the output either on tapes, the printer, or the card-punch. The job stream can consist of a System/360 Operating System SYSIN job stream with FORTRAN (G), Assembler (F), and PL/I (F) compile load and go jobs calling cataloged procedures or it can consist of CMS commands along with control cards and card decks for compile, load, and go jobs for the supported compilers.

Just as the conversational CMS does, the batch monitor can run from either a virtual machine or a real machine. Under CP, it can be used as a background monitor along with other conversational CMS users.

To eliminate the possibility of one job modifying the CMS nucleus in such a way as to effect the next job, CMS is re-IPLed before each job begins. Files can also be written onto the batch monitor's permanent disk and then punched or printed, such as files written by Fortran programs; these files should be of limited size and considered as temporary, as they are erased between each job control card.

V. Control Program Console Functions

Each of the CP-67 console functions that can be issued by a user from the terminal is described below.

BEGIN	begins execution at the specified address or, if no address is given, at the location at which execution was interrupted.
CLOSE	releases the spooling areas containing input from the card reader or output to the printer or card punch.
DETACH	removes the specified device from the user's virtual machine configuration.
DIAL	is used in place of LOGIN to connect a user's terminal with a virtual telecommunications operating system or a virtual time-sharing system.
DISPLAY	types at the terminal the contents of the specified register(s), core location(s), or program status word.
DUMP	prints the contents of the specified register(s), core location(s), or program status word on the offline printer.
EXTERNAL	simulates an external interrupt to the virtual machine, causing control to pass to the CMS DEBUG command when CMS is entered.
IPL	simulates the Initial Program Load sequence on the specified unit.
LOGOUT	releases the user's virtual machine, including his temporary disk area, and closes any spooling areas which have not been released.
MSG	types out the specified message at the terminal of the person whose USERID is specified.
QUERY	types out either the number of users logged onto the system, the names of these users, or the maximum number of users allowed to log on.
READY	simulates a device end for the specified unit.
RESET	simulates the system reset key on the 360 console by re-setting any pending I/O interrupts.
SET	controls the saving of virtual card-reader files and the typing of messages at the terminal.
STORE	replaces the contents of the specified register(s), core location(s), or program status word with the specified information.

TIMER	controls whether real time or CPU time is maintained for the virtual machine.
XFER	controls the passing of files between users.

VI. CMS Commands

Each of the commands that can be issued by a user to CMS are described below.

ALTER	changes all or part of the identifier (filename, filetype, and filemode) of a file stored on the user's permanent or temporary disk without altering the contents of the file.
ASSEMBLE	converts assembler language source code into relocatable object code using the OS/360 F level assembler.
CLOSIO	signals the Control Program that I/O to offline unit record equipment has been completed and that the spooling areas for this I/O may be processed. CLOSIO is generally issued automatically by the commands which access unit record equipment.
CLROVER	clears overrides set by the SETERR and/or SETOVER commands and causes all recorded trace information to be printed on the offline printer.
COMBINE	copies the specified file(s), concatenating them in the order given, into a new file which is placed on the user's permanent or temporary disk and assigned the specified identifier.
DEBUG	allows the user to stop and re-start programs at specified points and to inspect and change the contents of registers, core locations, and hardware control words online.
DISK	causes a CMS disk file to be punched out or read in from cards which are in CMS card format.
DUMPREST	dumps the contents of an entire disk to magnetic tape or restores the contents of an entire disk from magnetic tape.
ECHO	tests terminal line transmission by repeating as typeout whatever is typed in by the user.
EDIT	allows the user to create card-image files on disk and to make changes to existing files from his terminal.

ERASE	deletes the entry for a specified file (or files) from the appropriate directory, rendering the file inaccessible to the user, and freeing the disk area containing that file.
EXEC	executes a file containing one or more CMS commands, allowing a sequence of commands to be executed by issuing a single command.
FINIS	closes the specified file (or files) by writing the last record of that file on disk, updating the User File Directory, and removing the entry for that file from the user's table of active files.
FORMAT	prepares the user's permanent or temporary disk area for CMS use by writing blank records over the currently stored information.
FORTRAN	converts Fortran language source code into relocatable object code using the OS/360 FORTRAN G compiler.
GENMOD	creates a non-relocatable core-image file on the user's permanent disk which is a copy of the contents of core between two given locations.
GLOBAL	specifies (1) macro definition libraries to be searched during the assembly process or (2) text libraries to be searched when loading files containing relocatable object code.
KE	truncates information currently being typed at the terminal to 72 characters per line. This truncation will remain in effect for the duration of the currently executing command or user program.
KO	clears overrides previously set by the SETOVER or SETERR commands and causes all trace information recorded by these commands to be printed on the offline printer.
KT	stops typeout at the terminal for the duration of the currently executing command or user program.
KX	terminates the currently executing program; updates the User File Directory, and logs out from CMS, transferring control to the Control Program.
LISTF	either types out at the terminal the identifier and size of the specified disk file(s), or creates a file on the user's permanent disk containing information for use by the EXEC and/or $ commands.

LOAD	reads the specified TEXT file(s) -- containing relocatable object code -- from disk, loads them into core, and establishes the proper linkages.
LOADMOD	reads a MODULE file -- which is in non-relocatable core-image form -- from disk and loads it into core.
LOGIN	causes the user's permanent disk files to be either saved or deleted, as specified. If LOGIN is not issued, the files will be saved.
LOGOUT	compacts the User File Directory, executes any CMS command specified as an operand, and logs out of CMS, transferring control to the Control Program.
MACLIB	generates or adds to a specified macro library, or types out the contents of the dictionary of that library.
OFFLINE	creates a disk file from card input, prints a disk file on the offline printer, or punches a disk file on cards.
PLI	converts PL/I language source code into relocatable object code using the OS/360 PL/I F compiler.
PRINTF	types at the terminal the contents of all or part of a specified disk file.
REUSE	reads the specified TEXT file(s) -- containing relocatable object code -- from disk and loads them into core, establishing linkages with previously loaded files and changing the default entry point of these files to that of the first file specified in the REUSE command.
SCRIPT	either (1) allows the user to create arbitrary alpha-numeric text files on disk and to make changes to existing files of this type from the terminal or (2) types out the contents of the specified file, formatting it as indicated by control words contained in the text.
SETERR	sets error overrides which will cause trace information to be recorded for each SVC-called program which returns with an error code in general purpose register 15.
SETOVER	sets normal and error overrides which will cause trace information to be recorded for all SVC-called programs -- both those which are executed normally and those which return an error code in general purpose register 15.

SNOBOL	converts a card-image file in Snobol source language into SPL1 interpreter language and executes SPL1 programs.
SPLIT	copies the specified portion of a card-image file and appends it to a second specified card-image file.
START	begins execution of the loaded programs(s) at the specified or default entry point and passes the address of a string of user arguments to the program(s).
STAT	types statistics regarding the amount of permanent and/or temporary disk space used, or compacts the User File Directory.
TAPE	writes the contents of CMS disk files of any type or size onto magnetic tape, or restores these files by writing them from tape onto disk.
TXTLIB	either (1) generates or adds to a specified text library, (2) types out the contents of the dictionary for that library, or (3) creates a file containing a list of entry points and control section names contained in that library.
UPDATE	updates the specified disk file with a file containing control cards, where each control card indicates whether the information immediately following it is to be resequenced, inserted, replaced, or deleted.
USE	reads the specified TEXT file(s) -- containing relocatable object code -- from disk and loads them into core, establishing linkages with previously loaded files.
$	executes a file containing one or more CMS commands, or loads into core a file which is in either core-image form or relocatable object code and begins execution of that file.

VII. CP-67/CMS Terminal Session

A sample terminal session is given on the following pages. User input is in lower case; typeout from the system appears in upper-case. The following is a description of the terminal session:

After logging in to CP and utilizing CMS, a LISTF command is issued to obtain a list of all files stored on user CSC1's permanent disk. The file "MAIN FORTRAN" is then created and filed on the user's permanent disk. Compilation of the file is terminated due to program errors (indicated by a $ symbol below the error encountered). The file is then modified and edited to correct the line in error, and the new source file stored on disk. Again an error is encountered and the file re-edited.

After a successful compilation, the $ command is called to load the file into core and execute it. LOAD and START perform the same function as $, as shown. Specifying the XEQ option with the LOAD command will also cause execution to begin after the file is loaded.

LISTF and ERASE commands are used to selectively list and erase files, and the PRINTF command is used to print all and then part of the contents of a file. KT causes typeout to be discontinued if entered after the attention key is hit twice.

The OFFLINE command punches or prints the specified file on offline devices. STAT types out statistics regarding the amount of disk space used and remaining. The ALTER command changes the identifier of a file. KX, entered after hitting the attention key twice, stops execution of the current program and returns control to CP.

A new copy of CMS is obtained by issuing the IPL console function and an EXEC file (consisting of CMS commands) is created and filed. The file is then executed by issuing the EXEC command, which will cause each of the commands contained in the file to be executed individually. Operand substitution is illustrated by modifying and re-executing the file using & arguments.

Hitting the attention key once transfers control to the Control Program where the QUERY console function is issued to determine the number of users on the system, their names, and the message of the day from the operator. The BEGIN console function then returns control to CMS and the user logs out from both CMS and CP.

```
#
CP/67 Online     XDh65 Qsyosu

1   csc1                          The user's id is specified upon logging in.
ENTER PASSWORD::
                                  <-------The protected password does not
CP WILL BE UP UNTIL 1500 CONTINUOUSLY.  print when entered.
READY AT 09.55.29 ON 04/12/68

ipl cms
CMS...VERSION 4.0 - 04/01/68

listf
FILENAME FILETYPE MODE NO.REC.
INDIAN   LISTING   P1   003
DUMPREST SYSIN     P1   009
SUPERSCR SYSIN     P1   070
MY       FORTRAN   P1   001
INDIAN   TEXT      P1   002
FORTCLG  EXEC      P1   001
LOAD     MAP       P5   003
DUMPREST SYSUT1    P1   019
DUMPREST SYSUT2    P1   019
FIN      SCRIPT    P1   001
TUES     SCRIPT    P1   001
FRST     SCRIPT    P1   001
DUMPREST SYSUT3    P1   001
DUMPREST LISTING   P1   007
AGENDA   SCRIPT    P1   001
INDIAN   FORTRAN   P1   001
R; T=0.06

edit main fortran
FILE DOES NOT EXIST; WILL BE CREATED.
INPUT:
c main program  April 12, 1968
      write (6,10)
10#format (' a = ')<-------- The # is a logical tab character that inserts
#read (5,20) a               blanks and places the following characters
20#format (8.3)              typed on the line into column 7 of the card
#write (6,25) a,x            image in a FORTRAN file.
#end

EDIT:
file
R; T=0.11

fortran main
     0004        20    FORMAT (8.3)
                                 $
            01)  ERR 13  SYNTAX
COMPILATION CANCELLED DUE TO SOURCE PROGRAM ERROR(S).
E(00032); T=0.32
```

```
edit main fortran
EDIT:
p 20

C MAIN PROGRAM  APRIL 12, 1968
      WRITE (6,10)
10    FORMAT (' A = ')
      READ (5,20) A
20    FORMAT (8.3)
      WRITE (6,25) A,X
      END
EOF REACHED BY:
P 20
1 /format/
10    FORMAT (' A = ')
1 /format/
20    FORMAT (8.3)
c /8/f8/
20    FORMAT (F8.3)
u 2
10    FORMAT (' A = ')
c / '/ ?'/
10    FORMAT (' A = ?')
f 20
20    FORMAT (F8.3)
i #x = a**2

EDIT:
p
      X = A**2
t
p 20

C MAIN PROGRAM  APRIL 12, 1968
      WRITE (6,10)
10    FORMAT (' A = ?')
      READ (5,20) A
20    FORMAT (F8.3)
      X = A**2
      WRITE (6,25) A,X
      END
EOF REACHED BY:
P 20
file
R; T=0.51

fortran main
                                    ERR 22     UNDEFINED LABEL
      25
COMPILATION CANCELLED DUE TO SOURCE PROGRAM ERROR(S).
E(00032); T=0.40
```

```
edit main fortran
EDIT:
1 /25/
      WRITE (6,25) A,X
i 25#format (' a = 'f8.3,' x = 20.3)
p
25    FORMAT (' A = 'F8.3,' X = 20.3)
c /20/' f20/
25    FORMAT (' A = 'F8.3,' X = ' F20.3)
file
R; T=0.14

$ ma¢ <-------------------The ¢ deletes the line.
fortran main
R; T=0.46

$ main
EXECUTION BEGINS...
 A = ?
2.5
 A =    2.500 X =                    6.250
R; T=0.33

load main
R; T=0.25

start
EXECUTION BEGINS...
 A = ?
3.1
 A =    3.100 X =                    9.610
R; T=0.05

load main (xeq)
EXECUTION BEGINS...
 A = ?
3.2
 A =    3.200 X =                   10.240
R; T=0.28

listf main *
FILENAME FILETYPE MODE NO.REC.
MAIN     LISTING   P1   003
MAIN     FORTRAN   P1   001
MAIN     TEXT      P1   002
R; T=0.03

listf * listing
FILENAME FILETYPE MODE NO.REC.
INDIAN   LISTING   P1   003
MAIN     LISTING   P1   003
DUMPREST LISTING   P1   007
R; T=0.01

erase * listing
R; T=0.03
```

```
listf * listing
FILE NOT FOUND
E(00002); T=0.03

printf main fortran

C MAIN PROGRAM  APRIL 12, 1968
      WRITE (6,10)
10    FORMAT (' A = ?')
      READ (5,20) A
20    FORMAT (F8.3)
      X = A**2
      WRITE (6,25) A,X
25    FORMAT ('A = 'F8.3,' X = ' F20.3)
      END

R; T=0.06

printf main fortran * 3 25

C MAIN PROGRAM   APRIL 12
      WRITE (6,10)
10    FORMAT (' A = ?')


R; T=0.04

printf main fortran

C MAIN PROGRAM    APRIL 12, 1968
      WRITE (6,10)
10 "
                  <------------------------The ATTN key was hit twice to
kt                                        enter KT for killing the typeout.
R; T=0.06

offline punch main text@@@@fortran <------The four @ characters delete the
R; T=0.07                                   previous four characters.

offline print main fortran
R; T=0.06

offline printcc main listing
FILE NOT FOUND
E(00002); T=0.02

listf
FILENAME FILETYPE MODE NO.REC.
DUMPREST SYSIN     P1   009
SUPERSCR SYSIN     P1   070
MY       FORTRAN   P1   001
FORTCLG  EXEC      P1   001
LOAD     MAP       P5   003
MAIN     FORTRAN   P1   001
DUMPREST SYSUT1    P2   019
DUMPREST SYSUT2    P1   019
FIN      SCRIPT    P1   001
TUES     SCRIPT    P1   001
FRST     SCRIPT    P1   001
DUMPREST SYSUT3    P1   001
AGENDA   SCRIPT    P1   001
MAIN     TEXT      P1   002
INDIAN   FORTRAN   P1   001
R; T=0.05
```

```
stat
P-DISK:  0142 RECORDS IN USE, 0258 LEFT (of 0400), 36% FULL (of 010 CYL.)
R; T=0.02

alter main fortran * mainone * *
R; T=0.02

listf main fortran
FILE NOT FOUND
E(00002); T=0.03

listf * fortran
FILENAME FILETYPE MODE NO.REC.
MY       FORTRAN   P1   001
MAINONE  FORTRAN   P1   001
INDIAN   FORTRAN   P1   001
R; T=0.01

$ main
```

The ATTN key was hit twice to enter KX for killing execution of $.

```
kx
KILLING CMS EXECUTION...
P-DISK: 0142 RECORDS IN USE, 0258 LEFT (of 0400), 36% FULL (of 010 CYL.)
TOTAL CPU-TIME (IN SECONDS) = 11.18
CP ENTERED,  READY.

ipl cms
CMS...VERSION 4.0 - 04/01/68

listf mainonn@e *
```

The @ deletes one character.

```
FILENAME FILETYPE MODE NO.REC.
MAINONE  FORTRAN   P1   002
R; T=0.03

edit fortclgo exec
FILE DOES NOT EXIST; WILL BE CREATED.
INPUT:
fortran mainone
$@load mainone (xeq)

EDIT:
file
R; T=0.10

printf fortclgo exec

FORTRAN MAINONE
LOAD MAINONE (XEQ)

R; T=0.04

exec fortclgo
FORTRAN MAINONE
LOAD    MAINONE (XEQ)
EXECUTION BEGINS...
```

```
 A = ?
3.4
 A =    3.400 X =                    11.560
R; T=0.86

edit fortclgo exec
EDIT:
c /mainone/ &1/ * g
LOAD &1 (XEQ)
EOF REACHED BY:
C /MAINONE/ &1/ * G
file
R; T=0.11

exec fortclgo mainone
FORTRAN MAINONE
LOAD    MAINONE (XEQ)
EXECUTION BEGINS...
 A = ?
5.1
 A =    5.100 X =                    26.010
R; T=0.89

edit fortclgo exec
EDIT:
i &set err exit
p 9
&SET ERR EXIT
FORTRAN  &1
LOAD &1 (XEQ)
EOF REACHED BY:
P 9
file
R; T=0.11

edit mainone fortran
EDIT:
p 4

C MAIN PROGRAM   APRIL 12, 1968
      WRITE (6,10)
10    FORMAT (' A = ?')
b aa
      FORMAT ('A = ?')
file badone
R; T=0.15

listf * fortran
FILENAME FILETYPE MODE NO.REC.
MY       FORTRAN   P1   001
MAINONE  FORTRAN   P1   001
INDIAN   FORTRAN   P1   001
BADONE   FORTRAN   P1   001
R; T=0.03
```

```
exec fortclgo badone
FORTRAN BADONE
    0002                FORMAT (' A = ?')
                                $
            01)  ERR 02   LABEL
COMPILATION CANCELLED DUE TO SOURCE PROGRAM ERROR(S).
 !!! E(00032) !!!
R; T=0.44

edit fortclgo exec
EDIT:
c /&1/&1 &2 &3 &4 &5/ *g
FORTRAN  &1 &2 &3 &4 &5
LOAD  &1 &2 &3 &4 &5 (XEQ)
EOF REACHED BY:
C /&1/&1 &2 &3 &4 &5/ * G
file
R; T=0.12

exec m@fortclgo mainone
FORTRAN MAINONE
LOAD    MAINONE (XEQ)
EXECUTION BEGINS...
 A = ?
1.9
 A =    1.900 X =                  3.610
R; T=0.93

            <-------------------The ATTN key was hit once to enter CP.
q user
10 USERS

q user names
CURRENT USERS ARE...
 MADNICK    ON 029
 MARK       ON 04C
 MEYER      ON 047
 OPERATOR   ON 009
 BAYLES     ON 049
 CJONES     ON 020
 ROSATO     ON 027
 WHJ        ON 024
 SEYMOUR    ON 045
 LOVE       ON 028

q logmsg
CP WILL BE UP UNTIL 1500 CONTINUOUSLY.

begin
CMS         <-------------------BEGIN returns control to CMS.

logout
CMS LOGGING OUT...
TOTAL CPU-TIME (IN SECONDS) = 2.99
CP ENTERED, READY.

]
LOGOFF AT 12.06.06 ON 04/12/68
```

APPENDIX A

Devices Supported by CP-67

The minimum configuration required for CP-67 is as follows:

2067-1 or 2067-2 Central Processing Unit
2365 Core Storage Unit
1052 On-Line Console Typewriter
1403 Printer
2540 or 2501 Card Reader
2540 Card Punch
2 2311 Disk Drives or 1 2314 Storage Unit
2702 Transmission Control Unit

Required features:

#3233 15 Data-Set Adaptor
#4615 IBM Terminal Ctl Type 1
#8055 2741 Break (if 2741's used)

RPQ #E-46765 Break Command (if terminals used)

The devices that are supported in addition to the minimum configuration are described below:

1051/1052 Model 2 Data Communications System

Required features:

#26903 Receive Interrupt Control
#27428 Transmit Interrupt Control
#5050 Master Station

2741 (-1, 2) Communications Terminals

Required features:

#4708 Interrupt
#3255 Dial-up

RPQ #E-40681 Receive Interrupt

Desirable feature:

RPQ #E-46151 Print Inhibit

2250 Graphic Display
2820/2301 Drum Storage Controller/Unit
2303 Drum Storage Unit
2400 Series Magnetic Tape Drives

APPENDIX B

Devices Supported by CMS

The minimum configuration for CMS is any System/360 with the following specifications:

256K Core Storage
1052 On-Line Console Typewriter
1403 Printer
2540 Card Reader/Punch
2 2311 Disk Storage Drives or 1 2314 Disk Storage Unit

CMS will also support the following additional devices:

2 2400 Series Tape Drives
1 additional 2311 Disk Storage Drive or equivalent 2314 Disk Storage Unit

BIBLIOGRAPHY

For further information on the CP-67/CMS system and the virtual machine concept, refer to the following publications:

Adair, R.J., R.U. Bayles, L.W. Comeau, and R.J. Creasy, A Virtual Machine System for the 360/40, IBM Cambridge Scientific Center Report 320-2007, Cambridge, Massachusetts, May 1966.

Dorn, W.S., A FORTRAN Programmer's Introduction to CMS, IBM T.J. Watson Research Center Report #RC1942, Yorktown Heights, New York, November 1967.

_____, CP-67/CMS, Type III Documentation, Program #360D-05.2.005, IBM Corporation, DP Program Information Department, 40 Saw Mill River Road, Hawthorne, New York, May 1968.

_____, CP-67/CMS User's Guide, IBM Cambridge Scientific Center Report 320-2015, Cambridge, Massachusetts, October 1967.

IBM