System/370 capability in a desktop computer

by F. T. Kozuh
D. L. Livingston
T. C. Spillman

A desktop computer with System/370 capability was produced by enhancing the IBM Personal Computer XT with additional hardware and developing software that provides a compatible interface. The computer, the IBM Personal Computer XT/370, and this software allow users to run most System/370 Conversational Monitor System application programs unaltered in a desktop environment. The evolution of the development and details of the function of the hardware and software are described.

Large investments have been made by many computer users in System/370 software and in the skills needed to develop, maintain, and use such software. IBM's recently announced Personal Computer XT/370 (PC XT/370) running Virtual Machine/Personal Computer (VM/PC) software is a big step towards using System/370 software in microprocessor-based desktop computing systems. VM/PC is a single-user control program which provides an interface that is compatible with the Conversational Monitor System (CMS) and capable of supporting most existing CMS application programs. The application programs execute on the small, relatively inexpensive desktop computer just as they would on a larger System/370 mainframe and include user programs, editors, document composers, language processors, etc.

The PC XT/370 was produced by enhancing the hardware of the IBM Personal Computer XT (PC XT) with three cards: a communications card, a 512K memory card, and a coprocessor card. Together these cards provide the System/370 instruction execution capability along with the ability to connect to System/370 host systems. VM/PC not only provides the necessary interface to support CMS programs, but also provides a means of coupling its CMS environment with a CMS environment running in a host processor and the ability to transfer files between the IBM Personal Computer Disk Operating System (PC DOS) environment and the VM/PC CMS environment.

In this paper, the evolution of the PC XT/370 and VM/PC is first described. The remainder of the paper provides an overview of their function.

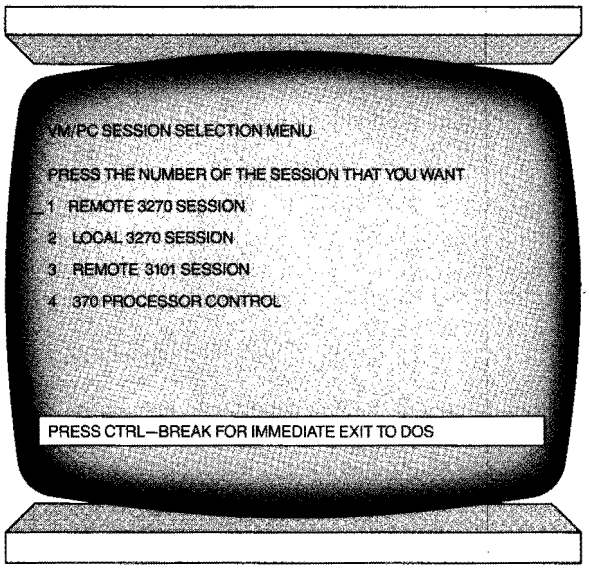
Beginnings

PC XT/370 and VM/PC evolved independently prior to February 1982.

Hardware development for the PC XT/370 began as an experiment to implement the System/370 architecture in a microprocessor environment. After studying several types of microprocessors to identify one architecturally suitable as a base for System/370, IBM selected the Motorola MC68000 microprocessor and began working with Motorola engineers to develop a customized microprocessor. At IBM's site in Endicott, New York, a group in the engineering organization wrote the internal microcode which allowed the device to directly execute a large subset of the commercial System/370 instructions. It was determined that the remaining general-purpose instruc-

© **Copyright** 1984 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.

Figure 1 Main menu of VM/PC



tions could be emulated using a generic MC68000 microprocessor and that the floating point instructions should be implemented in a dedicated chip to increase performance of scientific and engineering applications. To obtain this goal, IBM and Intel Corporation jointly defined another customized microprocessor based on the Intel 8087 to execute System/370 floating point instructions directly.

Using the multiple-microprocessor concept, engineers at Endicott designed and built a test card that proved the feasibility of a microprocessor-based System/370. Two problems remained: the first was to find a suitable base unit to contain the System/370 processor and the second was the development of an operating system that would make efficient use of it.

The VM/PC software traces its origins to an activity initiated at the IBM Cambridge Scientific Center to build a prototype of a CMS workstation. This prototype was called CnS (Cambridge nano-System) and is described in Reference 1. A group from IBM's Endicott laboratory, jointly with Cambridge, developed a version of CnS running on a "370" emulated entirely in software written for a Motorola MC68000 EXORmacs® system. Although CnS did not support virtual storage or full-screen displays and used diskettes for disk I/O operations, many of the feasibility questions were answered by this work.

By early February 1982, Cambridge had migrated CnS to an emulated System/370 environment housed as a coprocessor inside an IBM Personal Computer. Cambridge had built their prototype such that the System/370 environment could access its memory via a 16-bit bus but still share that memory with the PC. After evaluating the Cambridge prototype, the team from Endicott reached several conclusions:

- First, while software emulation of System/370 was feasible, it lacked sufficient performance for a viable product.
- Second, the required performance could be achieved by the custom System/370 processor chips under development in Endicott.
- Third, to fully capture System/370 applications, a virtual memory environment would be necessary. For this, a fixed disk was needed to support paging.

Thus, the Endicott team began developing a desktop version of System/370 CMS running on the System/370 coprocessor. The System/370 was packaged inside an unmodified PC XT, which contains the required fixed disk.

The PC XT/370 and the companion VM/PC software are discussed in detail in the remainder of this paper.

VM/PC compatibility and versatility

Compatibility with the VM/CMS application interface is the cornerstone upon which the "desktop System/370" was built. This interface is augmented by several other major features that combine to give the user convenient access to a broad variety of functions such as the ability to move files between PC DOS and the local CMS environments, the ability to print files locally or on a Virtual Machine/System Product (VM/SP) host printer, etc.[2]

When the VM/PC software is initiated, the user is presented with a menu offering access to four sessions. Each session (Figure 1) is independent and unique, and the four sessions are capable of being run concurrently:

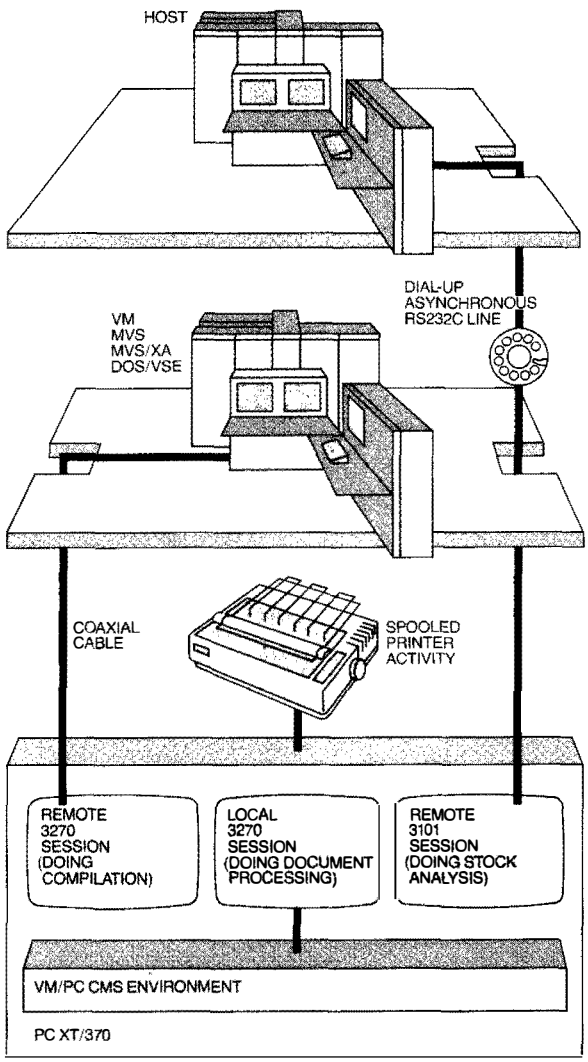
1. The *Local 3270 Session* allows full-screen access to the CMS environment running on the System/370 processor inside the PC XT/370.
2. The *Remote 3270 Session* provides use of the PC XT/370 as an IBM 3270 display device.
3. The *Remote 3101 Session* allows the operation of the IBM 3101 Emulation Program, which permits the use of the PC XT/370 as an IBM 3101 display. The Remote 3101 Session can be used to com-

municate with another computer (host or peer) via an RS232C connection.

4. The *System/370 Processor Control Session* allows manipulation of the memory, registers, and other facilities of the System/370 processor inside the PC XT/370.

The user can change from one session to another, without losing data, with a single keystroke. For example, a user could be checking the stock market quotations offered by a subscription news service over a telecommunications hookup (the Remote 3101 Session) while compiling a FORTRAN program on a host System/370 (the Remote 3270 Session) and, at the same time, be running a CMS document-processing application (the Local 3270 Session). These overlapped functions could all be executing while previously spooled files are being printed on the local printer (Figure 2).

Figure 2 VM/PC concurrent session activity



The VM/PC remote server

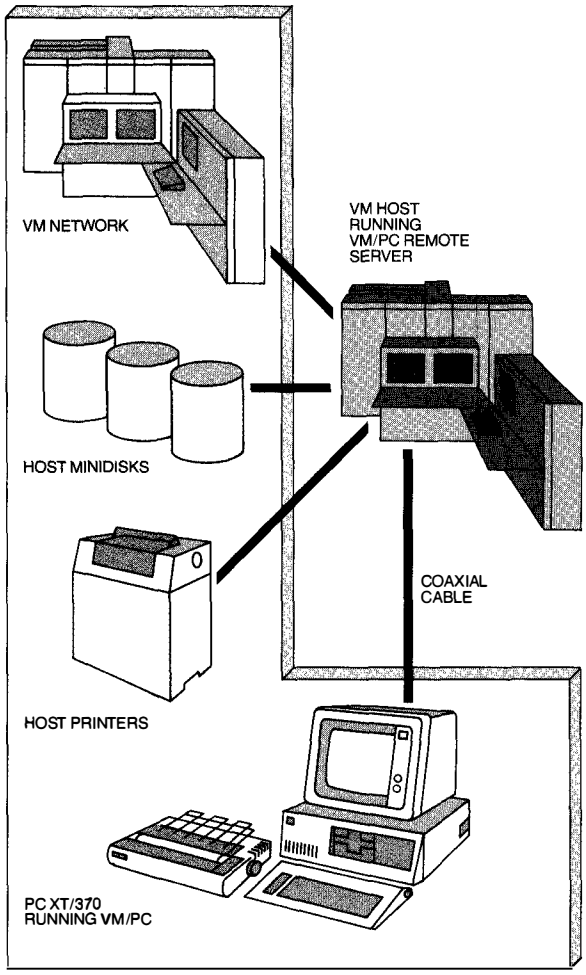
When running the Local 3270 Session, the user is provided with a CMS environment that is a compatible subset of its VM/SP counterpart on the host System/370 computer, and this local CMS interface can be expanded in a simple and effective way by coupling it to a CMS environment located at a host computer. This "coupling" is achieved by first activating a program called the VM/PC Remote Server at the user's host CMS machine (Figure 3), and then, while in the Local 3270 Session, issuing commands to link and access CMS minidisks on the host machine as if they were on the user's desk.

With the VM/PC Remote Server active, moving a file from the host CMS environment to the desktop CMS environment (or vice versa) is done with the COPYFILE command familiar to CMS users. It is also possible to tell the CMS editor (XEDIT), running in the Local 3270 Session, to edit a file that resides only on the host. This transparency of file access allows programs running in the desktop machine to read and/or write data on host files without creating a local copy of the file. A locally running CMS application can access a mix of up to 26 host and local minidisks with the familiar filemode mechanism used to determine the search order. Applications can be installed or designed in order to exploit the advantages of having different types of information on different media. For example, privacy considerations may suggest storing certain data on a minidisk on a removable diskette. The VM/PC Remote Server also gives locally running CMS programs the option of directing their printer output to the host or to the VM/SP Remote Spooling Communications Subsystem (RSCS) network at the host.

IMPORT and EXPORT commands

With all the facilities VM/PC provides the CMS user, it is possible to overlook the fact that VM/PC itself is a PC DOS application running under the standard version of that operating system. The PC XT/370 is also fully compatible with the PC XT and, when not running VM/PC, the user may be running PC DOS applications. Because both PC DOS and CMS applications

Figure 3 Coupling of host CMS and desktop CMS environments



may be run on the same system, two new CMS commands were added to VM/PC to move data between the VM/PC CMS environment and the PC DOS environment:

- IMPORT, which copies PC DOS files to files on CMS minidisks
- EXPORT, which copies CMS files to PC DOS files

Both commands perform data conversion (ASCII/EBCDIC) if specified.

PC XT/370 hardware overview

The PC XT/370[3] is comprised of a PC XT and three additional cards: the PC/3277EM, PC/370-P, and PC/370-M. The PC/3277EM card emulates an IBM 3277-2 terminal and provides a high-speed link by which programs and data are transferred between the PC XT/370 and a System/370 host. Local System/370 processing functions and address multiplexing between the PC XT and the System/370 processor are performed on the PC/370-P card. The PC/370-M card implements 512K of parity-checked, read/write, random access memory (RAM) as a two-ported memory. The PC/370-P and PC/370-M cards are connected via a ribbon cable and function as a unit. All cards use metal oxide semiconductor large-scale integration (MOSLSI) and standard 74S and 74LS series logic on multilayer printed circuit boards.

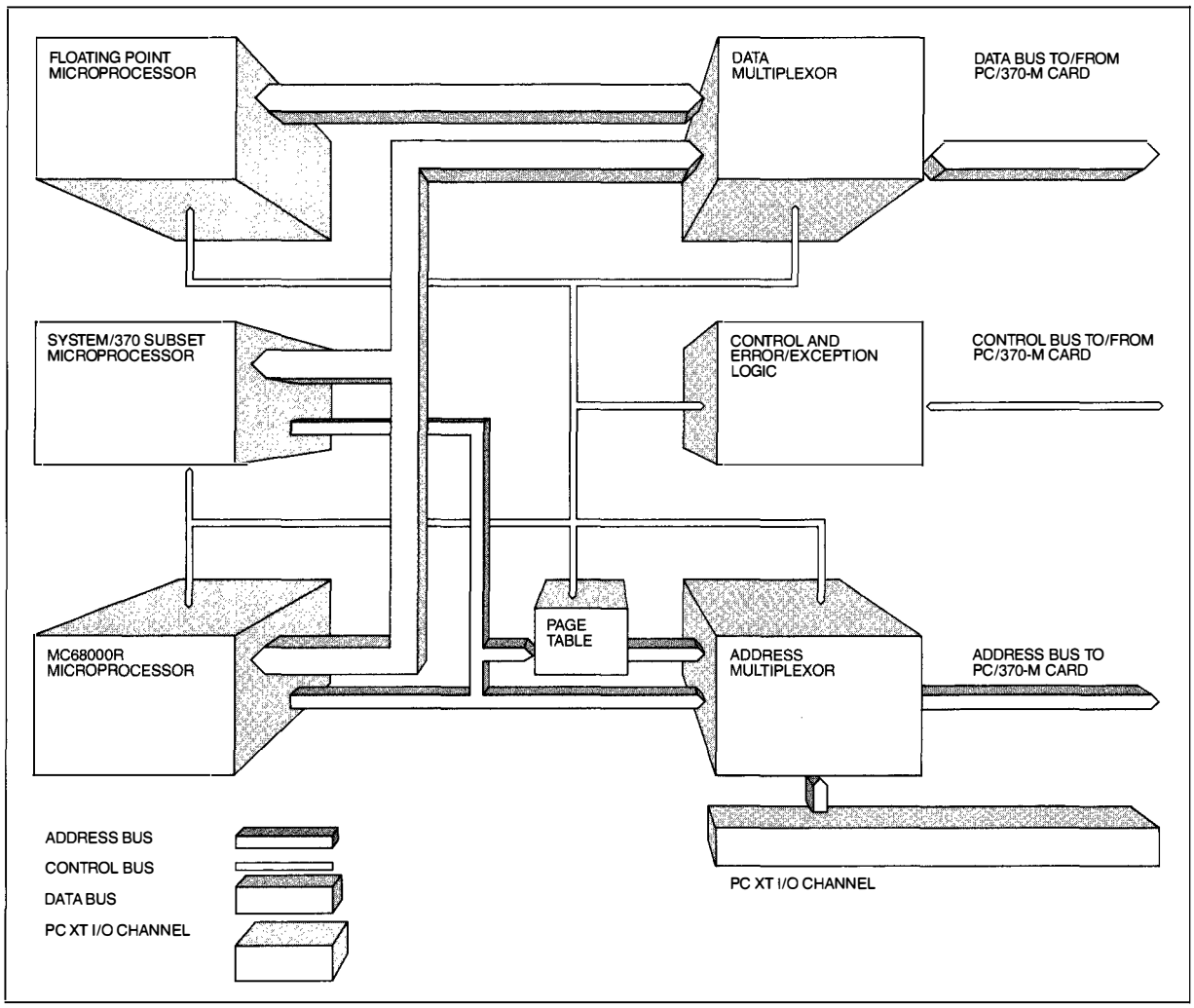
The PC/370-P card can be partitioned as shown in Figure 4. As previously mentioned, three microprocessors are used to implement the System/370 processing functions. A custom-developed System/370 Subset microprocessor performs most of the System/370 commercial instructions. Floating point, including extended precision, instructions are executed by a custom-developed Floating Point microprocessor, which works in close conjunction with the System/370 Subset microprocessor. The remaining instructions are emulated by an MC68000R microprocessor which also performs other tasks such as exception handling.

Dynamic address translation (DAT) is accomplished through the use of a page table consisting of static RAM chips. This realization provides for 4M of virtual memory arranged in 4K pages. Reference, change, page fault, and parity indicators are provided to assist the paging process and ensure proper operation. This method takes the place of the traditional main storage page table and table look-aside buffer (TLB). Its main advantage is the reduced complexity that the static RAMs offer compared to the TLB without the loss of speed associated with a page table located in main storage.

Address multiplexing is performed on the PC/370-P card to reduce the number of signal lines on the ribbon cable between the PC/370-P and PC/370-M cards. Four address spaces are multiplexed to the PC/370-M card:

1. Control storage
2. The 480K real address space of the PC XT/370 with DAT on
3. The 480K real address space of the PC XT/370 with DAT off
4. A 384K subspace of the PC XT

Figure 4 PC/370-P card



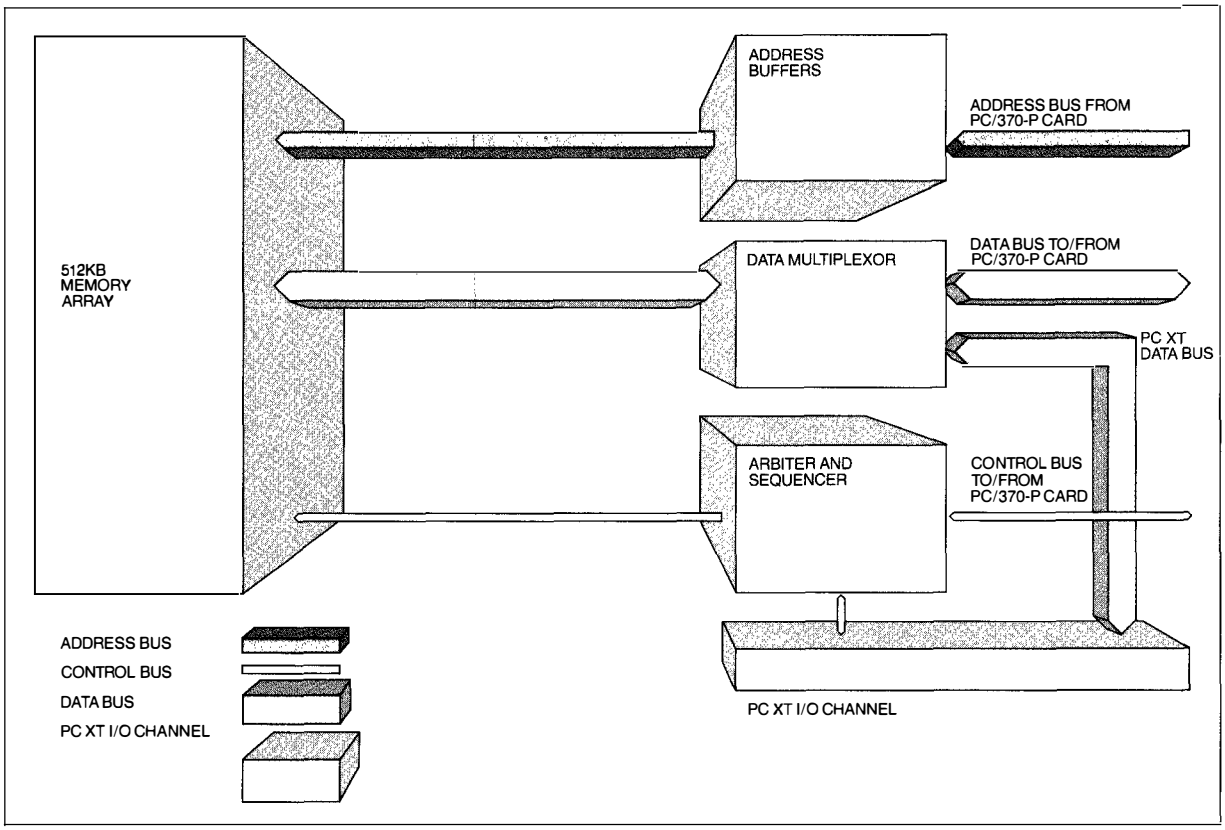
Error and exception detection logic is provided for

1. Off-boundary accesses
2. Odd program counter
3. Virtual storage out-of-bounds, i.e., beyond 4M
4. Real storage out-of-bounds, i.e., beyond 480K
5. Page faults
6. Page table parity errors

The remaining logic consists of control logic which coordinates system function and provides for hardware assistance of the "execute" instruction, modification of the page table, and selection of DAT on or off.

The PC/370-M card (Figure 5) provides a two-ported storage system consisting of 512K of memory, contiguous with respect to the System/370 address space. 32K are reserved for control storage which contains the system microcode and interprocessor communications areas. Only 384K of read/write RAM address space is available above the 256K on the system planar of the PC XT. To make the entire 512K of System/370 memory accessible to the PC XT, the center two 128K portions are interchanged via a bank select mechanism. Byte and halfword accesses by the System/370 processor are allowed, whereas all accesses by the PC XT are on a byte basis only. System/370 and PC XT accesses are arbitrated, with

Figure 5 PC/370-M card



priority going to the PC XT when simultaneous accesses occur. The use of a two-ported memory allows for ease of interprocessor communication and concurrent processor operation.

The PC XT/370 offers most of the facilities provided by a System/370 mainframe with the following exceptions. I/O instructions are not implemented since there are no I/O channels. However, there are equivalent functions implemented via the "DIAGNOSE" instruction. Since DAT is somewhat unique in its implementation on the PC XT/370, some modification of the "LOAD REAL ADDRESS" and "PURGE TLB" instructions is required. Two new operations associated with DAT were added: "MAKE ADDRESSABLE" and "MAKE UNADDRESSABLE." The timing functions available in System/370 are restricted since the PC XT clock is used as the time-of-day clock. These restrictions include decreased resolution, no interval timing, and no "TIME-OF-DAY CLOCK" control switch. Since the system is a single-user workstation, storage keys are not implemented and, if read, appear to be zero. Finally, a subset of the "PER" facility is implemented. Even with the aforementioned restrictions, a large number of applications will run unmodified. Hence, the PC XT/370 running under VM/PC is truly a desktop System/370.

VM/PC structure

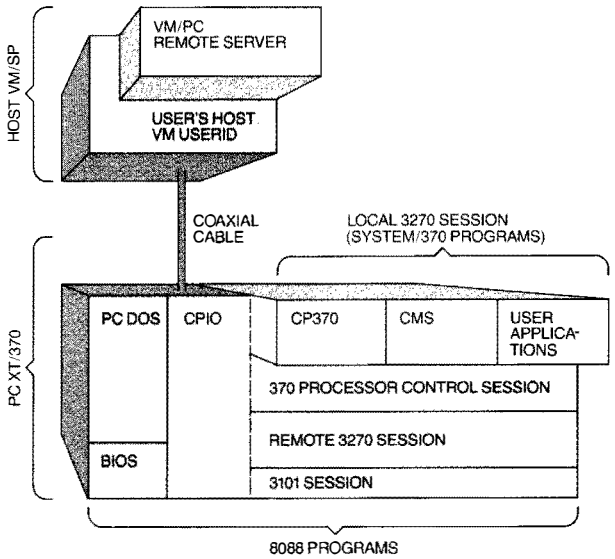
VM/PC consists of the following major components:

- *Installation/Local*—an interactive aid for copying the system files from the distribution diskettes to the user's fixed disk
- *Installation/Remote Server*—a bootstrap for installing the Remote Server on the host VM system from the distribution diskettes
- *Configurator*—an interactive data collection program that performs the "directory maintenance" function for VM/PC

- The *Session Group*—three VM/PC components that operate concurrently within the PC XT/370:
  - *Control Program I/O Services (CPIO)*—the part of the system that runs in the 8088 microprocessor and manages the I/O operations and the multiple sessions
  - *Control Program 370 (CP370)*—the control program (CP) functions that run in the System/370 and provide the virtual machine facilities for running environments such as the CMS component
  - *Conversational Monitor System (CMS)*—the virtual machine environment supplied with VM/PC
- *Remote Server (VMPCSERV)*—the host resident program that permits transparent interaction between the local CMS and host CMS environments

Figure 6 shows the structure of the session group and its relationship to PC DOS and to the VM/PC Remote Server. Note that all I/O operations are performed from the 8088 microprocessor environment, not the System/370 environment.

Figure 6 VM/PC structure



The VM/PC application interface permits most CMS applications to run unaltered.

The versatility of the VM/PC sessions was achieved by running the System/370 and 8088 processors independently and concurrently. By assigning all real device I/O operations, the 3270 and 3101 emulation functions, and the printing of spool files to the 8088 processor, the System/370 processor is able to run local CMS applications while the other activities proceed.

Like its VM/SP counterpart, VM/PC has both "CP" and "CMS" components. In VM/PC, the CP component operates partly in the 8088 environment and partly in the System/370 environment. The System/370 part of CP concentrates on managing the single virtual machine in which the CMS component runs its command and application interface.

The user, when interacting with the Local 3270 Session, has access to both CP and CMS commands compatible with those on the host VM environment. With these CP commands, the user can establish links, alter the spool characteristics of the printer, format minidisks, trace and debug programs running in the System/370 virtual machine, query the memory size, etc.

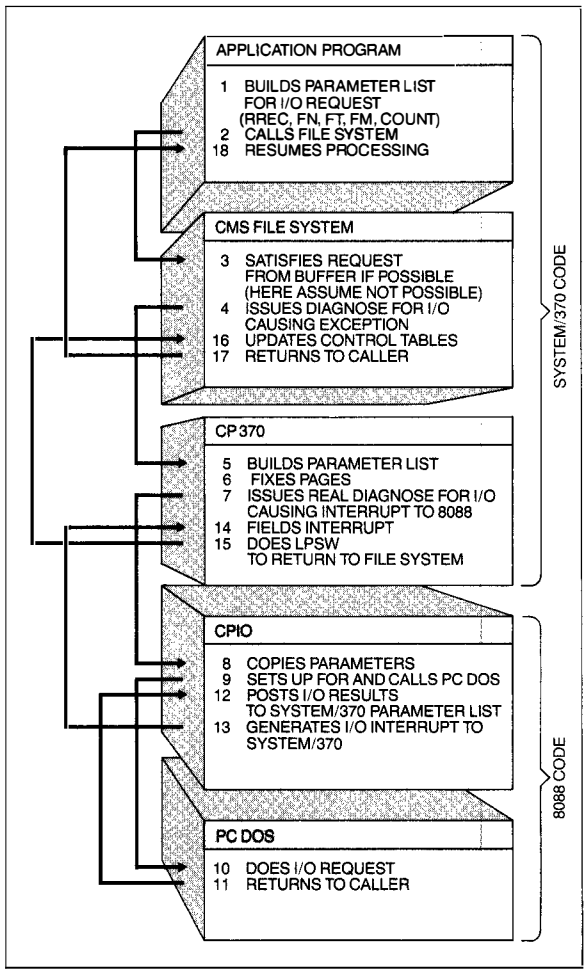
The CMS commands available to the user allow him to edit files, define and run "execs" in either EXEC or EXEC 2 languages, sort files, print files, load and run program text files, generate modules, run various application programs and language processor products, etc.

Application and system interfaces

Although the compatibility of the VM/PC application interface permits most CMS applications to run unaltered, the interface also allows the user to develop new or modified applications with confidence that such applications will run on both the host VM system and on VM/PC.

The CMS of VM/PC has a file system compatible with the VM/SP "extended disk formatting" file system.

Figure 7 Input/output flow within VM/PC



Also, the internal control block layouts and nucleus area contents of the CMS of VM/PC have been designed to permit existing applications that rely on these layouts and areas to function successfully.

VM/PC offers an application interface through its CMS, but it also makes available a system-level "DIAGNOSE" interface. This interface, similar to its counterpart in VM/SP, is available both to CMS applications and to those interested in developing software that will run in the System/370 virtual machine without the assistance of CMS.

The combination of application and system interfaces gives the VM/PC user a compatible environment for running existing applications and implementing new applications.

Files and I/O operations

All of the XT/370 disk files used by VM/PC are PC DOS files. In fact, all of the disk I/O operations are performed by VM/PC using PC DOS calls. By doing so, VM/PC was able to avoid the development expense of duplicating PC DOS file-support facilities as well as to take advantage of any future file-support enhancements that become available through PC DOS.

When an application program running under CMS in the Local 3270 Session requests a disk I/O service, a sequence of events is initiated that involves all three of the Session Group components as well as PC DOS. Figure 7 summarizes the steps involved in a typical request to read a disk record. The steps are as follows:

1. The application program builds a parameter list defining the read request. The list includes the name of the file, the file type, its disk location expressed as a filemode, the relative record number, the amount of data to be read, and the storage location into which to read the file.
2. The application program then issues a standard CMS read macro (FSREAD) to invoke the CMS file system.
3. The CMS file system analyzes the request and attempts to satisfy it using data already in one of its buffers. In this example, it is supposed that the buffers are found not to contain the needed data.
4. The CMS file system prepares for and issues an I/O DIAGNOSE instruction. Since CMS is operating in the System/370 problem state, the DIAGNOSE instruction results in a privileged operation exception, which causes program control to be transferred to CP370.
5. CP370, operating in the System/370 supervisor state, constructs a control block containing parameters defining the read request from the information made available by CMS.
6. CP370 then ensures that the buffers designated as I/O areas and control areas are in real memory and that paging activity will not overwrite these areas before the requested read is completed. This is called "page fixing."
7. CP370 next issues a DIAGNOSE instruction, which causes the System/370 hardware to present an interrupt to the 8088 processor.
8. The CPIO program, running on the 8088 processor, fields the interrupt and reads the control block constructed by CP370 to determine the nature of the request.

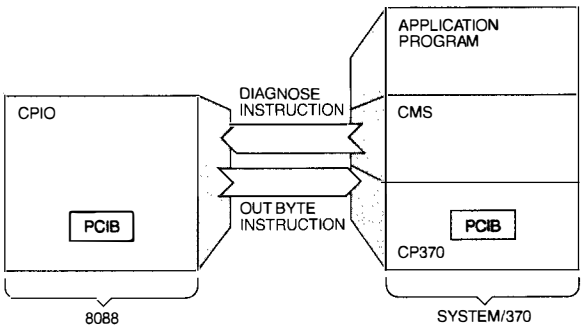
9. Determining that a file read is needed, CPIO prepares and issues a call to PC DOS.
10. PC DOS issues the file read using the file control block identified by CPIO. In this example, data read from the disk is read directly into the buffer that was page-fixed earlier (Step 6) by CP370.
11. Upon completion of the file read, PC DOS returns control to the application program.
12. The application program, in this case CPIO, updates the control block constructed by CP370 (Step 5) to indicate the completion of the requested service.
13. CPIO concludes by generating an I/O interrupt to the System/370 processor.
14. CP370 fields the interrupt and updates its internal control areas.
15. CP370 then concludes by initiating the standard Program Status Word exchange (LPSW instruction) used to transfer control to another program. In this case, the other program is the CMS file system, which had called CP370 (Step 4).
16. Upon regaining control, the CMS file system updates its file control tables.
17. The CMS file system then returns control to the application that had originally requested the read (Step 2).
18. The application resumes processing.

System/370–8088 interprocessor communication

In the previous description of the steps involved in an I/O activity, it was mentioned that the System/370 and the 8088 processors communicate by interrupting each other. This mechanism is used whenever the CP370 and CPIO need to communicate. Either program can interrupt the other. The CP370 program uses a DIAGNOSE instruction to generate a level 7 interrupt to the 8088 processor. The CPIO program executes an OUT BYTE instruction to set one of the PC XT/370 control registers so that a System/370 interrupt is generated.

Regardless of which program interrupts the other, the reason for the interrupt is conveyed in a control block constructed by the interrupting program. The control block, called a "Personal Computer Interface Block (PCIB)," contains the pointers and status information needed to respond to the interrupt (Figure 8).

Figure 8 The Personal Computer Interface Block (PCIB) is used during the System/370-8088 communication



Concluding remarks

Although VM/PC does not incorporate all the functions of the larger VM/SP, it does introduce a new and exciting capability to the System/370 user community. The "host feel" is real. The compatibility is real. The strategy of both the hardware and software development to preserve native PC XT capability is expected to accrue benefits for the PC XT/370 users as the base XT product evolves.

EXORmacs is a registered trademark of Motorola.

Acknowledgments

Development efforts for both the PC XT/370 and the VM/PC software were centered at IBM's Endicott Laboratory. The hardware was developed by the Processor Development organization, and the software and floating point microprocessor by the Engineering/Scientific Systems and Advanced Technology organization.

In addition to the contributions of IBM's Cambridge Scientific Center, the IBM Thomas J. Watson Research Center contributed the nucleus for the design of the VM/PC Remote Server.[4] The Research Center also provided a means of subsetting the CMS file system and provided valuable assistance in the development of significant parts of the CPIO component.

The IBM United Kingdom Laboratories Limited provided a CMS relocating loader technique which allowed VM/PC to support CMS SUBSET mode operation. They also contributed a means of establishing a reliable communications path between the host and the PC XT/370.

Cited references

1. T. D. Rosato, "MICRO CMS/370 (Cambridge nano-System)," *SHARE Proceedings,* SHARE 58, Los Angeles, CA (March 16, 1982).
2. *IBM Virtual Machine/Personal Computer User's Guide,* 6936733, IBM Corporation (1983); available through IBM branch offices.
3. *IBM PC XT/370 Technical Reference,* 6936732, IBM Corporation (1983); available through IBM branch offices.
4. B. C. Goldstein, R. A. Heller, F. H. Moss, and I. Wladawsky-Berger, "Directions in cooperative processing between workstations and hosts," *IBM Systems Journal* **23,** No. 3, 236–244 (1984, this issue).

Reprint Order No. G321-5222.

Frank T. Kozuh *IBM Systems Technology Division, P.O. Box 6, Endicott, New York 13760.* Mr. Kozuh joined IBM in 1966 and since that time has held numerous technical and managerial positions dealing with software and microcode development. In 1978 he became involved with various microprocessor-based activities in the laboratory, including the prototyping activities which preceded the development of VM/PC. In early 1982, he joined the Engineering/Scientific Systems and Advanced Technology organization to manage the development of the VM/PC software. Mr. Kozuh received a B.S. in mathematics from the University of Dayton in 1964 and an M.S. in applied mathematics from Purdue University in 1966.

David L. Livingston *IBM Systems Technology Division, P.O. Box 6, Endicott, New York 13760.* Mr. Livingston joined IBM in September 1981 and has since worked as a hardware engineer for the PC XT/370. He received a B.S.E. in 1976 and an M.E. in 1978 in electrical engineering from Old Dominion University.

Thomas C. Spillman *IBM Systems Technology Division, P.O. Box 6, Endicott, New York 13760.* Mr. Spillman joined IBM in 1960 and has been involved in various operating system and compiler development projects since that time. Currently with the Engineering/Scientific Systems and Advanced Technology organization, he has been involved in the specification and development of VM/PC from the early prototypes through delivery of the product, serving as the Project Manager of the effort. Mr. Spillman received a B.A. in mathematics from North Texas State University in 1960.

A tight coupling of workstations

by D. M. Chess

This paper addresses the problem of situations in which people at physically distant locations must have access to essentially the same computing environment at the same time. That is, each user must be able to provide input to whatever application or system is active, and must be provided with all relevant output. Common examples of this situation are demonstrations, presentations, education, and troubleshooting.

A prototype system has been developed to study ways of solving this problem in the microcomputer workstation environment. The prototype allows users at two IBM Personal Computers to share access to the computing environment through the keyboard and the display screen by tightly coupling the computers.

As computing power is distributed to small systems, away from large central sites, the physical distance between workstations presents some new problems and makes some old ones more severe. In particular, there are many situations in which two or more relatively distant people, at relatively independent workstations, need to be able to interact with the same computing environment[1] at the same time. Two examples may serve to give an indication of the problem.

In the first example, an independent software author in Portland, Oregon, wishes to demonstrate a new program to a business in New York that packages and distributes software. The author cannot afford the time and money to go there in person, but does not want to send the program by mail.

In the second example, an executive needs to learn to use the latest financial forecasting package from a teacher in his company's education department. The executive and the teacher both have the program, but the teacher is at a different site, and travel will be costly. Since the package runs on a personal computer that is only very loosely connected to the mainframe computer network of the company, no way exists to provide remote teaching through the central computing system.

These two examples represent a general type of situation that the spread of computing, and of distributed small-scale computing in particular, is making more common. These situations are characterized by the following conditions:

- There is a need for more than one person to interact with the same computing environment at the same time.
- The more nearly identical the computing environments, as perceived by each person involved in the interaction, the better. If, for instance, some I/O device is available to one person but not to another, it may still be possible to accomplish the objective, but it will be more difficult.
- Either the people involved are not at locations that are physically close, or the I/O devices involved do not lend themselves to use by more than one person.

The obvious approach to solving the problem in these situations is to have the people involved be in the same place, interacting with the same physical devices. In the case of the demonstration, the demonstrator and the interested audience can take turns at the keyboard, and all of them can see the display screen. With a larger audience, a closed-circuit tele-

© **Copyright** 1984 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.