Exchange

August 1985

Software

1 IBM PC Network SNA 3270 Emulation Program

10 Screen-Image Files

11 Writing TopView-Compatible Applications

17 The DOS CONFIG. SYS File

20 The XENIX Operating System

22 Obtaining a Filename Parameter in BASIC

23 PC Network Application Guidelines

Random Data

33 Customizing the Cursor

32 Electronic Bulletin Board System

35 New Products

36 Editor's Comments.

Exchange of IBM PC Information

IBM

Exchange of IBM PC Information is a monthly publication of the National Distribution Division, International Business Machines Corporation, Boca Raton, Florida, USA.

Editor	Michael Engelberg
User Group Editor	Bernard Penney
Associate Editor, Design Director	Karen Porterfield
Writer	John Warnock
Editorial Assistant	Steve Mahlum
Illustrators	John Alfred Dorn III Steven Guarnaccia Narda Lebo John Segal
Production	Cohen and Company
User Group Support Manager	Gene Barlow

Exchange of IBM PC Information is distributed at no charge to registered PC user groups. To register with us, please write to:

IBM PC User Group Support
IBM Corporation (2900)
P.O. Box 3022
Boca Raton, FL 33431-0922

To correspond with *Exchange,* please write to:

Editor, *Exchange*
IBM Corporation (2900)
P.O. Box 3022
Boca Raton, FL 33431-0922

POSTMASTER: send address changes to *Exchange of IBM PC Information,* IBM Corporation (2900), P.O. Box 3022, Boca Raton FL 33431-0922.

IBM cannot be responsible for the security of material considered by other firms to be of a confidential or proprietary nature. Such information should not be made available to IBM.

IBM has tested the programs contained in this publication. However, IBM does not guarantee that the programs contain no errors.

IBM hereby disclaims all warranties as to materials and workmanship, either expressed or implied including without limitation, any implied warranty of merchantability or fitness for a particular purpose. In no event will IBM be liable to you for any damages, including any lost profits, lost savings or other incidental or consequential damage arising out of the use or inability to use any information provided through this service even if IBM has been advised of the possibility of such damages, or for any claim by any other party.

Some states do not allow the limitation or exclusion of liability for incidental or consequential damages so the above limitation or exclusion may not apply to you.

It is possible that the material in this publication may contain reference to, or information about, IBM products, programming or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming or services in your country.

IBM PC Network SNA 3270 Emulation Program

Ken Ruka
IBM Corporation

Editor's note: Acronyms used in this article are explained at the end of the article.

The IBM PC Network SNA 3270 Emulation Program is a communications program that enables an IBM Personal Computer to emulate some of the functions of the IBM 3270 series of display terminals and IBM 3287 printers that serve as workstations connected to a host computer. The program also enables an IBM Personal Computer to emulate some of the functions of an IBM 3274 Control Unit, thereby providing a communications gateway between the host computer and other personal computers that are connected to the gateway via the IBM PC Network.

IBM PC Network SNA 3270 Emulation can be configured four ways:

- **Communications gateway.** In this configuration, an IBM Personal Computer emulates some of the functions of an IBM 3274 model 51C control unit.

  The 3274 provides the interface between the host computer and one or more 327X display terminals used as workstations. The IBM PC Network SNA 3270 Emulation Program enables an IBM Personal Computer to perform the role of the 3274. When used in this manner, an IBM Personal Computer is called a gateway.

  The gateway personal computer provides the interface between the host computer and one or more personal computers used as workstations. A workstation computer is called a network station because it connects to the gateway computer via the IBM PC Network. Therefore, the IBM PC Network SNA 3270 Emulation Program gives the IBM PC Network a communications gateway to a host computer.

- **Network station.** This configuration enables an IBM Personal Computer to emulate some of the functions of an IBM 3278 model 2 display terminal, an IBM 3279 model S2A color display terminal or an IBM 3287 model 1 printer. An IBM Personal Computer used in this manner is called a network station, because it uses the IBM PC Network to connect to the gateway station.
- **Combined communications gateway and network station.** This configuration enables an IBM Personal Computer to serve as both the communications gateway and an individual network station.
- **Standalone station.** A standalone station connects directly to the host computer; it is not part of an IBM PC Network and does not connect to a communications gateway station. This configuration enables a standalone IBM Personal Computer to emulate some of the function of an IBM 3274 model 51C control unit as well as some of the functions of the workstations listed above—IBM 3278 model 2, IBM 3279 model S2A, or IBM 3287 model 1.

Figure 1 gives an overview of these four configurations.

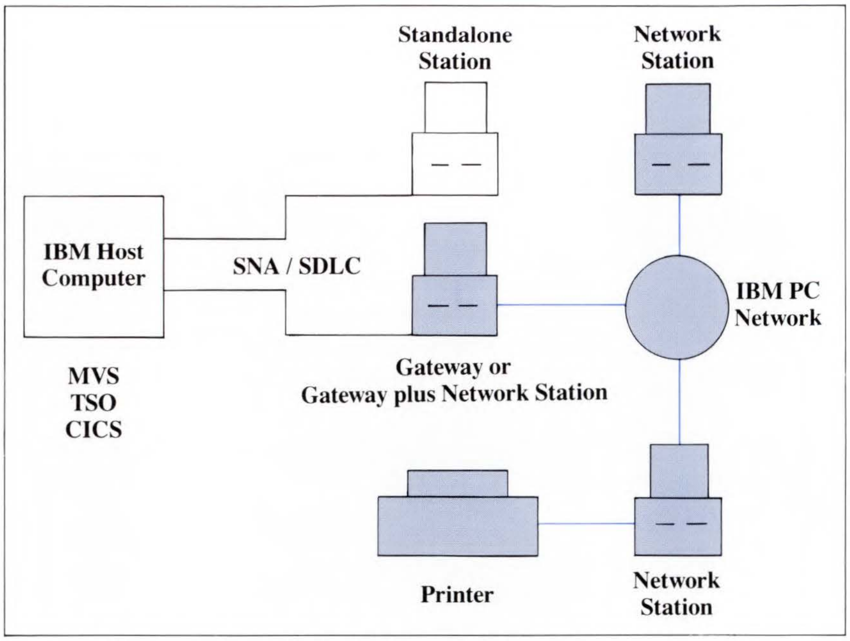


Figure 1. IBM PC Network SNA 3270 Emulation Program Configurations

At setup time, you give the IBM PC Network SNA 3270 Emulation Program certain information that corresponds to the type of station you require. For a gateway station, pertinent information includes the gateway name, the number of host sessions that the gateway will handle, the network station names, and telephone line and modem information. For a network station, the only pertinent information is the station's name. This configuration information is stored on the program diskette.

Starting an SNA Session

When you start an SNA session on a personal computer connected to the PC Network, the IBM PC Network SNA 3270 Emulation Program gives the gateway station name and the network station name to each respective PC Network Adapter.

You can load either the gateway station or the network station first. After you bring up the second station, the session begins.

The IBM PC Network SNA 3270 Emulation Program communicates with an IBM 30XX or 43XX host computer using SNA/SDLC protocol, LU types 1, 2 or 3. When configured as a communications gateway, the program supports up to 32 SNA sessions between network stations and the host computer.

If a network station has both a display and a printer, it can conduct two SNA sessions when configured for both at the host computer.

The IBM PC Network SNA 3270 Emulation Program at the gateway station maintains a status table that keeps track of all the SNA sessions that its network stations are holding. For each network station, the table contains the following information:

- **Session status for SNA**—SSCP-PU, SSCP-LU, and LU-LU session established
- **Session location**—either at the gateway station or a network station

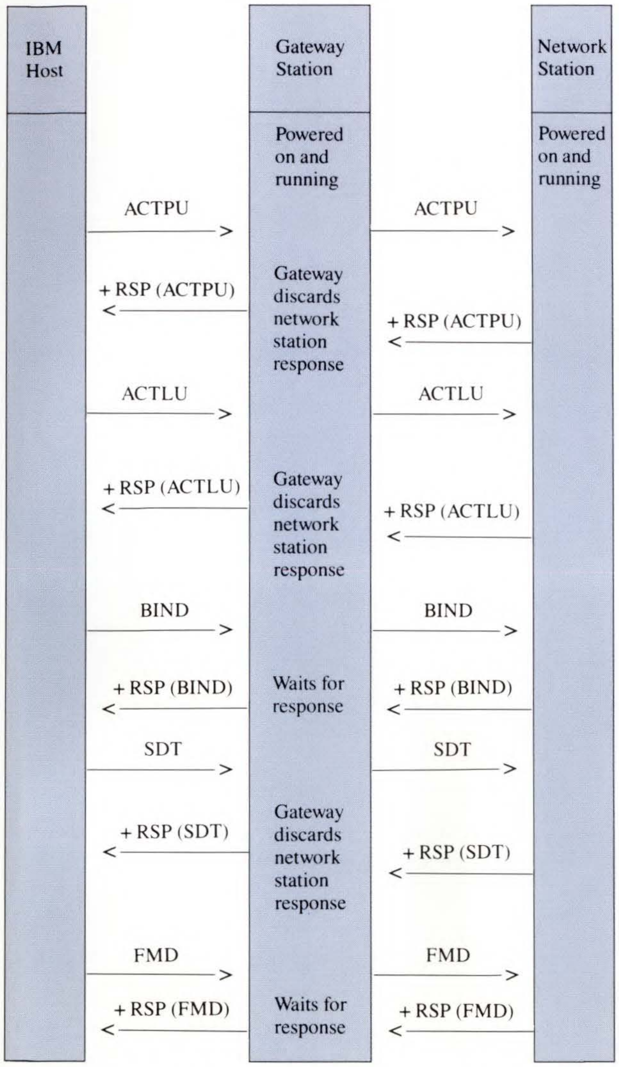


Figure 2. SNA Session Initiation for Powered-On Network Station

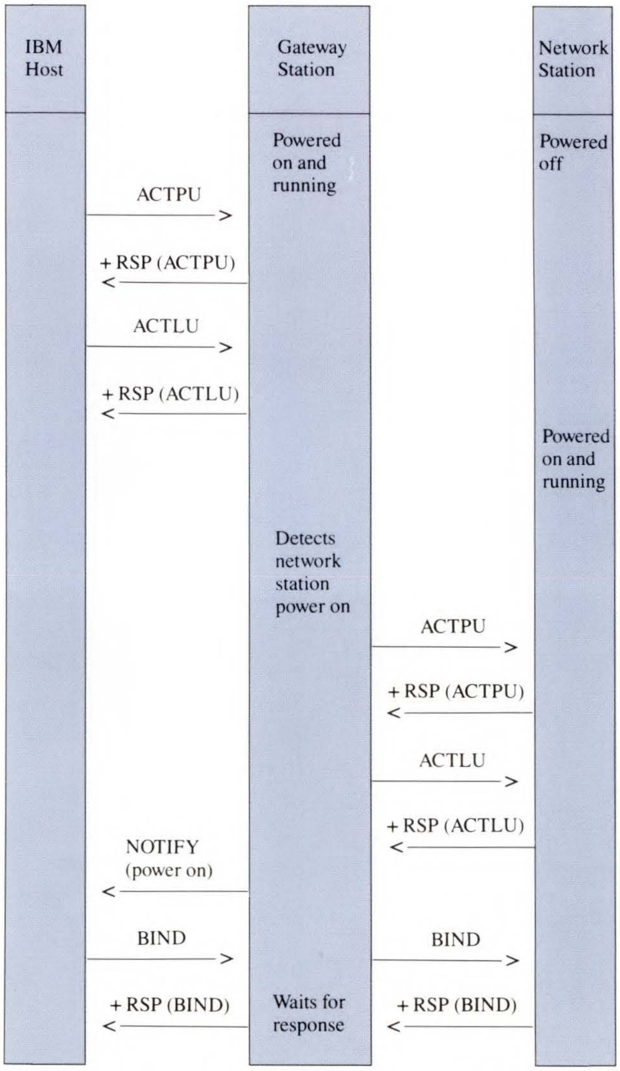


Figure 3. SNA Session Initiation for Powered-Off Network Station

- **Session address**—SNA Destination Address Field (DAF) of the network station
- **Session name**—name used on the IBM PC Network
- **Session status for IBM PC Network connection**—either active or inactive

The gateway station routes SNA commands to a network station as follows:

1. From the SNA command, the gateway obtains the Destination Address Field (DAF) of the network station.
2. The gateway uses the DAF to enter the gateway status table.
3. From the table, the gateway obtains the name for the network station.
4. Finally, the gateway sends the SNA command to that network name.

Figures 2 through 4 show the interactions between the host computer, the gateway station and a network station. In Figure 2, the network station is powered on at the beginning; in Figure 3, it is powered on later. Figure 4 shows session termination.

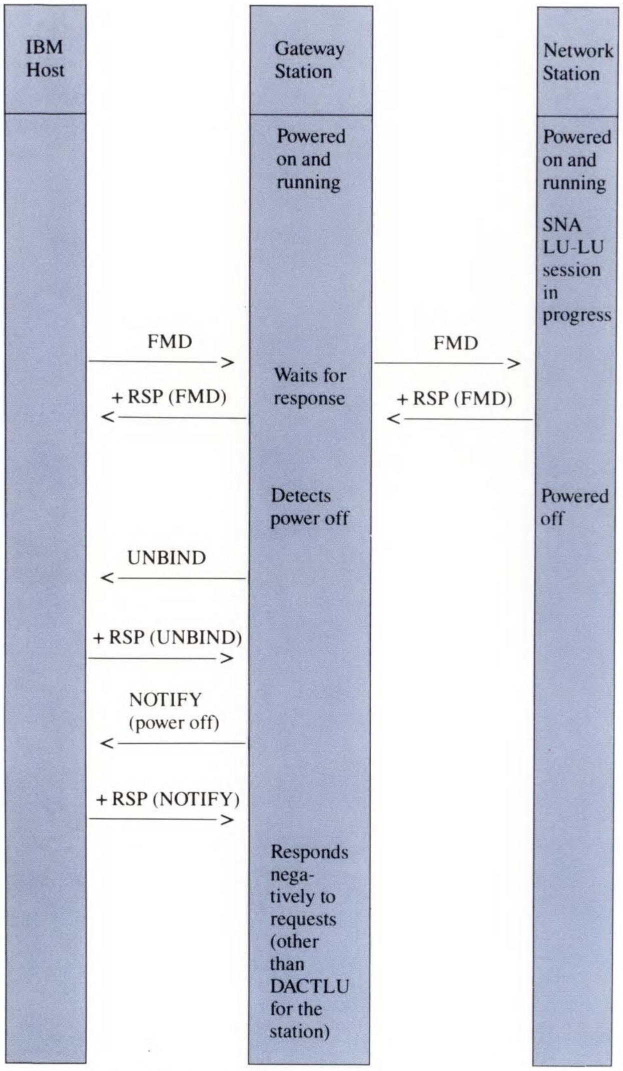


Figure 4. SNA Session Termination

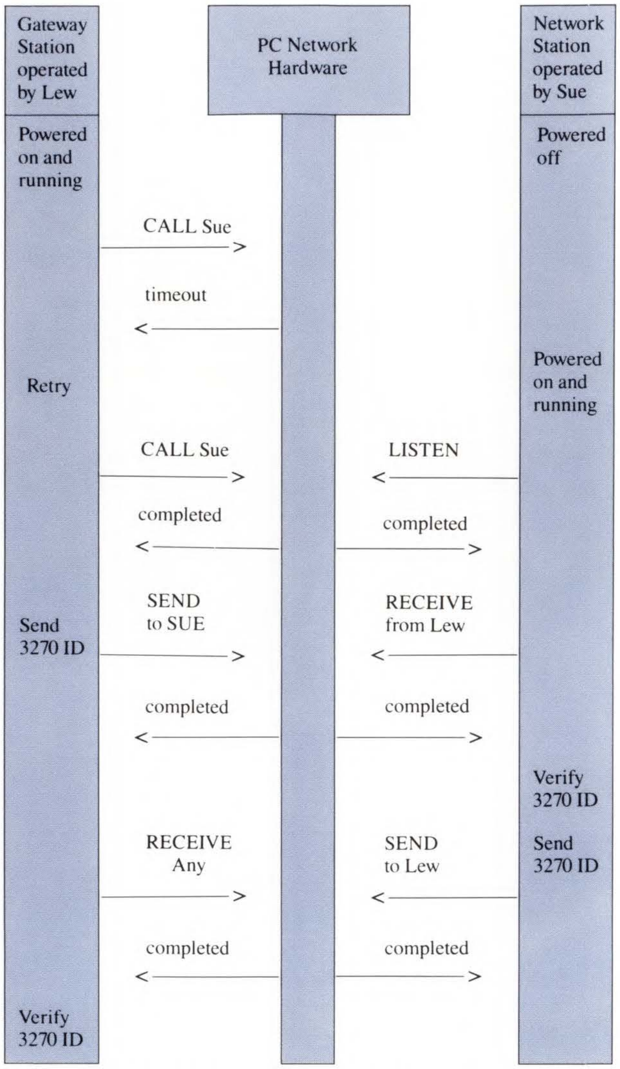


Figure 5. NETBIOS Commands Used During Network Session Initiation

Using NETBIOS

The IBM PC Network SNA 3270 Emulation Program uses NETBIOS as an interface to the IBM PC Network hardware. NETBIOS functions include bringing up a session between the gateway and a network station, maintaining the session, terminating it, and recovering the session.

The IBM PC Network SNA 3270 Emulation Program uses the following NETBIOS commands:

- General commands:
  —Reset (used only if the adapter status indicates that no other application is using the NETBIOS)
  —Cancel
  —Adapter Status
- Name support commands:
  —Add Name
  —Delete Name

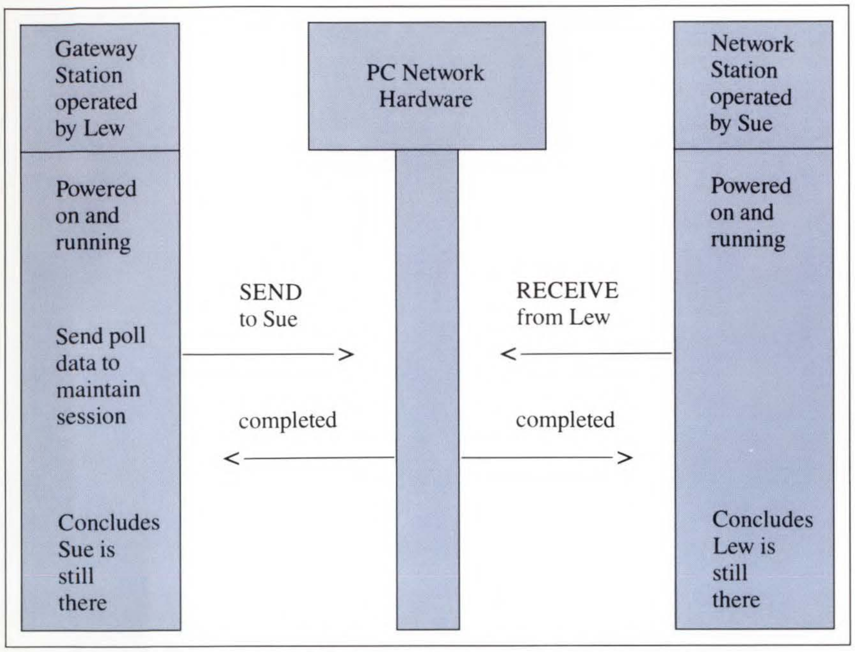

Gateway Station operated by Lew
PC Network Hardware
Network Station operated by Sue
Powered on and running
Powered on and running
Send poll data to maintain session
SEND to Sue
RECEIVE from Lew
completed
completed
Concludes Sue is still there
Concludes Lew is still there

Figure 6. NETBIOS Commands Used During Network Session Polling

- Session support commands:
  —Call (only in gateway configuration)
  —Listen (only in network station configuration)
  —Hang Up
  —Send
  —Receive (only in network station configuration)
  —Receive Any (only in gateway configuration)

Figures 5, 6 and 7 illustrate how the IBM PC Network SNA 3270 Emulation Program uses the NETBIOS commands during session initiation, polling and termination.

Emulation Program Features
The IBM PC Network SNA 3270 Emulation Program has several features intended to make you more productive.

- **Host File Transfer.** This allows a user to transfer files to and from the host.
- **Host-Initiated or Operator-Initiated Print.** This prints host data on a PC printer, whether the command was initiated from the host or a network station.
- **Deferred Print to Disk.** A user can transfer data to a network station for printing, but instead of printing it immediately, can defer it to a PC disk or diskette and print it later.
- **Screen Save to Disk.** With this feature, a network station user can save host screens to a predefined PC disk or diskette file.
- **Keyboard Remapping.** This feature lets the user redefine the positions of keyboard keys.
- **Suspend/Resume.** With this feature, an alternate foreground task can be used. Switching between PC Network 3270 and the alternate foreground task is accessed by pressing Alt-Esc.

The IBM PC Network SNA 3270 Emulation program always runs in Foreground #1. The user can switch back to DOS by pressing Alt-Esc. This places PC Network 3270 in the background, but still running. Once back in DOS, the user can initiate a well-behaved DOS application in Foreground #2. By pressing Alt-Esc again, the application in Foreground #2 will be suspended and PC Network 3270 will return to Foreground #1.

While PC Network 3270 is running in the background, it cannot access the display or the keyboard. However, the communications link is kept active. Therefore, if a file transfer or host print were in progress, it would continue to run, even though PC Network 3270 is running in the background.

Additional limitations and requirements for an alternate program in foreground #2 are:
—DOS 2.10, 3.00 or 3.10
—Relocatable
—Cannot alter certain software/ hardware interrupts
—Use SETINT to alter the other interrupts
—Restore altered interrupts before termination
—Use published DOS and BIOS interrupt and function call interfaces

- **Automatic session recovery.** With this feature, if a network station goes down, then comes back up, the program restarts the session; no operator intervention is necessary.
- **Dynamic maintenance of lists.** This lets you dynamically add to or delete from the gateway station's list of network stations without having to bring down the network.

Personal Computer Operating Considerations

The IBM PC Network SNA 3270 Emulation Program uses PC interrupt levels 3 and 4 for the SDLC Adapter, and either level 2 or 3 for the IBM PC Network Adapter. If any other hardware uses these interrupts, conflicts may arise and the results may be unpredictable. Only one PC Network Adapter may be present, configured for interrupt level 2 (the default interrupt level for the PC Network Adapter shipped by IBM is level 2) when the PC Network 3270 program is configured to use the SDLC Adapter (Gateway configurations or Standalone).

You should load resident code such as PRINT.COM and GRAPHICS.COM before you load the IBM PC Network SNA 3270 Emulation Program. Do not load resident code in the alternate partition (Foreground #2). When you terminate the IBM PC Network SNA 3270 Emulation Program, you will free the memory allocated to the program itself, and you will free the memory allocated to other programs that you loaded into the alternate partition after you loaded the emulation program.

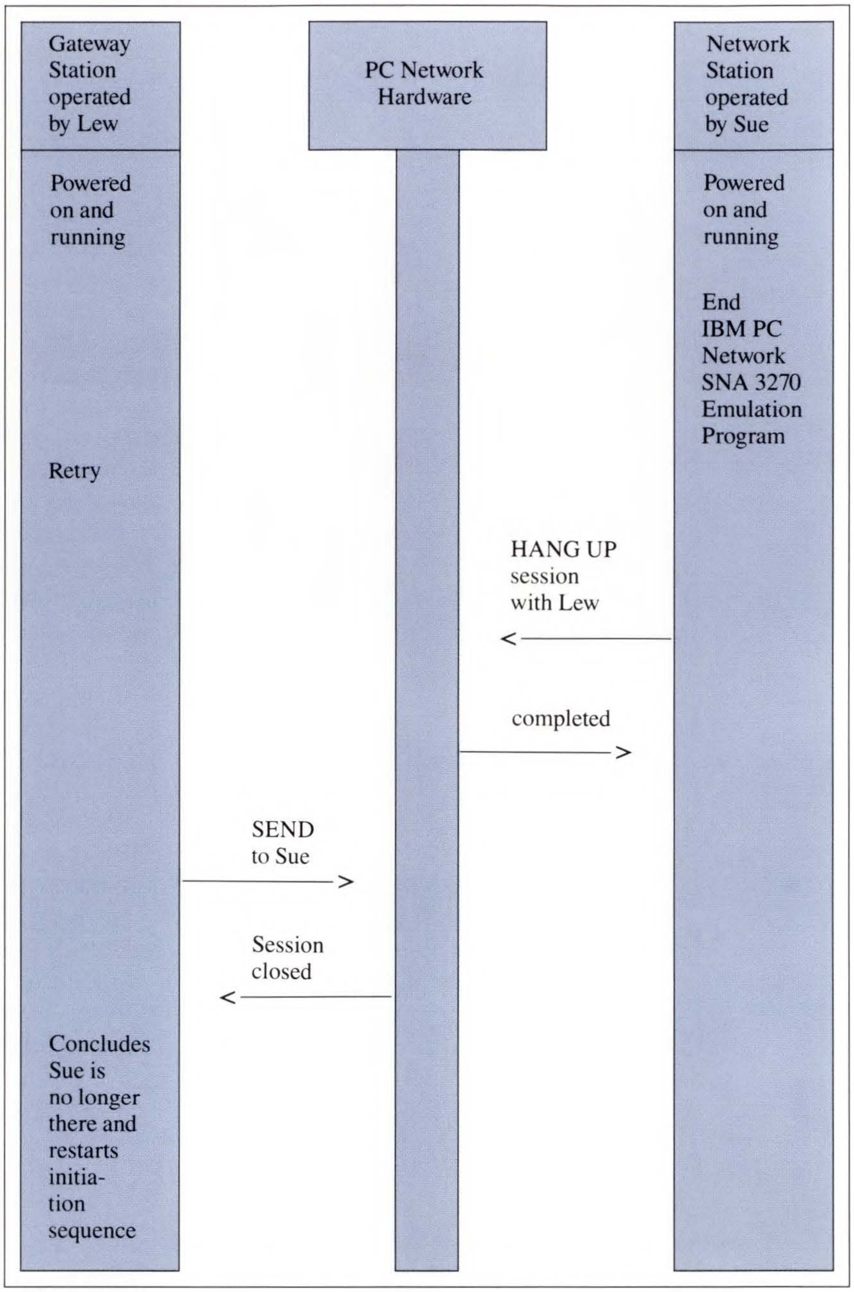

Gateway Station operated by Lew
PC Network Hardware
Network Station operated by Sue
Powered on and running
Powered on and running
End IBM PC Network SNA 3270 Emulation Program
Retry
HANG UP session with Lew
<
completed
>
SEND to Sue
>
Session closed
<
Concludes Sue is no longer there and restarts initiation sequence

Figure 7. NETBIOS Commands Used During Network Session Termination

	IBM Personal Computer	IBM Personal Computer XT	IBM Personal Computer AT	IBM Portable Personal Computer
Standalone Remote Station	Y	Y	Y	Y
Network Station	Y	Y	Y	Y
Communications Gateway	Y	Y	Y	N
Communications Gateway and Network Station	Y	Y	Y	N

Figure 8. Program Availability

	Gateway	Gateway with Network Station
Gateway (up to 8 SNA sessions)	256KB	320KB
Gateway (up to 16 SNA sessions)	320KB	320KB
Gateway (up to 24 SNA sessions)	320KB	320KB
Gateway (up to 32 SNA sessions)	384KB	384KB

Figure 9. Program Memory Requirements

Editor's note: The following material is reprinted from a notice sent to Authorized IBM Personal Computer Dealers.

IBM PC Network 3270 Hardware and Software Requirements

The IBM Personal Computer hardware requirements to run PC Network 3270 are 256KB of memory and one double-sided diskette drive.

Figure 8 describes which members of the IBM Personal Computer family support the PC Network 3270 configurations.

Memory requirements for IBM Personal Computers running PC Network 3270 which includes file transfer function, printer support, alternate task support, DIRectory key function, and IBM Personal Computer DOS are shown in Figure 9.

• Standalone Remote Station	256KB
• Network Station	256KB
• Network Station with IBM PC Network Redirector	320KB

Certain IBM Personal Computer applications may require memory in addition to that required to run PC Network 3270. (Refer to your application documentation.)

In addition, the following IBM Personal Computer hardware and software is required for each of the basic functions of PC Network 3270:

Standalone Remote Station. One SDLC Adapter (part number 1501205 or 1502090) and DOS 2.10, 3.00 or 3.10

Network Station. One IBM PC Network Adapter (part number 6450213) and DOS 2.10, 3.00 or 3.10

Communication Gateway. One SDLC Adapter (part number 1501205 or 1502090), one IBM PC Network Adapter (part number 6450213), and DOS 2.10, 3.00 or 3.10

Communication Gateway and Network Station. One SDLC adapter (part number 1501205 or 1502090), one IBM PC Network Adapter (part number 6450213), and DOS 2.10, 3.00 or 3.10

With the PC Network 3270 Program, the SDLC Adapter with part number 1501205 will operate at line speeds up to 9.6Kbps. (Part number 1501205 is required for the IBM Personal Computer AT.)

With the PC Network 3270 Program, the SDLC Adapter with part number 1502090 will operate at line speeds up to 4.8Kbps. (Part number 1502090 has been replaced by part number 1501205.)

The PC Network Program and DOS 3.10 are required if the Network Station is running the Redirector function.

System Operating Considerations for the IBM Host Computer

From the viewpoint of an IBM 43XX or 30XX host computer, PC Network 3270 causes a standalone IBM Personal Computer or an IBM Personal Computer functioning as a gateway on the IBM PC Network to emulate a subset of the IBM 3274 model 51C control unit functions.

IBM host computers communicating with IBM PC Networks which use PC Network 3270 must have the following PTFs installed:

- For VTAM with TSO, PTF UZ90213

- For MVS, Version 2, Release 1, PTF 0Z69591
- For MVS, Version 1, Release 3, PTF 0Z76005

The programs required to be installed on the IBM host 43XX or 30XX for file transfer are:

- 3270 PC File Transfer Program 5665-311, Release 1, Mod 0, (for TSO)
- 3270 PC File Transfer Program 5664-281, Release 1, Mod 0, (for VM)
- CICS/VS 3270 PC File Transfer Program 5798-DQH, Release 1, Mod 0 (for CICS/VS)

Other System Operating Considerations

When an IBM Personal Computer is using IBM PC Network 3270 with the IBM PC Network Program, only the network station configuration may be used with the IBM PC Network Redirector function to access IBM PC Network disks and print servers.

PC Network 3270 cannot coreside with the IBM PC Network Program in the same IBM Personal Computer when the IBM PC Network Program is configured as a Receiver, Messenger or Server.

When you are using the IBM PC Network redirector function while IBM PC Network 3270 is loaded, you should not use the full screen interface or IBM PC Network commands. Unpredictable results may occur if you attempt to do so.

The IBM PC Network 3270 Program emulates a subset of IBM 3274 Model 51C functions and does not support the Network Management Vector Transport (NMVT) Communications Network Management (CNM) Architecture.

Program Name	Part Number	Notes
Assistant Series		
—Filing Assistant	6024145	
—Graphing Assistant	6024147	
—Reporting Assistant	6024146	
—Writing Assistant	6024144	
—Accounting Solutions	6024152	
—Executive Solutions	6024151	
—Home Solutions	6024150	
Business Management Series		
—Accounts Payable Edition	6410951	
—Accounts Receivable Edition	6410952	
—General Ledger Edition	6410950	
—Inventory Accounting Edition	6410955	
—Payroll Edition	6410953	
DisplayWrite 1	6024188	1
DisplayWrite 2 Version 2.2	6024198	1, 2, 3
DisplayWrite 3	6024177	1, 2, 3
DisplayWrite Legal Support	6024190	
DisplayWrite Medical Support	6024197	
Office Correspondence Retrieval System	6824160	
Personal Decision Series		
—Data Edition	6410936	

Figure 10. Program Compatibility Listing

Notes:

1. DisplayWrite 1, 2, and 3 use a DWx.BAT file which must be modified to remove KQE.COM.
2. If DisplayWrite programs exist in a sub-directory, the files DW2.BAT and DW3.BAT must be edited to include a PATH command.
3. DisplayWrite 2 and 3 require the DOS Background Print program PRINT.COM to be in memory before loading IBM PC Network 3270.

IBM printers supported by PC Network 3270 are:

- IBM 5152 Graphics Printer
- IBM 5182 Color Printer
- IBM 5216 Wheelprinter
- IBM 5201 Quietwriter

Product Packaging

The PC Network 3270 program package contains:

- IBM PC Network 3270 Program double-sided diskette
- Learning the IBM PC Network 3270 Program double-sided diskette
- IBM PC Network 3270 Reference Document 8544-2286-0
- Keyboard templates for the IBM Personal Computer, IBM Personal Computer XT, IBM Portable Personal Computer, and the IBM Personal Computer AT

Application Program Compatibility

Figure 10 lists the IBM Personal Computer DOS applications programs that were found to be compatible when running in the alternate partition of the IBM PC Network 3270 Program operating under IBM Personal Computer DOS 2.10 and 3.00. IBM has not tested PC Network 3270 with the listed programs running in the alternate partition under DOS Version 3.10.

When you are running the IBM PC Network 3270 Emulation Program, if you also run other programs that are not compatible with IBM PC Network 3270, you may get unpredictable results.

Glossary of Acronyms and Commands

ACTLU—Activate Logical Unit. Establishes a connection between ports in a SNA network.

ACTPU—Activate Physical Unit. Establishes a connection between physical devices in a SNA network.

BIND—Activate session between two logical units.

CICS—Customer Information Control System. A program product that allows you to build and maintain data bases, and share databases and application programs among many users.

FMD—Function Management Data. End-user information exchanged between LUs and SSCPs in a SNA network.

LU—Logical Unit. A port that allows you to access a SNA network and other users. Logical units support at least two sessions: with an SSCP and with another logical unit.

MVS—Multiple Virtual Storage. A time sharing system control program designed for production use on the System/370 computers.

NODE—A junction (usually connected to other nodes) where physical units are clustered.

PTF—Program Temporary Fix. A temporary solution to a program problem.

PU—Physical Unit. A device that monitors and controls communications with other devices (printers, displays, etc.).

SDLC—Synchronous Data Link Control. A protocol for transmitting information over any link connection.

SDT—Start Data Traffic. A message sent between VTAM nodes to allow communications to begin.

SESSION—The connection of any two addressable units in a SNA network for communications.

SETINT—Set Interrupt.

SNA—Systems Network Architecture. A set of formats and rules for transmitting information over large networks and a set of methods for controlling the resources in large networks.

SSCP—System Services Control Point. A device within a SNA network that controls all or part of the network and handles problem determination.

TSO—Time Sharing Option. An extension of the operating system that allows remote stations conversational use of the computer.

UNBIND—Terminate a session between two logical units.

VM—Virtual Machine. A time sharing. A time sharing system control program designed for development use on the System/370 computers. It allows each user to have a virtual computer of their own with unique operating systems and devices.

VTAM—Virtual Telecommunications Access Method. A set of programs that control communications between systems and applications running under MVS.

+RSP—Positive Response.

Screen-Image Files

Larry Lockwood
Orange County IBM PC Users Group

Last December I chaired a meeting at which over 30 representatives of Naval and Air Force installations from around the country attended a demonstration of a cruise missile data base using an IBM PC with a 23MB external hard disk. All morning, these people stood while the software laboriously ground through the data base, time after agonizing time. Since the objective of the demonstration was to present results (not the manipulations required to produce the results), we all wasted a morning that could have been more productively spent on other agenda items.

That night, as I thought of how the demonstration might have been handled if only I had had some additional equipment, it occurred to me that snapshots and software might work together. I'd create a program that would take "snapshots" of CRT displays and store them in a file. A second program would recall them. It would take a week or so and wouldn't cost a thing (except the programming time at home each evening). By March, I had created two assembly language programs, which I called PUSH and POP because I saw a similiarity between saving/restoring a CRT image on a file and pushing/popping a register to a stack.

PUSH.COM is the program that, once loaded, remains resident in memory. When it has been loaded, the CRT display can be copied to a screen-image file by concurrently depressing the Shift-PrtSc keys. You can no longer use Shift-PrtSc to copy the CRT display to the printer until you reboot (IPL) the computer.

The syntax for loading PUSH.COM is:

PUSH drive:filename (do not use a file extension)

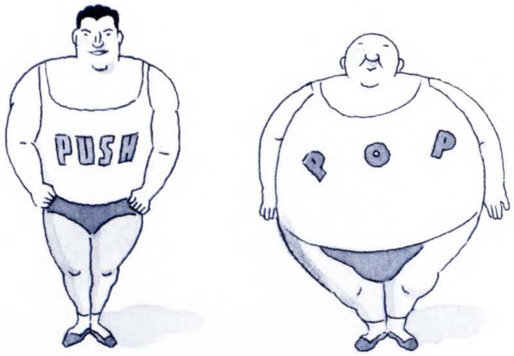
For example, suppose PUSH.COM is on the diskette in drive A and you wish to create a series of screen-image files named SNAPSHOT on a diskette in drive B. The installation entry would be:

```
PUSH b:snapshot
```

What finally evolved was a POP.COM program that restores the original screen image and allows you to modify almost everything.

When you first press the Shift-PrtSc keys, PUSH will create a screen-image file named SNAPSHOT.001 on the diskette in drive B. The second time the Shift-PrtSc keys are pressed, another screen-image file, SNAPSHOT.002, will be created, and so on.

When you save a text screen, the corresponding screen-image file size is 4100 bytes. A graphics screen will create a corresponding 16,300 byte screen-image file. You can copy approximately 88 text images or 22 graphics images to a 360KB diskette. Once you create a series of screen-image files, use the DOS RENAME command to give your files more meaningful names.



POP.COM restores a screen-image file to the display. The syntax, using the example above, would be

```
POP B:SNAPSHOT. 001
```

When I first tried POP.COM, I found that it always restored graphics images on a black background with the palette 0 foreground colors (cyan, white, and magenta). After digging through the Technical Reference Manual, I concluded that these are the default IBM PC color settings and there was no way that I could rewrite PUSH.COM to capture the Color Select Register setting (it's a write-only register). My only choice was to modify POP.COM to provide a means of restoring the original colors via function keys. While I was at it, it occurred to me that the function keys could also be used to change the attributes of individual characters in text files.

One thing led to another and what finally evolved was a POP.COM program that restores the original screen image and allows you to modify almost everything. The function keys do the following:

F1—Changes background colors of graphics images and border colors of text images.
F2—Toggles graphics foreground colors back and forth between palettes 0 and 1.
F3—Selects background colors for individual characters in text mode.
F4—Selects foreground colors for individual characters in text mode.
F5—Exercises your left forefinger.
F6—Controls individual character blinking.
Shift-F7—Reloads the original screen-image file again. (Use this if you've made modifications that you don't like and decide it would be quicker to just start all over again.)
Shift-F8—Saves modified image back to the disk with the original filename. (Careful—this destroys the original image.)
Esc—Exits and returns to DOS without saving modified screen.
Other keys—The cursor keys are functional in text and graphics modes, as are the alphanumeric keys. In graphics mode, however, the cursor is implemented as a one-pixel dot that moves around the screen. It starts in the upper left-hand corner of the screen, but you won't see it until one of the cursor keys is depressed. The Home key moves the cursor up-left and the End key moves it down-left. The Ctrl key, when used in conjunction with the alphanumeric keys, produces the IBM PC symbols associated with the various control codes. If you have the IBM Technical Reference Manual, the symbols are shown on page C-1. If not, just experiment.

These programs were written specifically for an IBM PC with a Color/Graphics Monitor Adapter. Any compatibility with other PCs or adapters is strictly coincidental. These programs will not work with a Monochrome Display Adapter, because both PUSH and POP read from, and write to, the color display memory at B800H. Also, the IBM Monochrome Display Adapter has no graphics capabilities.

Perhaps the simplest way to provide continuous, sequential viewing of screen-images files is to create BATch files with the filenames separated by PAUSE commands.

Editor's note: We have placed Mr. Lockwood's PUSH. COM and POP. COM programs on the IBM PC User Group Support Electronic Bulletin Board in the <F>iles section. For information about accessing our bulletin board system and downloading these files, see the article "IBM PC User Group Electronic Bulletin Board System" in this issue.

Writing TopView-Compatible Applications

Michael Engelberg
IBM Corporation

When you write a program for the TopView environment, your application has to follow certain guidelines to be compatible with TopView. Your application may not run under TopView if it does not adhere to the following guidelines:

1. Do not use BATch files.
2. Do not try to load your application at an absolute memory location.
3. Do not use INTerrupt 27H or INTerrupt 21H, function call 31H, to terminate and remain resident.
4. Do not modify hardware interrupt vectors (except keyboard) or directly control hardware that must be shared with other applications.
5. Do not access absolute memory locations except where required to control unique hardware.
6. Do not access memory outside the partition that your program was loaded into, except where required to control unique hardware.
7. Do not use direct memory access to change or read the DOS software vectors Termination (22H), Ctrl-Break (23H), and Critical Error (24H).
8. Do not use program loops to compute elapsed time.

Also, your application may not run normally if there is not enough memory to load it, or if a file that it requires is being used by another application.

Applications that can run in the background are considered "well-behaved." Well-behaved applications adhere to additional guidelines:

1. They do not directly access video memory except through the INTerrupt 10H call.
2. They do not use graphics video modes.
3. They do not read the BIOS keyboard buffer or change the BIOS keyboard data areas.

Following are detailed discussions of these guidelines.

Direct Video Memory Access: A program that writes directly to the hardware video buffer can run only when it is the foreground application. This kind of application takes over the entire screen, so you cannot use the Move, Size and Zoom functions of windowing. If this kind of program is switched into the background, it cannot operate there and will be suspended.

In the TopView environment the only proper way to access video memory directly is with a special subfunction of the BIOS call INTerrupt 10H that is defined in TopView. TopView uses this call to determine the address of an application's assigned video memory.

Therefore, an application that uses this call can access its video memory directly whenever it is processing in either the foreground or background. However, to run outside the TopView environment, the application itself must examine the video controller.

TopView provides a new set of video BIOS function calls that allow your application to access video memory directly, whether or not your application is operating in a TopView environment. Use of these function calls allows your program to run concurrently with other applications in the TopView environment and allows your program to be windowed. If your application is running outside the TopView environment, BIOS considers these new calls to be no-operation functions. All other video BIOS function calls operate as described in the IBM Personal Computer System Technical Reference manual and the IBM Personal Computer XT/IBM Portable Personal Computer System Technical Reference manual.

The new video BIOS function calls are:

Get Video Buffer (INT 10H, AH = 0FEH)

Input to this call is AH = 0FEH. On entry to the call, ES:DI contains the assumed address of the hardware video buffer. The values in ES:DI are:

B000H:0000H—for a Monochrome Display and
B800H:0000H—for a Color Display

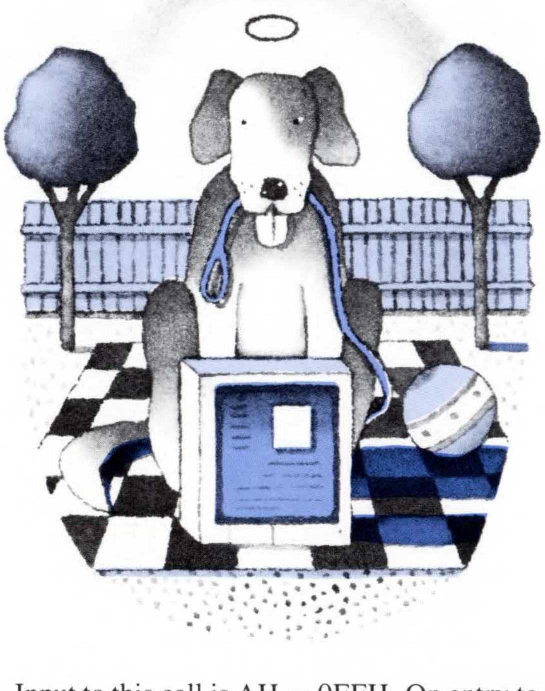
On return, ES:DI contains the actual address of the video buffer if your program is running under TopView. If not, then ES:DI is unchanged.

If the address of the actual video buffer is different from the address of the assumed buffer, the application must use the returned address to access its video buffer. In this case the video controller does not need to be polled for horizontal or vertical retrace. Video buffer access can be made at any time by the calling program.

If the address returned is B800H:0000H and the system is running in 80-column text mode, the video controller must still be polled for horizontal and vertical retrace before any accesses to the buffer are made. This occurs only when you are not running under TopView.

Update Video Display (INT 10H, AH = 0FFH)

You should make this call only when the video buffer has been changed, and you want your application to ensure that the user sees the changes on the display.



Input to this call is AH = 0FFH. On entry to the call, ES:DI points to the first character in the video buffer that has been modified. Also on entry, CX contains the number of sequential characters or attributes that have been modified.

On return, all registers are unchanged.

The Get Video Buffer call and the Update Video Display call apply only to applications running in a text video mode (0, 1, 2, 3 or 7). Incorrect operation will occur if these calls are issued from applications running in a graphics video mode (either 4, 5 or 6).

Following is a discussion of how to use the Update Video Display call:

Under the DOS/BIOS environment, there is no effect from using the Update Video Display call (INT 10H, AH = 0FFH). You don't need to make this call if the video buffer address returned by the Get Video Buffer call (INT 10H, AH = 0FEH) is one of the standard hardware addresses that you supplied in ES:DI. However, when the video buffer address returned by the Get Video Buffer call is not a standard hardware address, you must use the Update Video Display call to ensure correct program operation.

The frequency and manner in which you use the Update Video Display call can have a dramatic overall impact on your application's performance.

You don't need to issue the Update Video Display call immediately after making a change to the video buffer. You should issue the Update Video Display call only at the point where the application must guarantee that the user can see the updates on the display screen.

This means that you may have to make some tradeoffs when you design your application. For example, suppose ten characters are updated in non-contiguous, widely separated portions of the video buffer every time the user presses a function key. There are two acceptable practices. The first is to issue an Update Video Display call after each character is stored in the video buffer. Each call should specify the address of the updated character and a length of 1. This is the simplest, and usually the preferred, solution.

The second approach is to perform all updates to the video buffer and then issue a single Update Video Display call which specifies (1) the address of the first character that was modified in the buffer, and (2) a length that includes all characters between, and including, the first character modified and the last character modified. Note that the number of characters in this call can cause lines on the display to wrap; a 25-row, 80-column display could be entirely updated by specifying the start of the video buffer and a length of 2000.

Unless the modified characters are fairly close to each other, the first approach is usually better than the second. This is because the amount of time to perform an Update Video Display call is proportional to the number of characters in the call (i.e., in the CX register). Ten calls of length 1 are faster than one call of length 200. However, there is a threshold at which this is no longer true. Therefore 500 calls of length 1 are probably slower than one call of length 2000.

Determining the best method for using the Update Video Display call depends on the function of the application and the amount of user interaction.

- Never issue the Update Video Display call more often than necessary.
- If only a few characters are updated at a time, and the characters are at widely separated locations in the video buffer, issue single character requests to update each position rather than one large request.
- If a major update must be done, modify the buffer as required, then issue a single Update Video Display call which specifies the length of the entire buffer (or as much of the buffer as needs to be updated).

The frequency and manner in which you use the Update Video Display call can have a dramatic overall impact on your application's performance.

This discussion also applies to updates performed on display attributes. When an update is requested, the update applies to the attribute as well as to the characters at the specified locations in the video buffer.

If your application makes a request to change the video mode, its video buffer may be moved. You must re-execute the Get Video Buffer (INT 10H, AH=0FEH) function and use the returned video buffer address.

Graphics Support: TopView supports graphics applications by giving them control of the full screen when they switch to a graphics video mode. Both medium-resolution color (320 x 200 four-color pixels) and higher-resolution black-and-white (640 x 200 two-color pixels) video modes can be used for graphics applications.

Since most graphics applications write directly into display adapter memory, graphics applications can run only in the foreground with a full-screen window that cannot be moved, sized or scrolled. Several graphics applications can be started under TopView, but they are not allowed to run as background tasks.

When an application switches into a graphics video mode through the BIOS INTerrupt 10H, TopView allocates a 16KB image buffer for the application's logical window. An application that is currently doing graphics can switch back to a text video mode. At that time TopView deallocates the 16KB image buffer and allocates a 4KB logical window buffer to contain the characters and attributes.

Programming Restrictions

The following programming techniques can cause your application to be incompatible (i.e., not run) with TopView.

- **Absolute Memory Loading:** TopView loads applications anywhere in free memory. Your application should not try to access or load at absolute memory locations related to the application itself. All applications should use unsigned integer arithmetic when doing memory size calculations, so that applications that run in systems with more than 512KB of memory will run properly.
- **Accessing BIOS Keyboard Data Areas:** If a program reads the keys directly by reading the keystroke value from the BIOS keyboard buffer, its Program Information File should state that the program does not run in the background. If the program changes the BIOS keyboard data area, its Program Information File also should state that it does not run in the background.

> *Determining the best method for using the Update Video Display call depends on the function of the application and the amount of user interaction.*

If the Program Information File record is not set in this manner, keystrokes may be lost for other programs running under TopView, or not returned properly to the program that manipulates the BIOS keyboard data area.

- **Direct Control of System Hardware:** To provide certain system services (switching, concurrent execution, windowing, etc.) to several applications at one time, TopView must control the hardware related to those services. TopView can provide multitasking services only when it controls the hardware devices.

If your application controls a hardware device such as the video controller, it interferes with all other DOS programs running concurrently that must also access that hardware, and defeats the purpose of multitasking. Your application must be programmed to permit other applications to share the hardware.

An application can control some special devices. All applications, however, must be able to access the standard hardware devices at any time. For TopView to provide these services, applications must not modify standard hardware interrupt vectors directly. Well-behaved applications must explicitly open and close shared devices using standard system names. This allows TopView to provide resource sharing for applications that use the standard system devices.

- **Interrupt Vectors:** There are interrupt vectors for the hardware, the software and the keyboard.

Software interrupt vectors can be used for transferring control between sections of code in a program. For each task running under TopView, TopView maintains a copy of the software vectors within a range specified by the program's information file. This permits a program to use the user-assigned software vectors to share and transfer control between common code, whether or not other tasks in the system also use the same interrupt vectors.

You must make certain that the exact range of vectors used is specified correctly in the program information file. If they are incorrectly specified, the program may not run correctly.

TopView maintains the standard DOS vectors—termination (22H), Ctrl-Break (23H), and critical error (24H)—and your program information file need not include them in the range of vectors to be saved and restored. These vectors should only be set and read using the DOS function calls AH=25H (set vector) and AH=35H (get vector.) Direct memory accessing of these vectors will not provide the desired result.

Also, TopView maintains the Keyboard Break (1BH) and Timer Tick (1CH) interrupt vectors automatically for each task running in the system. As with the standard DOS vectors, these interrupt vectors need not be included in the range specified in the program information file.

TopView does not maintain the hardware vectors 08H to 0FH within each task, because these vectors stay the same from one task to the next. Programs should avoid controlling the hardware directly, so that two or more programs will not try to control the same hardware device.

The keyboard hardware vector 09H is handled differently. If a program changes the keyboard hardware vector, TopView notes the change and uses a new vector whenever a keyboard interrupt occurs. This vector is in effect only when the program that set it is the foreground program. In this way, each task running in the TopView environment can define its own keyboard interrupt handler.

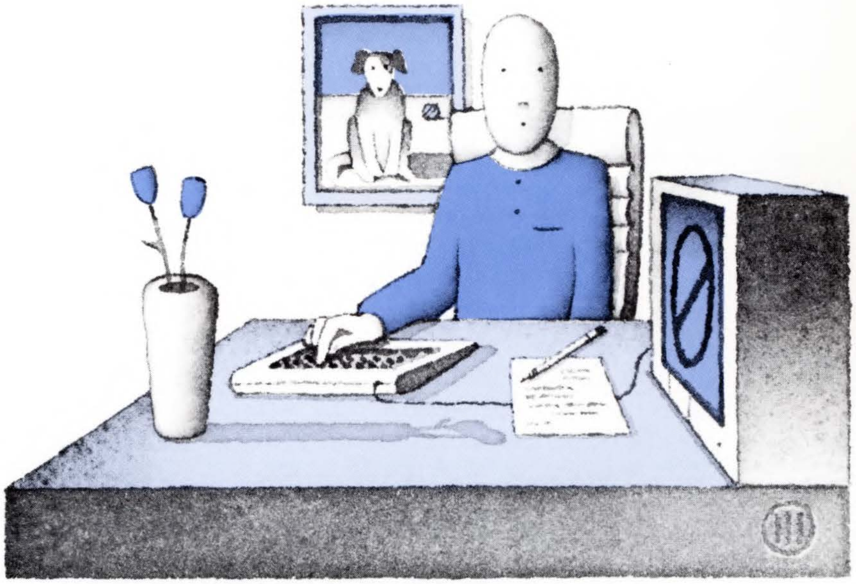
- **Multiple Applications:** Problems may occur if your program loads another program, and that program checks to see if it has been loaded in the same session and then tries to use the previously loaded copy of the program. An example is an application that uses a load-and-stay-resident structure to provide shared code. TopView may not be able to determine which task is running that code.
- **BATch Files:** TopView does not support BATch files. You cannot use BATch files to start, end or manipulate a file or program in the TopView environment.
- **Copy Protection:** If a DOS program has a copy protection scheme, that scheme must not violate any of the TopView compatibility rules. Any copy protection scheme that violates these rules may not run in the TopView environment.

  A program may not work under TopView if its copy protection scheme requires:
  —booting a special diskette or a modified DOS, or
  —AUTOEXEC loads only, or
  —critical timing to read disk sectors or tracks.

Supporting DOS File I/O

DOS 2.00 and higher versions provide multiple directory access, installable device drivers to handle non-standard devices, and redirected I/O for standard input devices. TopView provides multiple DOS tasks with the same environment as though they were single tasks running under DOS—each task has its own file environment, independent of any other running task.

This is implemented by defining an individual task file environment when the task is started. The task's file environment includes the default drive, default directory, write verification state and Ctrl-Break state. The default values for the first two are taken from the application's Program Information File. The write verification state and Ctrl-Break state are taken from the defined states at the time TopView is started.



Any application calls made to DOS are intercepted, and the application's current file environment is provided for the call. The application can change its file environment at any time. If the task is removed from the system and is restarted, the original default values (from the application's Program Information File) are restored.

DOS Restrictions

The DOS interface in the TopView environment is basically unchanged. However, there are a few restrictions:

- **File Handles:** DOS 2.00 and higher versions support the use of file handles. However, DOS restricts the total number of file handles in the system to a maximum of 20 at one time. When you use an ASCIIZ-type open, you open a file handle. As file handles are closed, they are returned to the system.

  Your application should avoid keeping files open the entire time the application is running. This will ensure that handles are available for other applications that are running concurrently.
- **File Locking:** TopView provides an automatic file locking mechanism intended to protect your file from other tasks that are updating the file you are using. However, multiple tasks can open the same file at the same time for read-only access.

  A lock is activated when a file is opened, and it remains active until the file is closed or the task is ended. The task that opens a file can access that file as many times as required; however, if the file was opened for write, read/write or using FCBs, no other tasks in the system can open that file for updates until it is closed.

If you want other copies of your application (which are running at the same time) to access the same files for updates, the files must be closed when they are not being used.

If a program uses overlays to segment the code, the overlay file should be closed when it is not being used. If it is left open, a second copy of the program cannot be started, because access to the overlay file is denied by the TopView file locking mechanism.

- **ANSI.SYS Keyboard Device Driver:** The ANSI.SYS keyboard device driver is not supported by TopView because once loaded, the device driver would affect all applications. TopView cannot load the ANSI.SYS device driver and selectively apply it. Applications that make calls to ANSI.SYS will not function normally, because TopView will not pass the call on to the device driver. However, if you load ANSI.SYS from the CONFIG.SYS file, it will not affect TopView.
- **DOS Command Interpreter:** The DOS command interpreter COMMAND.COM should not be loaded from TopView. When running under DOS 3.00, TopView allows COMMAND.COM to be loaded from the Start-a-Program menu, but many DOS functions, including all extended error functions, will cause TopView or application programs to fail, requiring you to reset (Ctrl-Alt-Del) the computer.

Testing Your Application's Compatibility with TopView

Although the TopView Programmer's ToolKit provides the tools and guidelines for TopView applications, you should perform certain testing to ensure that your application will run in the TopView environment and will co-exist with other applications that have been started.

Following are guidelines for testing your application's compatibility.

1. Make sure the information in your application's Program Information File is correct.
2. Test your application by itself in the TopView environment. Do these tests:
   —Exercise all of the application's functions.
   —Exercise all of TopView's functions.
   —Switch back and forth between TopView's functions and the application's functions.
3. Start multiple copies of your application in the TopView environment, and perform the tests in item 2.
4. Start other applications in the TopView environment and perform the tests in item 2 plus these additional tests:
   —If your application runs in the background, test all functions that run in the background while other applications run in the foreground.
   —Test your application in the foreground with other applications running in the background.
   —Hide and suspend your application.
   —Run your application while others are hidden and suspended.
   —Run your application and others using different start orders (i.e., start your application first, then start it last, and so on).
   —Attempt to use the printer from your application (if appropriate).

You should test your application for all possible combinations of hardware and software:

Levels of DOS: 2.00, 2.10, 3.00 and 3.10

Different system units: IBM Personal Computer AT, IBM Portable Personal Computer, IBM Personal Computer XT and IBM Personal Computer

Different amounts of memory: from 256KB up through 640KB

Different storage media: two double-sided diskettes, one double-sided diskette and one fixed disk. Also test a fixed disk that resides in an IBM Personal Computer Expansion Unit

Different displays: IBM Monochrome Display, IBM Color Display, and compatible 80-column color and black-and-white monitors

Different pointing devices: keyboard and the supported mice (listed below under Hardware and Software Requirements)

Programmer's ToolKit Product Contents

The TopView Programmer's ToolKit is distributed with:

- One diskette containing the TopView Programmer's ToolKit tools
- One diskette containing the sample high-level programming package
- TopView Programmer's ToolKit Reference manual
- TopView Programmer's ToolKit Quick Reference card

Programmer's ToolKit Hardware and Software Requirements

The TopView Programmer's ToolKit requires the following minimum configuration:

- An IBM Personal Computer AT, IBM Portable Personal Computer, IBM Personal Computer XT or IBM Personal Computer
- 256KB of random access memory (although 512KB or more is recommended)
- Two double-sided diskette drives or one double-sided diskette drive plus a fixed disk drive
- An 80-column display and its corresponding adapter card. The IBM Monochrome Display requires the Monochrome Display and Printer Adapter or the Enhanced Graphics Adapter. The IBM Color Display, or a compatible monitor, requires the Color/Graphics Adapter or the Enhanced Graphics Adapter. The IBM Enhanced Color Display requires the Color/Graphics Adapter or Enhanced Graphics Adapter. The IBM Professional Graphics Display requires the Professional Graphics Adapter running in emulator mode.

  Note: TopView supports the Enhanced Graphics Adapter and the Professional Graphics Adapter only when they are configured to run in compatibility mode. The extended graphics modes of both the Enhanced Graphics Adapter and the Professional Graphics Adapter are not supported by TopView.

  When you use the IBM Professional Graphics Adapter, you must configure TopView to use the keyboard or the parallel mouse pointing device. TopView does not support the serial mouse with the Professional Graphics Adapter.
- DOS 2.00, 2.10, 3.00 or 3.10
- The TopView program, which provides the overall TopView environment

  A parallel printer is optional.

  A customer-supplied mouse is optional. TopView and the Programmer's ToolKit support four specific mice:

—Microsoft Mouse for IBM Personal Computers™ (Parallel Interface), part number 037-099

—Microsoft Mouse for IBM Personal Computers™ (Serial Interface), part number 039-099

—PC Mouse by Mouse Systems,™ Inc., part number 900120-214 (serial interface)

—Visi On Mouse by VisiCorp,™ Inc., part number 69910-1011 (serial interface)

IBM has tested these devices with TopView functions as of the date of announcement. However, IBM does not endorse or recommend one non-IBM product over another, and does not warrant these devices in any way.

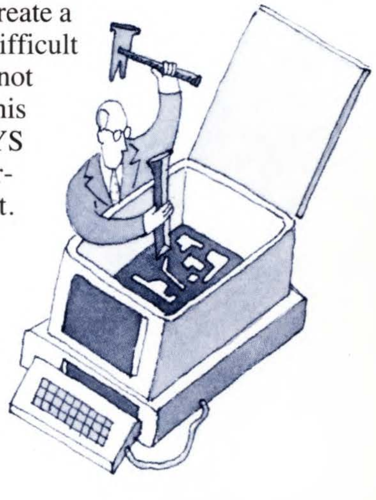
The DOS CONFIG.SYS File

R.K. Schmitt
Southwestern Michigan PC Users Group

The CONFIG.SYS file allows you to configure DOS to your specifications (to a certain extent). This capability is very convenient, considering that not too many years ago, the logic that ran the available options (one or two floppy diskette drives, a serial or parallel printer, and a color or monochrome display) was placed on printed circuit cards, and the choices facing the operating system were limited—everybody could use the same one.

However, to their everlasting credit, IBM opened the architecture of the PC and let any interested party know everything there was to know about the machine. This led to the explosive growth of the PC marketplace, especially for third-party add-on devices.

Thus, while you can still use the default setting of your operating system, which will probably satisfy most people, using special facilities or overcoming particular problems requires that you modify the operating system's settings using some of the special features provided as part of DOS. One of the ways to modify DOS settings is to create a CONFIG.SYS file. The file is not difficult to create or use, and the result is not obtrusive. In fact, while writing this article, I discovered a CONFIG.SYS file I had created and completely forgotten about—so don't be afraid of it.



First, boot your machine with all RAM-resident items eliminated and check the size of the operating system that remains resident in memory. To do this, use the CHKDSK command. The last portion of the results displayed on the screen will show you how much memory is installed and how much is available. The difference is the memory consumed by the operating system. As you make changes, you can do a CHKDSK periodically to see what is happening to the size of the operating system.

Editor's note: An IBM Personal Computer AT cannot execute the DOS CHKDSK command when operating in extended memory.

CONFIG.SYS is not just a command—it is a file containing configuration commands. When you boot (IPL) your computer, the system checks for a CONFIG.SYS file present on the drive from which the operating system is being loaded. If the CONFIG.SYS file exists, the commands in the file are used to configure the operating system.

Note: Information in this article is written in context of DOS 3.00. If you are not yet using DOS 3.00 or DOS 3.10, but are using at least DOS 2.00, you will be able to use some of the information.

In the following listing of configuration commands, the boldfaced option is the default value chosen by DOS.

BREAK=**OFF** / ON
Example: BREAK=ON

If BREAK is set to OFF, the Ctrl-Break key combination will work only when a program requests standard I/O operations. If it is set to ON, the system will continually check to see if the Ctrl-Break key combination has been pressed. BREAK=ON will slow down the operation of the machine, but not noticeably.

BUFFERS=1 / **2** / 3 (AT default) / ... 99
Example: BUFFERS=20

This command determines how much memory should be set aside to be used as intermediate storage between disk files and the loaded program. All data involved in disk I/O (coming from the disk to your program or going from your program to disk) pass through buffers. The more buffers you have available, the more likely it is that the data may already be present in memory, negating the need for a disk access.

Data base applications recommend or require that the number of buffers be increased beyond the standard 2 or 3 up to a value between 10 and 20, just to reduce the number of times DOS accesses the disk drives. Specifying 99 buffers is not a panacea—it takes up more memory and increases the time needed for DOS to check whether the data is already present in memory. Experimenting is the only way to find the best setting, and the setting you specify will be reliable only if the BUFFERS setting was the only setting changed. In other words, besides satisfying your curiosity, there is probably little or no value in specifying more than 10-20 buffers.

COUNTRY=xxx (001 for the U.S. is default)
Example: COUNTRY=358

There is probably little or no value in specifying more than 10-20 buffers.

The 358 setting shown in the example sets the date, time, currency symbol and decimal separator to that used in Finland. You probably won't have any reason to change this setting from the default.

DEVICE=filename.ext (no default)
Example: DEVICE=ANSI.SYS

This command is used to load device drivers. The subject of device drivers is complex, and other than these few paragraphs, is beyond the scope of this article. Most likely, you will never have a need for them unless you buy a piece of specialized hardware; and, in that case, the supplier would probably not only provide the device driver but would also give you specific instructions for incorporating it into your CONFIG.SYS file.

Since IBM distributes two device drivers—ANSI.SYS (with DOS 2.00 and higher) and VDISK.SYS (with DOS 3.00 and higher)—I will briefly discuss them here.

ANSI.SYS loads the ANSI standard screen and keyboard routines rather than the default IBM settings. Using ANSI.SYS will slow your system (you probably would not be aware of the difference) and increase the resident memory of DOS. However, some programs are designed to run better when you load ANSI.SYS. Try it and see if you like it better. If you don't, you can always erase the command from your CONFIG.SYS file. If you add this device driver, be sure that the ANSI.SYS file is in the root directory or that you specify the path.

VDISK.SYS creates a virtual (RAM) drive. You do this by entering the command:

```
DEVICE=VDISK.SYS mem sss ddd
```

where mem is the KB size of the drive's memory capacity; sss is the sector size with allowable values of 128, 256, or 512; and ddd sets the maximum number of directory entries for the drive. A setting of

```
DEVICE=VDISK.SYS 128 256 64
```

would set up a virtual drive with 128KB of memory, 256-byte sectors, and a maximum of 64 directory entries. The amount of memory you choose to set aside in your virtual drive will depend on how much memory your system has, and how much is available after you load your regular programs.

Default settings for VDISK.SYS are 64 128 64. Notice that you separate the parameters with spaces rather than commas. Again, if you load this device, be sure that the VDISK.SYS file is in the root directory, or that you specify the path.

```
FCBS=....
```

The FCB stands for "File Control Block." This command has been put in the system in preparation for the arrival of IBM's networking system and DOS 3.10 and is used only when sharing files on a network. If you're interested in this command, you can read more about it in chapter four of the DOS 3.00 or DOS 3.10 manual.

```
FILES=8 / 9 /... 255 /
Example: FILES=16
```

Unless you're a programmer, the explanation in the DOS manual probably makes no sense.

Editor's note: In DOS 2.00 and higher, the operating system calls allow file access (reads, writes, closes) to be performed using a 2-byte "handle" instead of always using the traditional File Control Block (FCB). The FILES command allows you to set the maximum number of file handles (up to 255) for an entire system, but you should remember that the maximum number of file handles that a process can have concurrently open is 20. The resident portion of DOS increases by 48 bytes for each additional file above the default value of 8. Since many programs don't take advantage of the calls that allow the 2-byte handle, you don't need to worry about this command unless you get an error message indicating you do not have enough handles.

```
LASTDRIVE=A / B / E /... Z
Example: LASTDRIVE=F
```

This command lets you specify the highest drive letter that the system will recognize. Users who have systems with four or more physical disk drives and who use the VDISK.SYS command to set up a virtual disk drive might use this command. VDISK.SYS installs the virtual drive as one letter beyond the last physical drive of your system, but unless the drive letter is beyond E, the default setting, you don't need to use this command.

```
SHELL=filename.ext
```

This command lets you load a command processor in place of COMMAND.COM. I assume you will not have occasion to use any other command processor, and if you do, you will already know how to use the SHELL command.

If you have plenty of memory and have not tried a virtual (RAM) drive, try one. You will be amazed how much more quickly things can happen.

To build a CONFIG.SYS file, you can use EDLIN or any other editor that can create ASCII files. You also can create it directly from the console using the commands below as an example:

```
COPY CON: CONFIG.SYS <CR>
BREAK=OFF <CR>
BUFFERS=2 <CR>
COUNTRY=001 <CR>
FCBS=4 <CR>
FILES=8 <CR>
LASTDRIVE=E <F6> <CR>
```

where <CR> indicates that you press the Enter key, and <F6> indicates that you press the F6 function key. Obviously, you would substitute different settings from the ones shown above, since those shown are the default settings.

Be aware that the various options and facilities will increase the amount of memory needed to hold your operating system. For example, I booted the system using DOS 3.00 under a variety of conditions and then did a CHKDSK to see the amount of memory occupied by the operating system. The results were as follows:

		DOS Size	Increase
default		38,928	—
BUFFERS=15	(from 3)	45,264	6,336
DEVICE=ANSI.SYS		46,848	1,584
FCBS=20	(from 4)	47,744	896
FILES=20	(from 8)	48,416	672

Other commands such as ASSIGN, GRAFTABL, GRAPHICS, KEYBxx, PRINT and SHARE are not part of a CONFIG.SYS file but also will increase the resident portion of DOS.

Working with the CONFIG.SYS file may be a case where ignorance is bliss. If everything is working fine and you are happy, don't feel that you must create a CONFIG.SYS file. On the other hand, if you have several devices, are heavily using a data base management system with lots of disk accessing, etc., you might experiment to see if the performance can be improved. If you have plenty of memory and have not tried a virtual (RAM) drive, try one. You will be amazed how much more quickly things can happen. However, when using a RAM drive, don't forget that RAM memory is volatile—whenever the machine is turned off, or the power fails (even flickers), the drive is cleared.

Happy CONFIG.SYSing.

The XENIX Operating System

Bob Mix
Sacramento PC Users Group

XENIX is the new multiuser operating system offered by IBM for the enhanced IBM Personal Computer AT. There are three major XENIX packages available: the Base System, the Text Formatting System and the Software Development System. The Base System is required to run XENIX. Professional word processors, printers and publishers will be interested in the Text Formatting System, because it can feed verbiage to typesetters. Professional C programmers will want the Software Development System.

IBM's XENIX is Microsoft's adaptation of AT&T's Unix. Since Microsoft developed both DOS and XENIX, you will find many similarities, such as the hierarchical file system, redirection, pipes, filters, and some commands. In fact, Microsoft has been working for the last couple of years to merge the two operating systems so they will operate in an identical work environment. Ideally, this means DOS software will eventually be usable without change on XENIX. Until then, XENIX includes several programs to permit easy two-way transfer of files between DOS and XENIX via diskette, but not across fixed disk partitions.

The XENIX Base System costs six times as much as DOS, but you easily receive proportional value—in bulk alone. The package includes three slipcases containing five manuals and four high capacity (1200KB) diskettes. IBM standard XENIX will service up to three simultaneous users.

Installation is simple. IBM explains it in 16 pages, and minimum time required is under an hour.

I failed to install it on my 30MB disk, so I tried installation on a 20MB disk. That went smoothly, so I lied to my AT, telling it I have a 20MB disk. I erased my DOS partition and reformatted. This time all went well.

XENIX offers the option of dedicating the drive to XENIX, or sharing the drive with another operating system. (I reserved 5.5MB for DOS.) The FDISK command on either system allows you to specify which one "boots," XENIX or DOS.

After completing all steps in the Installation Manual, you will have a single-user operating system without operational peripherals. Additional printers, terminals and modems must be installed using instructions in the System Administration Manual. Following the manual, I had no trouble attaching peripherals.

Printing under XENIX is more complex than under DOS, because multiple users may be trying to print at the same time. To avoid conflicts when printing multiple documents (and to avoid having to cut up a printout into ticker-tape strips, sort them by hand, and then tape the associated strips to the blank sheet of paper), XENIX has a program called a "printer daemon" which runs continuously and handles printing. This little daemon is possessed. I printed a three-line file and received 5 pages of output due to mainframe mind set and "separator" pages. (The documentation mentions other daemons, but nary a wood nymph.)

When XENIX starts up, you are given the option of entering system maintenance mode or multiuser mode. Only the System Manager will ever do system maintenance. Who is the "System Manager"? He/she/it is the person who imposes order on your normal chaos. The size and power of XENIX requires a responsible, intelligent, meticulous individual to maintain it. The volume of work is small, but a maintenance schedule must be rigorously followed. The consequences of failure are generally lost or corrupted data.

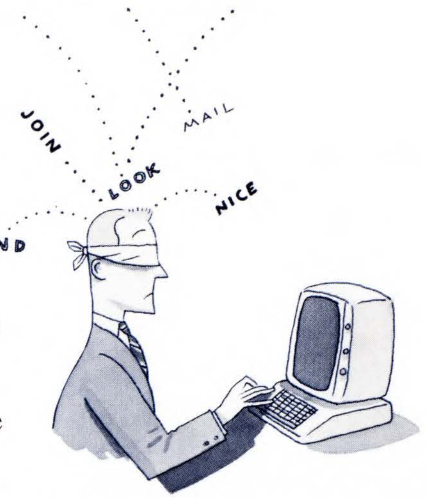
XENIX is full of goodies. I counted over 200 commands, some of which cascade further commands upon the blindfolded beginner. Even for those who have mastered complex programs, XENIX is a challenge.

XENIX lets you pick from three shells. A XENIX shell is a complex program that enables users to use the XENIX kernel without having to know very much; it is the equivalent of the DOS Command Interpreter (COMMAND.COM), but goes much further. For those who can learn shell programming, the shell is a giant lever to move the earth of XENIX.

A kernel is an operating system without the human interface. The XENIX kernel uses 121KB of RAM and buffers another 40KB. The rest is available for shells and user programs. Each user has a choice of shells — Bourne, Visual and C — each of which may be tailored to individual requirements.

XENIX is an outstanding bargain, especially if you consider the per-user cost for multiple users.

IBM appears to favor the Bourne shell (since it is the only shell documented in the Basic Operations Guide) and Microsoft favors the Visual shell. If you use Microsoft application programs such as Word or Multiplan, then you will feel at home in the Visual shell. Without learning any command syntax, you can perform complete file maintenance chores and run any program. Key assignments minimize strokes. Neophytes should start with the Visual shell and graduate to the Bourne shell.



The Bourne shell serves as command interpreter and high-level language. You can use it to initiate multiple simultaneously-executing processes (programs). Processing can be done in the background or even be scheduled for execution during off hours. Shell procedures may be used to automatically control every resource of XENIX. All you need is a programming muse.

XENIX contains many programs that you would expect to pay extra for. You get four editors: Ed, Ex, Vi and Sed; several kinds of communication software: remote, network, and electronic mail; and two programmable calculators. You also will find a reminder service that watches the clock for you. You can print banners and calendars with Banner and Cal, and you can reassign function keys with Setkey.

XENIX is an outstanding bargain, especially if you consider the per-user cost for multiple users. The flip-side is the application software situation. Compared to DOS, XENIX choices are few and relatively expensive. However, if you have a serious need for a multiuser operating system, then XENIX is the best available choice on microcomputers.

Obtaining a Filename Parameter in BASIC

Joe Kilar
Borg-Warner Research Center IBM PC Users Group
Des Plaines, Illinois

While writing the latest version of my public domain BASIC program, a screen-oriented text editor called SCREDIT, I wanted to add a feature that would allow a filename parameter to be entered with the SCREDIT command on the DOS command line. In other words, to edit the "filename.ext" file, a user would type "SCREDIT filename.ext" at the DOS prompt. Previous versions of my program used the BASIC INPUT command to prompt the user to enter a filename.

Documentation in the IBM DOS Technical Reference Manual tells how to accomplish this for a .COM file, but not for an .EXE file created using the IBM BASIC Compiler. I have discovered a method and am providing sample programs to show how to include such a feature in your own compiled BASIC programs.

In technical terms, the filename in FCB format appears at the memory location 163 bytes below the beginning of the code segment (the place CS points to).

An IBM Macro Assembler source subroutine is shown in Figure 1. This subroutine, which I call GETFILE.ASM, can be assembled into an object file GETFILE.OBJ. A BASIC program segment is shown in Figure 2. It allows the file name obtained from the subroutine to be stored as a BASIC character string. Standard OPEN statements can then be used to allow file input/output. Finally, the DOS LINK command is used to link the compiled BASIC program with GETFILE.OBJ to create the executable program.

Here is a summary of the steps required to obtain a file name parameter in BASIC:

1. Insert instructions similar to Figure 1 in your BASIC program at the appropriate place to get a file name and open it.
2. Use the IBM BASIC Compiler to compile the program.
3. Using EDLIN or another ASCII file editor, create a file named GETFILE.ASM as shown in Figure 2.

```
CREATE GETFILE.ASM
DISSEG  SEGMENT PARA   PUBLIC'CODE'
        ASSUME  CS:DISSEG
        PUBLIC  GETFILE
GETFILE PROC    FAR
        PUSH    BP
        MOV     BP,SP
        PUSH    ES
; address of string,  point BX to string
        MOV     BX,[BP + 6]
        MOV     AX,[BX + 2]
        MOV     BX,AX
        MOV     DX,BX          ;save in DX
; point SI at filename from DOS
        MOV     AX,[BP + 4]    ;BASIC's value of CS
        SUB     AX,16
        MOV     ES,AX
        MOV     SI,92
; determine drive
        MOV     AL,ES:[SI]
        INC     SI
        CMP     AL,0
        JNZ     TST1
        MOV     AH,' '
        MOV     AL,' '
        JP      GTDRNO
TST1:   ADD     AL, 64         ;convert to ASCII char
        MOV     AH,':'         ;supply : for drive spec
GTDRNO:MOV      [BX],AL        ;move to string location
        INC     BX
        MOV     [BX],AH
        INC     BX
; move the 8 byte filename
        MOV     CX, 8
        CALL    MLP
; insert the dot for suffix start
        MOV     AL, '.'
        MOV     [BX],AL
        INC     BX
; move the 3 byte suffix
        MOV     CX,3
        CALL    MLP
; restore registers and return
        POP     ES
        POP     BP
        RET     2
GETFILE ENDP
; subroutine MLP - move loop, moves No. of
; chars in CX from ES: [SI] to DS : [BX] in order
; to move from FCB area to string in BASIC
MLP     PROC    NEAR
        MOV     AL,ES:[SI]
        MOV     [BX],AL
        INC     BX
        INC     SI
        LOOP    MLP
        RET
MLP     ENDP
DISSEG  ENDS
        END
```

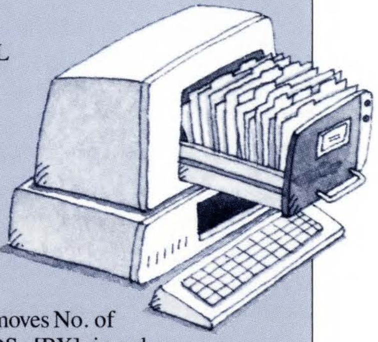


Figure 1. Macro Assembler Subroutine

4. Use the IBM Macro Assembler to assemble GETFILE.
5. To create the executable program, issue the DOS command LINK basfile + GETFILE, where basfile is the name of the BASIC program.

GETFILE.ASM as written assumes that the file is on the current default drive. If not, it can be easily modified to allow a drive prefix (e.g., B:).

```
10 ' Sample BASIC program segment that uses file name
11 ' from DOS command line obtained using subroutine
12 ' GETFILE.
39  Dummy name to reserve enough string space
40  F$="ABCDEFGHIJKL"
49 ' Call to subroutine
50  CALL GETFILE(F$)
59 ' OPEN file
60  OPEN F$ FOR INPUT AS #1
```

Figure 2. BASIC Segment

Application Development Guidelines for the IBM PC Network

Lou Davis
IBM Corporation

The requirements for application development on the IBM Personal Computer family of products have changed significantly with the introduction of more sophisticated systems such as the IBM PC Network. Application developers should carefully consider the new environments that these new systems create when they are developing or adapting their software.

Important considerations for the networking environment include application installation, application start-up or initialization, and application addressing of programs, data and devices.

There are three possibilities for applications in the network environment: (1) "transparent" applications that operate the same way whether stand-alone or on the network, (2) "network-wise" applications that determine whether they are on the network or not and alter their operation accordingly, and (3) "network-specific" applications that are coded to run only on the network.

In case (1), network transparency is the most important consideration. This type of application corresponds to the existing "compatibility mode" applications. These are well-written, well-behaved applications that can be installed and run on the IBM PC Network with no modifications, but which don't allow the user to make full use of the PC Network Program's functions and capabilities.

Transparency in the networking sense means that the application makes no assumptions about where its program code or data resides, or what kind of computer configuration it is running on. Instead, the application relies on setup actions to establish its addressability before the application is started.

A transparent application can be run either stand-alone or on the network. In the network environment, the user can set up network connections either directly through the IBM PC Network Program or through BATch files before invoking the application. The application then runs almost exactly like it does stand-alone, unaware that the devices and data it is referencing are actually on the network.

In cases (2) and (3) above, advanced network functions and networking efficiency are the most important considerations. These applications give the user the ability to manage a network environment more completely.

To be network-wise or network-specific, an application must meet a different set of requirements: it should be coded to use network

Table 1. Application Organization

	Network-Transparent	Network-Wise or Network-Specific
Installation	Use of APPEND command, READ ONLY attribute to adapt application to multi-user environment	Program/overlay sharing Separate user profiles Unique temporary files and work files
Startup	NET USE/NET SHARE sets up environment	User can dynamically specify full network paths through prompts
Operation	Application references drives and printers as if local	Application does dynamic redirections to resources, releases them when not being used
Error Handling	Errors mapped to stand-alone environment	Full network errors reported; error recovery based on actual location of error

functions and interfaces in order to be as efficient as possible in the networking environment, allow the user to fully address network devices and data through prompts in the application, and co-operate fully with other applications or invocations of the same application when sharing devices, data and programs.

Table 1 summarizes the differences in application organization and operation.

Table 1 clearly shows the advantages of developing either network-wise or network-specific applications. IBM highly recommends that developers take one of these two approaches when developing their applications.

Whichever approach the application developers take, they must follow certain guidelines to ensure the application operates properly in the IBM PC Network Program environment.

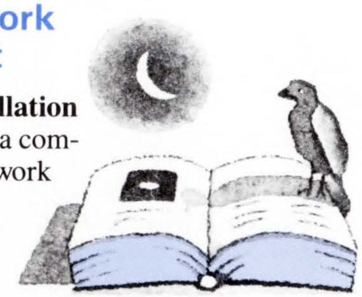
Application developers should always use the highest level of system interface to accomplish what they are trying to do. This means using regular, published DOS interfaces first, then PC Network Program interfaces, and BIOS interfaces last. They should avoid using hardware interfaces (direct port I/O) whenever possible. For example, to print characters on LPT1, developers should use the DOS INTerrupt 21H function 3DH OPEN FILE and function 40H WRITE TO A FILE OR DEVICE instead of INTerrupt 17H. Refer to Chapter 5, "DOS Interrupts and Function Calls" in the DOS Technical Reference, Version 3.10, for more information about DOS interfaces. See "Controlling the Printer" later in this article for more detail on printer interfaces.

Using regular high-level interfaces ensures a higher degree of transparency, portability and upward compatibility of an application across systems, and generally leads to a more well-behaved application in the networking environment.

In addition to the DOS guidelines, the PC Network Program has special requirements. These requirements are discussed below in the following categories:

- General guidelines for the networking environment
- Multi-user considerations
- Co-existence of applications with the PC Network Program (use of resources and conflicts in using certain functions)
- Application-to-application communication on the network
- Error recovery and problem determination

General Guidelines for the Network Environment



Application Installation

When installed on a computer in the PC Network environment, an application should not adapt itself to the hardware configuration of that computer. Although installed on a network server, the application may actually execute on another computer connected to the network whose hardware configuration is different. Instead, the application should be organized internally to (1) load an initialization program that determines, at application start-up time, the configuration of the computer on which the application has been invoked, and then (2) load the remaining parts of the application program based on that configuration. Configuration differences may include display type, printer, and number and type of disk drives.

Application Startup

In the past, on stand-alone computers, many applications were designed to be started by Ctrl-Alt-Del or by turning the computer on. The application was usually on a diskette that also had the correct version of DOS for the application.

In a networking environment this design does not work. The PC Network Program and the correct version of DOS to support it will have already been started. Some network redirections and PC Network Program setup will probably already have been done (e.g., the desired PC Network Program configuration, buffer sizes, message logging, additional names and other PC Network Program parameters). Applications should be designed to start at the DOS command line as an executable file or through a BATch file.

Network Resource Addressing
To achieve application transparency, applications should contain no hard-coded drives or paths. The application should not stipulate which drive the application program or the data is on, even if the restriction is based on equipment flags or other equipment determination of the computer from which the application is invoked. The PC Network Program environment can dynamically extend the computer's effective equipment (drives and printers), and the user should not be prevented from using these additional resources.

The user should instead be prompted for the entire device, drive or path information at program start-up or during application execution. This information also can be passed to the application through a BATch file.

The application program's environment, pointed to by offset 2CH in the Program Segment Prefix, can also be used to determine the location of overlays, as well as help and message files. See Chapter 6, "DOS Control Blocks and Work Areas" in the DOS Technical Reference, Version 3.10.

Multi-User Considerations

Use of File Control Blocks (FCBs)
While FCBs are still supported in DOS 3.10 and the PC Network Program, their use is more restricted (see Co-Existence of Applications with the PC Network Program below). Moreover, the new DOS sharing modes are not available through the FCB interface. For optimum performance, applications should use the DOS interrupt 21H handle function calls in the PC Network Program environment.

Use of DOS File Sharing on the PC Network
Applications should use the new sharing modes when opening their program and data files. A program file should be opened for READ ONLY with DENY WRITE mode sharing, because it will not be modified and because this mode allows multiple users to load the same application concurrently. Data files should be opened according to how they will be used (READ ONLY or READ/WRITE) and how they can be shared with other users. Normally this will be DENY READ/WRITE mode, so that the data will not change while it is being used. However, for applications that permit multiple concurrent updating of data, the DENY NONE mode can be used, and byte locking should then be used to prevent concurrent update of the same portion of the file.

Figure 1 gives the normal sequence of operations during multiple update sharing. In Figure 1:

1. Applications A and B open the same file for READ/WRITE, DENY NONE.
2. Application A locks the byte range in the file corresponding to the data it wishes to read and update.
3. Application A then reads the locked data, which is guaranteed to be the latest copy of the data.
4. Application B tries to lock the same data (even if it wants only to read it) and fails. This operation can then be retried repeatedly for a count, or after a specified time delay, or after prompting the user that the data is temporarily not available and subsequently receiving the user's request to try again. Infinite retry loops should never be used.
5. Application A modifies the data read, writes it back out to the file, and then unlocks the byte range of the data. This whole sequence should take as little time as possible to ensure that other users can access the data in a reasonable amount of time.

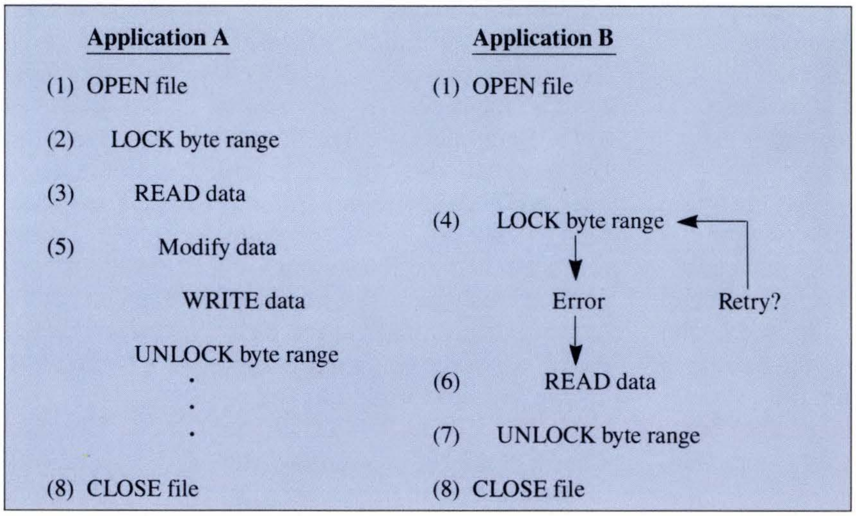


Figure 1. Multiple Update Sharing

6. Application B now succeeds in locking the byte range of the data and can read it with the assurance that it is the latest copy.
7. Application B unlocks the byte range of the data when done.
8. Both applications close the file.

The IOCTL Change Sharing Retry Count function call can control retrying of all sharing errors except Lock/Unlock File Access.

If the data to be locked is in more than one place in the file, or multiple locks are necessary in more than one file, then the applications must establish a semaphore mechanism or hierarchical order in locking the data to avoid deadly embrace. Avoiding deadly embrace is the application's responsibility. This mechanism is similar to GATE/UNGATE or ENQUEUE/DEQUEUE operations in system code.

Files that are loaded entirely into the remote computer's memory for processing, such as spreadsheets and word-processor documents, should be left OPEN during processing whether they are locked or shared. Otherwise, other users can access them, and every user who writes the file back will write over all the previous copies. A file must be OPEN if it is to be locked in a sharing environment. A file should be closed only after the last update is complete.

If an application is trying to extend an existing file past the current end of the file, the application should be programmed as depicted in Figure 2 to ensure that it is really writing at the current end of file. In Figure 2:

1. The application opens the file.
2. The application first locks the entire file (position 00000000H, length FFFFFFFFH).
3. The application does an LSEEK operation to the EOF.
4. The application writes the data, extending the EOF.
5. After writing the data, the application unlocks the entire file.
6. After all updates are complete, the application closes the file.

```
(1) OPEN file
(2)   LOCK entire file
(3)     LSEEK TO EOF
(4)     WRITE data
(5)   UNLOCK entire file
(6) CLOSE file
```

Figure 2. Extending the End of File

Special Rules for Applications on a Server

On an IBM PC Network Program file server, both the user and the application have a slightly different status from users and applications on remote computers. Because NET USE on a server to a disk, diskette or directory that has been NET SHAREd on that same computer is not permitted, all DOS file requests on the server go directly to DOS, bypassing the PC Network Program access checking at the directory level. This means that although applications on the server are still subject to DOS file sharing rules, they can write into directories that have been NET SHAREd as READ ONLY. Developers must consider this bypass when developing applications where an invocation of the application might run on the server, if the application uses data in a directory that has been NET SHAREd as READ ONLY. The DOS ATTRIB command can be used instead to make each file in the directory READ ONLY.

Temporary Work Files

Many applications use temporary work files. In most cases, the temporary files are not multi-user shared files. If the application uses a fixed name for these files and is run concurrently by several users, problems can occur. To solve these problems, all work files should be created by the DOS Create Unique File function (5AH) to ensure that each occurrence of the application will get a private copy of the work file.

Creating Files on the PC Network

When prompted to open a file, most applications check to see if the file exists. If the file does not exist, the application creates the file. Creating the file can cause problems in a multi-user environment because one application may check for the file and find that it does not exist, but before the file can be created, another application (or another occurrence of the same application) creates the file. In this case, the first application will destroy any data the second application may have put into the file. To avoid this problem, applications should use the DOS Create New File function (5BH). This function executes the check for existence and the file creation in one uninterrupted sequence. Create New File also returns an indicator if the file already exists.

The following PC Network programming interfaces and the DOS 3.10 extensions for sharing can be used to provide file and record locking control:

- DOS 3.10 Extensions

  Get Extended Error Code
  Create Temp
  Create New File
  Changeable IOCTL

(See the DOS Technical Reference, Version 3.10 for more information.)

- Sharing Extensions

  Open File Handle
  Lock Block
  Unlock Block
  IOCTL Set Retry

  (See the INTerrupt 21H function calls in the DOS Technical Reference, Version 3.10 for more information.)

Co-Existence of Applications with the PC Network Program

Applications should be coded to determine the presence of the PC Network Program (see Appendix C, "Programming Interfaces" in the PC Network Program User's Guide for more information). The restrictions listed below must be observed to ensure successful coexistence of applications with the PC Network Program.

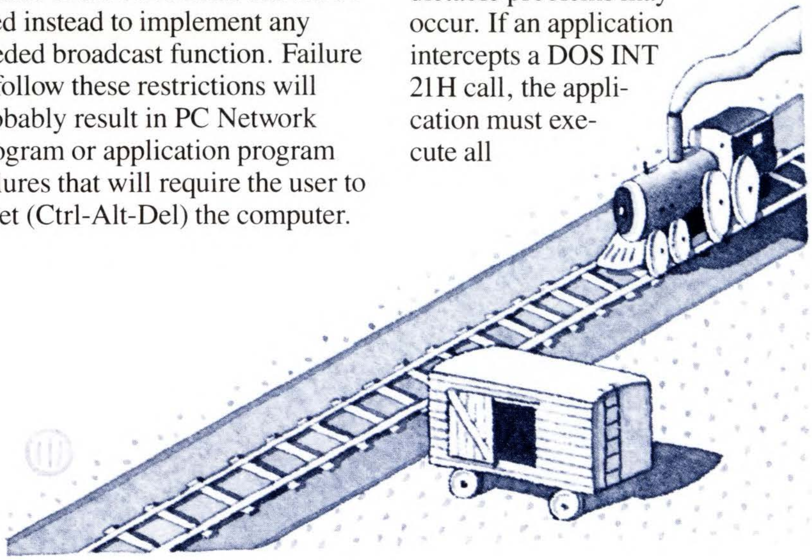
Use of NETBIOS
Any applications coded to use the NETBIOS interface (either by the EXECNCB Interrupt 2AH or by Interrupt 5CH) must not use all of the available functions. In particular the Reset (32H) and Receive Any to Any Name (16H or 96H to an NCB_NUM of 0FFH) should not be used. Also, applications should not use NETBIOS names added by the PC Network Program. Use of the name and name number returned by the DOS Get Machine Name (5E00H) function is allowed. Applications should not create a NETBIOS name with the 16th character position set to the values 00H-1FH. These values are reserved for names added by the PC Network Program. It is recommended that the 16th character position be a blank (20H) to ensure that name duplication with the PC Network Program can be detected.

Any application that issues NETBIOS requests must ensure that the required network resources exist. Failure to ensure that the resources exist can result in unpredictable errors in both the PC Network Program and the network application. The Get Network Resource Information function call (0500H INT 2AH described in Appendix C of the PC Network Program User's Guide) should determine which resources are available. If the values found are not sufficient, the application should instruct the user to restart the PC Network Program and specify larger values for the /SES and /CMD parameters.

Applications should not monitor NETBIOS requests issued by the PC Network Program, and under no circumstances should the requests be altered in any way. Use of the Receive Broadcast Datagram (23H or A3H) is not allowed. The Non-Unique Name feature of the NETBIOS should be used instead to implement any needed broadcast function. Failure to follow these restrictions will probably result in PC Network Program or application program failures that will require the user to reset (Ctrl-Alt-Del) the computer.

Replacement of Interrupt Vectors
Any application that replaces interrupt vectors must take special care when the PC Network Program is loaded. In particular, no interrupt should be replaced or set to some predefined value. In all cases, the old vector contents must be saved and control transferred to the saved address before leaving the interrupt service routine. This is true for hardware as well as software interrupts.

The PC Network Program makes strong use of the hardware timer (vector 08) and keyboard (vector 09) interrupts. This means that these interrupts must be passed to the PC Network Program for it to work correctly. Applications should not alter or reset the keyboard buffer based on what they consider to be invalid keys or key sequences, because such key sequences may be important to other applications in the same computer, especially the PC Network Program. Also, the PC Network Program assumes that each Interrupt 08 (timer) represents the passage of 1/18.2 seconds of time. If this is not true, unpredictable problems may occur. If an application intercepts a DOS INT 21H call, the application must execute all



possible functions or pass control to DOS to do them. If control is passed to DOS, then interrupts should not be enabled before control is passed. Failure to do this will cause the PC Network Program to fail.

Applications that have interrupt handlers that service hardware interrupts should be careful about what the application does in its interrupt service routine. In the PC Network Program environment, where the application is only one of several tasks, the interrupt handler cannot assume that the application will be active when the interrupt occurs. Therefore, certain rules should be followed in servicing interrupts:

1. The routine should not mask other interrupts assuming that it is the only interrupt service routine loaded on the computer.
2. The stack should be used as little as possible.
3. The interrupt routine should be as short as possible to avoid interrupt overruns and lost interrupts.
4. No DOS calls should be made during the routine. Instead, the routine should leave a "mark on the wall" and exit. The main application should then process the interrupt when its task is active and such calls can be made.

Direct Disk/Diskette I/O

Applications running on a server configuration must not do direct disk or diskette accesses except under certain circumstances. If the applications make direct accesses to a disk or diskette, multiple concurrent updates to the media can occur. If updates do occur, the File Allocation Table (FAT), the directory, or the contents of a file will be destroyed. This can result in the loss of all information on a disk or diskette. Direct access is allowed only when the application can determine that the drive is not currently shared (NET SHARE command) or if the server component is paused (NET PAUSE command). See "0300H (INT 2AH) Check Direct I/O" on page C-6 of the PC Network Program User's Guide for more information.

Applications running on a server configuration must not do direct disk or diskette accesses except under certain circumstances.

Use of FCBs

Any application that does I/O over the network cannot use FCBs to:

- Construct its own open FCB.
- Save an FCB in a file and use it later in a different run.
- Change the FCB reserved areas.
- Close the FCB and then continue to use it as if it were still open.

An application wanting to ensure that data is written to the disk must either close and reopen the FCB or issue a DISK RESET (DOS function 0DH) to commit the data. These methods are slower than those listed above, but are the only methods supported on the network. Note that CLOSE ensures that the data is written to the server disk, while DISK RESET ensures only that the data is transmitted to the server.

Performance of Foreground Applications

When the PC Network Program is running, activity occurs in the background. This means that the computer is executing more than just the user program. As a result, the user program does not have control of the entire computer and may have inconsistent keyboard response. A noticeable delay may occur from the time that a key is pressed to the time that the user program receives the keyboard input and processes it. Developers should plan for this to occur when designing a user interface. Also, highly interactive programs, such as arcade-like games, may not function acceptably when the PC Network Program is running on the computer.

Use of the APPEND Command

The APPEND command was designed to make existing, non-networking applications usable in a networking environment. For example, if an application was coded to find its user-specific profiles or temporary work files in the same location as its program file(s), then multiple users would have to use the same profile, which would be marked READ ONLY by the ATTRIB command, and there would be a conflict when accessing the work files, since they would all have the same name and must be opened for WRITE access. Using the APPEND command, profiles and work files can be appended to the directory where the program resides and can therefore be user-specific, eliminating name conflicts.

In the new IBM PC Network Program and DOS Version 3.10 environment, applications should be coded so that the APPEND command is not necessary. Applications should avoid depending on the APPEND command to handle profile and workfile structures.

The APPEND command will direct a search for all files, regardless of the extension, in the appended directories only on the following function calls:

Code	Function Call
0FH	FCB File Open
3DH	Handle File Open
23H	Get File Size

Use of the Display Controller
An application should not reprogram the display controller directly. Instead, it should establish all mode information with BIOS calls (INT 10H). Failure to do this can cause the PC Network Program's network request key (Ctrl-Alt-Break) support to improperly restore an application screen image.

Any application that reprograms the display controller should record all changes in the display variables kept by the BIOS at segment 0040H. And any changes set into this area should be tested with the PC Network Program active to verify that the changes are effective. Changes can be verified by sending a message to the computer with the data area changed, viewing the message with the network request keys (Ctrl-Alt-Break), then exiting the menus. The display should properly restore all changes.

Controlling the Printer
There are three ways to send characters to a local printer: through the DOS INTerrupt 21H interface, through the INTerrupt 17H BIOS interface, and directly through the output ports. Only the first two methods are supported in the PC Network Program environment. The PC Network Program redirector function determines whether characters output through either interface are being sent to a printer that has been redirected to a network print server. If they are, the characters are buffered and sent to the server where the printer is located.

Print streams directed to a shared printer are spooled on the server computer and queued to be printed when the spool file is closed. The PC Network Program determines when to close the spool file as follows:

1. The application program on the remote computer terminates.
2. The user on the remote computer presses Ctrl-Alt-PrtSc.
3. The application opens or closes the printer through the DOS INTerrupt 21H interface, functions 3DH and 3EH respectively.
4. The application switches between the INTerrupt 21H and the INTerrupt 17H interfaces while doing the printing.
5. The process ID of the originator of the print characters changes (for example, printing switches from one application on the remote computer to another application on the same computer, or from one process to a subprocess within the same application).

Applications can use the NETBIOS interface to do application-to-application communication.

Method 1 above is too cumbersome for most applications. Method 2 requires user intervention. Methods 4 and 5 prevent concatenation of files by multiple applications on the same remote computer.

In accordance with the general guideline of using the highest system interface, application developers should code their applications to use method 3 above, using the DOS INTerrupt 21H functions, 3DH Open File, 3EH Close File, and 40H Write to a File or Device to print characters on a network printer. Opening and closing the printer through DOS delineates the print stream and causes the spool file to be closed and printed.

High-Level Language Use of DOS File Sharing and Byte Locking
High-level programming languages that do not offer the DOS file sharing and byte locking functions as language primitives can often make use of these functions through assembly language call routines. If these routines are coded correctly and are used as if they were part of the high-level language in the application, they can be easily replaced should these DOS functions make their way into the languages.

Application-to-Application Communication

Within the restrictions stated above, applications can use the NETBIOS interface to do application-to-application communication. Applications that intend for messages to be sent to them using the functions provided

by the PC Network Program (NET SEND and the Messenger) and wish to handle those messages within the application program should use the user post interface (see Appendix C, "Programming Interfaces" of the PC Network Program User's Guide for more information).

When the PC Network programs are loaded, the DOS redirection capability (NET USE and NET SHARE commands) should be used if at all possible. If this capability is not sufficient, application programs may use the network functions provided by NETBIOS (with the restrictions mentioned above) to perform network functions. To provide for enhanced function, the PC Network Program has the INTerrupt 2AH (Execute NCB or EXECNCB) interface to NETBIOS. See Appendix C, "Programming Interfaces" of the PC Network Program User's Guide for more information about the INTerrupt 2AH interface.

For future compatibility, applications should use the INTerrupt 2AH interface to NETBIOS. Using the interface requires that the application determine whether INT 2AH is installed (meaning that the PC Network Program is started). The application should fail to run if INT 2AH is not installed.

The PC Network Program provides a message presentation interface. For the receiver configuration, the interface presents a beep followed by the display of the message on the user's screen. If the user is in a full-screen application, the message overlays the information on the application screen. The interface for the messenger and server configurations is a beep, saving the user's display image, painting a box, and waiting for a signal to display the message. The PC Network Program provides an interface for an application program to intercept incoming messages before they are presented to the user. This allows the application to provide better notification of message reception (such as an indication in a status line). It also allows for better presentation of the message. For example, in a windowing system, the application could have a special window to show messages so that the window would not interfere with any other user activities. See Appendix C, "Programming Interfaces" of the PC Network Program User's Guide for more information about using this interface.

Transaction-Processing Applications

Applications that perform transaction processing require that messages be interchanged between two applications and also have defined data formats and data protocol. Transaction processing applications normally are executed at two computers by establishing sessions on each computer and then sending and receiving transactions in agreed-upon formats.

Transaction processing applications can use the following NETBIOS functions (INTerrupt 2AH) to interface with the PC Network Program:

- Cancel Network Request
- Get Network Adapter Status
- Add Name
- Delete Name
- Call
- Listen
- Hangup
- Send
- Chained Send
- Receive
- Receive Any
- Get Session Status
- Send Datagram
- Receive Datagram

See the PC Network Technical Reference for more information about NETBIOS functions.

Error Recovery and Problem Determination

Use of Common Error Messages

Application developers should be aware that in a local-area networking environment, users tend to use more applications than they do in a stand-alone environment. If every application uses different error messages to explain the same error to the user, the situation can get very confusing. To avoid confusion, we recommend that application developers keep the same text and even the same error number when they are handling and reporting network and DOS errors to the user through their application. This basic number and text can be expanded if necessary, but it makes the whole system more usable if users see the same message in the same situation no matter which application they are using, including the IBM PC Network Program itself.

Extended Error Support
Extended error support is available for DOS functions that are redirected across the network. Applications that use networking should be coded to check for network-specific errors that may require special action from the application or the user. For example, a network request can fail as a result of a timeout. The user or the application may want to retry the operation that failed. DOS Extended Error support provides explicit error codes for each error as well as an error class, action, and locus information. This information should be used in application error recovery procedures. See the DOS Technical Reference, Version 3.10 for more information.

Program Termination
Any application that issues NETBIOS requests must ensure that these requests have ended when the application terminates.This is not a problem if all NETBIOS requests are in "wait" mode, because DOS will terminate an application only at a DOS call. However, requests in "no wait" mode may remain outstanding at a DOS call, and if DOS terminates the application, the request is not terminated. This means that the request is still running for a program no longer in memory, resulting in unpredictable (and often fatal) behavior. To prevent this problem, the application must set up traps for Ctrl-Break, Ctrl-C, and DOS critical errors so that the application can gain control and issue NETBIOS Cancels for each outstanding request NCB before the DOS call is executed. See the DOS Technical Reference, Version 3.10 for more information about setting traps.

Physical Resource Sharing Applications
Some of the application's messages and operating instructions may not be meaningful in a network environment because physical resource sharing applications use a remote device, (i.e., a disk, directory, or printer) through the redirector, and instructions that reference a device may not be applicable if the device is a remote device.

The PC Network Program provides a message presentation interface.

The PC Network programming interfaces provide information about the status of remote and local devices. The application can use these interfaces to issue meaningful messages to the computer, whether the computer is using a local device or redirecting to a remote device. For example, if an error occurs because data is not available, a computer using a local drive may receive a message telling the user to insert the correct diskette. However, this message won't be applicable to a computer that is using a remote drive. In the case of a remote drive, the computer should receive and display a message stating that the appropriate diskette is not available.

The following interfaces are useful in developing physical resource sharing applications:

- DOS 3.10 Extensions

  Get Extended Error Code
  Create Unique File
  Create New File
  Get Machine Name
  Changeable IOCTL

  (See the DOS Technical Reference, Version 3.10 for more information.)
- Redirector Extensions

  Get Assign List Entry
  IOCTL Is Redirected Block
  (INTerrupt 21H)
  IOCTL Is Redirected Handle
  (INTerrupt 21H)
  Installation Check
  (INTerrupt 2AH)
  Installation Check
  (INTerrupt 2FH)

  (See Appendix C, "Programming Interfaces" of the PC Network Program User's Guide and the DOS Technical Reference, Version 3.10 for more information.
- EXEC System Call

  NET USE (PC Network Program Command)
  NET SHARE (PC Network Program Command)
  MODE (DOS Command)
  PERMIT (DOS Command)

  (See Chapter 7, "PC Network Program Command Reference" in the PC Network Program User's Guide for more information.)

IBM PC User Group Support Electronic Bulletin Board System

Steven Mahlum
IBM Corporation

To assist you in acquiring timely information about the IBM Personal Computer and IBM PC products, IBM's PC User Group Support Department offers not only the *Exchange* newsletter but also an electronic bulletin board system. Through these two media, PC users from user groups around the country can exchange technical ideas, insights, problems and solutions. We invite you to participate.

The electronic bulletin board system is located in Boca Raton, Florida and consists of several IBM Personal Computer XTs connected to a Personal Computer AT using the IBM PC Network. Each PC XT system has a 2400 baud modem to receive your calls. The bulletin board system database is contained on the Personal Computer AT and is available to multiple users over the PC Network. A specially written program coordinates all functions of the bulletin board system.

The bulletin board system database contains a wealth of information about the IBM Personal Computer family, including announcement material, hints and tips, users comments, and other valuable information. The information section of the database is divided into many topic areas. An index of all topic areas will help you locate the specific information you may want.

Another section of the bulletin board system database contains files that you may download and use on your PC system.

At times, we will use the bulletin board system to complement articles in *Exchange*. You will notice that an article in this issue, "Screen Image Files," refers to user-written programs that are not included in our newsletter. Rather than print the programs, we will make them available to you through the bulletin board system.

The electronic bulletin board system is now available to all user group members. It can be accessed by most communications programs (such as IBM's Personal Communications Manager, PC-TALK, QMODEM, PC-DIAL, etc.) and is available at all times.

To use the PC User Group Support Electronic Bulletin Board System:

- Set your program with the following parameters:
  (a) 300, 1200, or 2400 (use 2400 baud if your modem permits)
  or
  8 data bits, no parity, one stop bit (for binary data transmission).
  (b) Host echo (full duplex) can be changed to local echo (half duplex) for a noisy line).
- Dial 305-998-EBBS (305-998-3227)
- Enter your name as prompted.

You will be asked if you want to change the spelling of your name. By answering <N>o, you will be permitted a session to read information and download files.

Once your session has begun, you will be guided through a series of command menus. Each command in a menu may be selected by pressing one of the keys shown in brackets. From the main menu, you may choose the <I>nfo section or the <F>iles section. (Do not press enter after you press letters shown in brackets.)

The <I>nfo section contains nearly fifty topics, each with files that discuss pertinent information about that topic. For example, after pressing "I", you will be asked to enter a topic. If you are interested in information about the IBM Personal Computer AT, you would enter PCAT. The PCAT topic contains over 45 information files which you may browse through, selecting only those you wish to read.

The <F>iles section contains program files available for you to download. The two programs, PUSH.COM and POP.COM discussed in the "Screen Image Files" article are available in this section.

All menus include on-line help (pressing the H key displays explanations of the functions available in a menu). You can terminate the help text by pressing Ctrl-C.

We know you are always interested in making your PC a more productive tool. Through your participation, our newsletter and our bulletin board system will continue to provide you, the user group community, with timely and interesting information about the IBM Personal Computer family.

Customizing the Cursor

Bud Thurber
North East Indiana IBM PC Club

This article contains a simple program that allows you to modify the shape of the DOS cursor. It should work on any member of the IBM Personal Computer family.

We'll use DEBUG (which comes on your DOS Supplemental Programs diskette) to write the assembly language program shown below. The highlighted text shows what you enter from the keyboard. Anything to the left of the highlighted text is displayed on the screen by DEBUG, and anything to the right of the highlighted text is my comment. The XXXX in the address represents a hexadecimal address plugged in by DEBUG; the actual numbers that appear on your display are not important to us here. Place a diskette containing the DEBUG.COM file into drive A, and make A your default drive. Enter the following (emphasized text only):

```
A > debug cursor.com        This calls DEBUG to build
                              your file.
File not found              Don't worry about this message.
-a                          This will put DEBUG into
                              assembly
XXXX:0100 push ds           These three lines set
XXXX:0101 xor ax,ax           up the stack for the
XXXX:0103 push ax             RETF instruction.
XXXX:0104 mov ax, 0100      Tell ROM routine that you want
                              to set the cursor style.
XXXX:1007 mov cx,ssee       See instructions below for ssee.
XXXX:010A int 10            Call ROM routine to set
                              the cursor.
XXXX:010c retf              Tells DEBUG to return to DOS.
XXXX:010d                   Press the enter key to end
                              assembly.
-rcx                        Displays the CX register.
0000 0d                     This tells DEBUG how many
                              bytes to write for your file.
                              Since your program goes
                              from 0100 to 010c, it is 0d
                              bytes in length.
-w                          Tells DEBUG to write your
                              program to the new file
                              CURSOR.COM.
Writing 000D bytes          This is the computer's
                              response.
-q                          Quits DEBUG.
```

The value you enter for ssee in the above program depends on which display and adapter you use, and what style cursor you want. Below are descriptions of appropriate values for ssee for the IBM Monochrome, Color, and Enhanced Color Displays when used with their corresponding adapters.

IBM Monochrome Display and Adapter

Let's examine the structure of characters on the IBM Monochrome Display. Figure 1 shows an expanded DOS prompt and some sample values for ssee with the corresponding cursor each value creates.

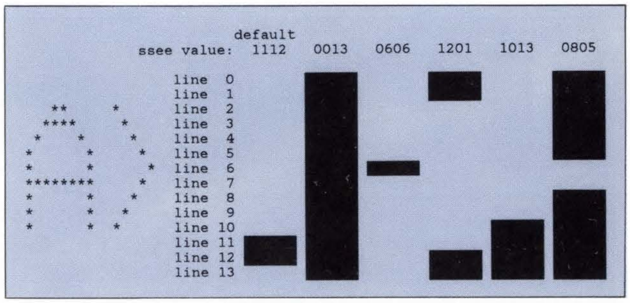


Figure 1. IBM Monochrome Display Cursor Samples

If you look closely at the DOS prompt "A > " you will see that it is made up of pixels as I've shown with the asterisks. The Monchrome Display uses 14 lines numbered 0 through 13. I think you can see the pattern. In the program above, the value of 'ss' should be the starting line of the cursor, and the value of 'ee' should be the ending line. An ssee value of 0000 will create a single overline cursor. The default cursor setting in DOS for a Monochrome Display is 1112, activating only lines 11 and 12.

If ss is greater than ee, the cursor will "wrap," creating a split cursor (1201 and 0805). Values for ssee with line numbers exceeding 13 often leave no visible cursor at all. If you cause a 'no cursor' condition, you might find it best to reboot DOS rather than go through DEBUG with no cursor.

Making the first s in ss a 6 (e.g. 6713), will cause the cursor to blink at half speed.

Once you've specified the new cursor style and written your file to disk, just enter CURSOR at the DOS prompt:

```
A>CURSOR
```

and your new cursor will appear. You will find that many programs (such as PC-Write and IBM Personal Editor) designate a specific type of cursor. When you exit these programs and come back to DOS, you will find that your cursor has reverted to the default value 1112. Re-enter the CURSOR command to make your custom cursor appear.

Once you find a cursor you like, put the CURSOR command in your AUTOEXEC.BAT file so you will get your custom cursor whenever you boot DOS.

IBM Color Display

The structure of characters on the IBM Color Display is different from the Monochrome Display. Figure 2 shows an expanded DOS prompt and some sample values for ssee with the corresponding cursor each value creates.

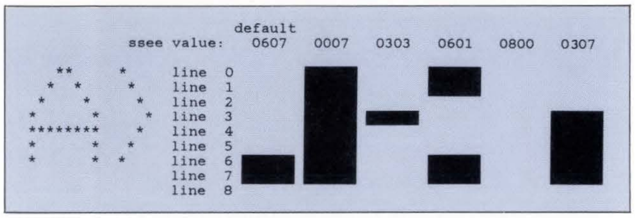


Figure 2. IBM Color Display Cursor Samples

The IBM Color Display uses nine lines numbered 0 through 8. The default cursor setting in DOS for the Color Display is 0607, activating only lines 6 and 7. In the program above, the value of 'ss' should be the starting line of the cursor, and the value of 'ee' should be the ending line. If ss is greater than ee, the cursor will "wrap," creating a split cursor (0601).

Making the first s in ss a 6 will cause the cursor to blink at half speed.

Note that line 8 never displays, and 0800 leaves you with no visible cursor. Values for ssee with a specified line number exceeding 7 often leave no visible cursor. However, since this program does not affect the five-line cursor that DOS displays in insert mode, you can press the Ins key and go through DEBUG with the thicker cursor.

As with the Monochrome Display, once you've specified the new cursor style and written your file to disk, enter the command CURSOR at the DOS prompt:

```
A>CURSOR
```

to make your new cursor appear. Also, when a program designates a specific type of cursor, and you exit it and return to DOS, your cursor reverts to the default value 0607. Re-enter the CURSOR command to make your custom cursor appear.

Once you find a cursor you like, put the CURSOR command in your AUTOEXEC.BAT file so you will get your custom cursor whenever you boot DOS.

IBM Enhanced Color Display

This program also applies to the IBM Enhanced Color Display, but the values for ssee differ, depending on whether you are running in the enhanced mode or not. If you have the IBM Enhanced Graphics Adapter and the Enhanced Color Display, you may want to experiment with different values for ssee to see what type of cursor you get. Once you find an ssee value that produces the type of cursor you want, put the CURSOR command in your AUTOEXEC.BAT file.

Altering the Custom Cursor

If you want to make a different cursor, enter the following (highlighted text only) to change only the ssee instruction:

A > debug cursor.com	This calls DEBUG to build your file. Notice no error message this time.
-a 107	This will put DEBUG into assembly at the single instruction point.
XXXX:0107 mov cx,ssee	See instructions above for ssee.
XXXX:010d	Press the enter key to end assembly.
-w	This will rewrite your program to the file CURSOR.COM.
Writing 000D bytes	This is the computer's response
-q	Quit DEBUG.

Editor's note: For more information on customizing the cursor in your own BASIC programs, see John Herzfeld's article "Shaping Your Cursor in DOS" which appeared in the April 1984 diskette issue of Exchange. *His article and his programs also are available on the IBM PC User Group Support Electronic Bulletin Board System. The article is in the <I>nfo section under the subject heading "DOS" (message #8), and his program files CRL.BAS, CRS.BAS, CRL.COM, and CRS.COM are available for downloading from the <F>iles section. For information about accessing our electronic bulletin board system and downloading these files, see the article "IBM PC User Group Support Electronic Bulletin Board System" in this issue.*

New Products

IBM PC BASIC Compiler 2.00

IBM Personal Computer BASIC Compiler Version 2.00 is a high-function BASIC language compiler designed to run under TopView and in the IBM PC Network environment. Based on the IBM PC BASIC Compiler Version 1.00, Version 2.00 contains many enhancements and new features, while maintaining upward compatibility with Version 1.00.

Version 2.00 complements the functions contained in the latest version of IBM BASIC Interpreter distributed with DOS 3.00 and 3.10, making IBM PC BASIC Compiler 2.00 an excellent programming tool to use with the BASIC Interpreter. You can create and debug programs using the BASIC Interpreter, then compile the programs to increase execution speed, increase program security (the BASIC language source code is not required to execute the application once it has been compiled), and increase program access to all DOS supported random access memory.

In addition to the features of BASIC Compiler 1.00, Version 2.00 offers the following enhancements:

- Ability to run under TopView and in the IBM PC Networking environment.
- Improved program control structures, including named subroutines, user-defined multi-line functions, separately compiled BASIC subprograms, and the ability to branch to alphanumeric labels.
- Capacity to compile larger programs because the code space has been expanded to 64KB and is now separated from the data space (also expanded to 64KB); the symbol table has also been expanded.
- Larger numeric dynamic arrays, restricted only by available memory and maximum index (maximum index for any dimension of a numeric array is 32767). Also, the arrays can be separated from code and data space.
- Expanded graphics capabilities which include support for all graphics features of BASIC Interpreter 3.00 with significant changes to the VIEW, WINDOW, PMAP, LINE, DRAW, POINT, and PAINT statements.
- Graphics statements that use line clipping instead of wraparound.
- Enhanced event trapping: on timer, on play and on key.
- Filespec syntax allowing specification of path for device or file; redirection of standard input and output; lock/unlock file features; ISAM file support.
- Support for the advanced features of PCjr BASIC, including play - multi-voice; play - volume control; noise; enhanced screen statement; enhanced clear statement; pcopy; and additional screen modes.
- Expanded use of DOS functions. Changes affect the DOS SHELL, IOCTL, IOCTL$, ERDEV, ERDEV$, MKDIR, RMDIR, CHDIR, and ENVIRON$ commands.
- Ability to return termination codes when the compiler exits, allowing termination codes to be tested using the DOS IF batch subcommand.
- File access control to the OPEN statement. Up to five levels of nested $INCLUDE files.
- Three new metacommands: $DYNAMIC—allows dynamic allocation for array space; $MODULE—allows name change of an internal module passed to linker; $STATIC—allows static allocation for array space.
- New statements: REDIM—changes space allocated to a dynamic array; STATIC—designates variables as local to subprogram or multi-line function; SUB, SUB/END, SUB/EXIT—designates start and end of subprogram; DEF FN, END DEF, EXIR DEF—designates start and end of a multi-line function.
- New functions: COMMAND$—returns parameters from command line to invoke current program; LBOUND—returns value of lowest subscript available for any array, depending on the option base statement setting; UBOUND—returns value of largest subscript for any array.

IBM PC BASIC Compiler 2.00 comes with two volumes, a Fundamentals Manual and a Reference Manual; a Quick Reference Card; two diskettes, one containing IBM PC BASIC Compiler 2.00 library modules, and one containing the Indexed Sequential Access Method (ISAM).

IBM BASIC Compiler 2.00 runs on all members of the IBM Personal Computer family. It requires DOS 2.10 or higher; 128KB of random access memory; one double-sided diskette drive; and one of the following displays with the appropriate adapter: IBM PC Color Display, IBM PCjr Color Display, IBM Monochrome Display, or IBM Enhanced Graphics Display (supported in compatibility mode only). Additional random access memory, additional storage capacity (two diskette drives or hard disk), an 80 column display, and a printer are highly recommended.

Editor's Comments

Active User Group Participation

In the PC User Group Support Department, one of our main missions is to facilitate the exchange of technical information among IBM PC user group members. To make this happen, we distribute this newsletter at no charge to all *active* members of PC user groups that have registered with us for support. We stress the word active because it is only through the participation and contribution of members that a user group becomes a viable entity.

The hard work of various groups' members is evident in user group newsletters and in *Exchange*. User group members who research new ideas, add to other people's ideas, develop applications, and record their insights in articles in their own group newsletters are contributing greatly toward the exchange of IBM PC information. These authors also benefit from writing down their ideas because it helps them define their ideas more clearly. Finally, after ideas are published, the authors often receive well-deserved recognition for their work. We in User Group Support want to help circulate useful user-written articles by publishing them in *Exchange* and distributing them to PC user groups throughout the United States.

I'd like to encourage you to put your thoughts into print. Many potential authors have excellent ideas but are uneasy about writing them down and developing an article. Our staff looks for material in user group newsletters, selects helpful articles, requests permission to publish them, and revises and edits the articles. As one contributing author said, "I don't mind if IBM makes me look good, especially at their expense." Hopefully this will encourage you to write down your ideas and to publish them in your group newsletter. After all, we want to disseminate *your* ideas.

We thank all of you who have contributed to the exchange of IBM PC information, and we look forward to your increased support and participation.

Bernard H. Penney
User Group Editor

Copyrights, Trademarks and Service Marks

ColorPaint by Marek and Rafal Krepec Incorporated.

ColorPlus is a trademark of Plantronics Corporation.

CompuServe is a trademark of CompuServe, Incorporated.

CP/M is a registered trademark of Digital Research, Incorporated.

CP/M-86 is a trademark of Digital Research, Incorporated.

Data Encoder and its associated documentation are under the U.S. Department of State Munitions list, Category XIII(b) and, as such, must be licensed by the U.S. Department of State prior to export from the United States.

DIF is a trademark of Software Arts, Incorporated.

Dow Jones News/Retrieval Service is a registered trademark and Dow Jones is a trademark of Dow Jones & Company, Incorporated.

EasyWriter is a trademark of Information Unlimited Software, Incorporated.

Electric Poet is a registered trademark of Control Color Corporation.

Fact Track is a trademark of Science Research Associates, Incorporated.

HomeWord is a trademark of Sierra On-Line, Incorporated.

IBM is a registered trademark of International Business Machines Corp.

INTERACTIVE and IS/5 are trademarks of Interactive Systems Corporation.

Jumpman is a trademark of EPYX, Incorporated.

King's Quest is a trademark of Sierra On-Line, Incorporated.

Logo is a trademark of Logo Computer Systems Incorporated.

Lotus and 1-2-3 are trademarks of Lotus Development Corporation.

Managing Your Money is a trademark of MECA (TM).

MECA is a trademark of Micro Education Corporation of America, Incorporated.

Microsoft and the Microsoft logo are registered trademarks of Microsoft Corporation.

Motown is a registered trademark of Motown Records Corporation.

Multiplan is a U.S. trademark of Microsoft Corporation.

NEC is a trademark of Nippon Electric Co., Ltd.

PCjr is a trademark of International Business Machines Corp.

PC Mouse is a trademark of Metagraphics/Mouse Systems.

Peachtext is a trademark of Peachtree Software Incorporated, an MSA company.

Personal Computer AT is a trademark of International Business Machines Corp.

Personal Computer XT is a trademark of International Business Machines Corp.

pfs: is a registered trademark of Software Publishing Corporation.

PlannerCalc is a trademark of Comshare.

REALCOLOR is a trademark of Micro Developed Systems, Inc.

SHAMUS is a trademark of SynSoft(TM).

SMARTMODEM is a trademark of Hayes MicroComputer Products, Inc.

Synonym information in PCWriter and Word Proof is based on the American Heritage Dictionary Data Base, Roget's II, The New Thesaurus, owned by Houghton Mifflin Company and used with permission. Copyright 1982 by Houghton Mifflin Company.

Teletype is a trademark of Teletype Corporation.

The Learning Company reserves all rights in the Rocky, Bumble, Juggles and Gertrude characters and their names as trademarks under copyright law. Rocky's Boots, Bumble Games, Bumble Plot, Juggles' Butterfly, Gertrude's Puzzles, Gertrude's Secrets and The Learning Company are trademarks of The Learning Company.

THE SOURCE is a service mark of Source Telecomputing Corporation, a subsidiary of The Reader's Digest Association, Incorporated.

Time Manager is a trademark of The Image Producers, Incorporated.

TopView is a trademark of International Business Machines Corp.

UCSD, UCSD p-System and UCSD Pascal are trademarks of the Regents of the University of California.

UNIX is a trademark of AT&T Bell Laboratories.

VisiCalc is a trademark of VisiCorp.

Visi On is a trademark of VisiCorp.

WD212-X is a trademark of Wolfdata, Inc.

Word is a U.S. trademark of Microsoft Corporation.

WordStar is a trademark of MicroPro International Corporation.

XENIX is a trademark of Microsoft Corporation.

Z-80 is a registered trademark of Zilog.

"What finally evolved was a POP. COM program that restores the original screen image and allows you to modify almost everything. (page 10)

"The frequency and manner in which you use the Update Video Display call can have a dramatic overall impact on your application's performance. (page 13)

"Determining the best method for using the Update Video Display call depends on the function of the application and the amount of user interaction. (page 14)

"There is probably little or no value in specifying more than 10–20 buffers. (page 18)

"If you have plenty of memory and have not tried a virtual (RAM) drive, try one. You will be amazed how much more quickly things can happen. (page 19)

"XENIX is an outstanding bargain, especially if you consider the per-user cost for multiple users. (page 21)

"Applications running on a server configuration must not do direct disk or diskette accesses except under certain circumstances. (page 28)

"Applications can use the NETBIOS interface to do application-to-application communication. (page 29)

"The PC Network Program provides a message presentation interface. (page 31)