Volume 2, Number 6-1 March 1985

# IBM Personal Computer Seminar Proceedings

**The Publication for Independent Developers of Products for IBM Personal Computers**

**Published by International Business Machines Corporation**
**Entry Systems Division**

IBM®

Changes are made periodically to the information herein; any such changes will be reported in subsequent Proceedings.

It is possible that this material may contain reference to, or information about IBM products (machines and programs), 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 products, programming or services in your country.

IBM believes the statements contained herein are accurate as of the date of publication of this document. However, IBM makes no warranty of any kind with respect to the accuracy or adequacy of the contents hereof.

This publication could contain technical inaccuracies or typographical errors. Also, illustrations contained herein may show prototype equipment. Your system configuration may differ slightly. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever.

All specifications are subject to change without notice.



# Contents

# Introduction and Welcome

These are the Proceedings of the IBM Personal Computer Seminar, designed for independent developers of products for IBM Personal Computers. The purpose of these Proceedings is to aid you in your development efforts by providing relevant information about new product announcements and enhancements to existing products. This issue is prepared in conjunction with this seminar. The Proceedings of future seminars for the IBM Personal Computers also will be published and will cover topics presented at those seminars.

Throughout these Proceedings, the term IBM Personal Computer and the term family of IBM Personal Computers address the IBM Personal Computer, the IBM Personal Computer XT, the IBM PC*jr*, the IBM Portable Personal Computer, and the IBM Personal Computer AT.

## Purpose

What is our purpose in issuing a publication such as this? It is quite simple.

The IBM Personal Computer family is a resounding success. We've had a lot of help in achieving this success, and much of it came from the independent developers.

As you proceed with your development, do you at times wish for some bit of information or direction which would make the job easier? Information which IBM can provide? This is the type of information we want to make available to you.

Since we want to be assured of giving you the information you need, we ask you to complete the questionnaire which appears at the end of these Proceedings. Your response to this questionnaire will be taken into account in preparing the content of future issues, as well as the content of seminars we will present at microcomputer industry trade shows.

## Topics

The following list gives a general indication of the topics we plan to cover in future seminars and include in the IBM Personal Computer Seminar Proceedings:

- Information exchange forum — letters to the editor format
- Development tools — languages, database offerings
- Compatibility issues
- New devices — capacities and speeds
- System capacities — disk and memory
- Enhancements in maintenance releases
- Tips and techniques
- New system software
- Hardware design parameters
- Tips on organizing and writing documents for clear and easy reading
- Changes to terms and conditions

# TopView

## Introduction

TopView provides a new operating environment that allows users to take full advantage of the power of their IBM Personal Computer. TopView provides a multitasking and windowing environment for PC-DOS applications. All of TopView's facilities are available to applications that are specifically designed to run with TopView, i.e., programmed using the TopView applications programming interface (API). Many existing applications also run in the TopView environment. Existing applications, in general, may not be able to use every TopView feature. However, the ability to switch from one application to another and the copy feature is always available.

TopView provides a single-user operating environment and runs with DOS Version 2.00 or later. With DOS Versions 3.00 and 3.10, TopView supports only the function calls (INT 21) available in DOS Versions 2.00 and 2.10.

## Operating Environment and Programmer's ToolKit

The TopView product consists of two separately priced packages. The first product, called TopView, is the TopView operating environment. The second product, the TopView Programmer's ToolKit, contains the system interfaces (TopView API) and programming tools for developing applications that take full advantage of TopView features. The documentation for the TopView Programmer's ToolKit also includes compatibility guidelines for existing applications.

### Major Features of the Operating Environment

- Multitasking — In the TopView environment, a single user can manage as many applications as will fit into memory. The applications may be different or the user may run multiple copies of some applications.

  At any given time, the user interacts with one application running in the foreground. Within the window of the foreground application you can do things such as enter data, select items, window, and edit. You accomplish some of these with the help of a pointing device, controlled by either the keyboard or a mouse.

  TopView supports the execution of more than one application or task at the same time. Applications that follow certain guidelines may be running simultaneously in the background and performing such activities as computations, printing, and communications.

  You can switch between foreground and background applications by either using a pop-up menu or pressing keyboard or mouse buttons to cycle from one application to the next.

- Window Oriented — A window is a rectangular area that one application uses on the screen. A window can occupy the entire screen or a portion of the screen as small as a single character. Windowing is especially useful for viewing several different applications on the same display at the same time. The window of a foreground application has a double-line border. If the application is using the entire screen, the window border is not seen. The windows of background applications have a single-line border. Windows can overlay other windows without destroying the contents of the overlaid windows.

  TopView can window applications if they do not write directly to the video buffer and if they follow some general compatibility guidelines. TopView windowing functions include sizing a window — making a window larger or smaller; moving a window — changing the location of a window on the screen (if the window is less than full screen); scrolling data through a window — allowing data outside of a window that has been reduced in size to be viewed by a user; zooming a window — changing a window that has been reduced in size back to its original size; and unzooming a window — reversing the zoom process to instantly restore the size of a zoomed window to its previous, reduced size. Any window can also be hidden — removed from view on the screen.

- Menu Driven — A menu is a list of items, contained in a window, from which you can make selections. TopView uses pop-up menus. After a menu pops up on the screen, you can select from the menu. After you make your selection, the pop-up menu disappears. Pop-up menus allow applications to use all parts of the screen, if required.

Pop-up menus provide access to all the functions of TopView. Most of the pop-up menus are under user control — you decide whether to make a menu appear. Some pop-up menus are controlled by TopView and appear when TopView needs additional information or has information for you.

When a pop-up menu appears, its window occupies only a portion of the screen, temporarily overlaying but not destroying the previous contents of that portion of the screen. When a pop-up menu disappears, the previous contents, which had been overlaid by the pop-up menu window, once again appear.

- Interactive Tutorial — An interactive tutorial program is provided with TopView. The tutorial lets the user learn about many of TopView's functions by trying them in a controlled environment.

- Copy, Cut and Paste — TopView provides the user with the capability to transfer data between applications or within an application. TopView provides applications with the facilities for performing data movement operations where they are applicable. The copy function copies up to a full screen of text that has been marked by the user into a buffer where the data is kept until it is used as input for a paste operation. The copy function is available for all applications in text mode. The cut function is the same as the copy function except that the marked text block is deleted from the window (and the application) by the application. The paste function passes the data in the buffer (from the last cut or copy function) to the receiving application so it can insert the data at the location marked by the user. The cut and paste functions are not available for all applications since all applications do not support the appropriate input or block delete functions. A list of existing IBM Personal Computer applications supported is provided in the *TopView Application Guide*. The user performs the cut, copy and paste operations by using the TopView pop-up menus.

- Pointing Device Support — TopView uses a pointing device to control the pointer on the screen. A typical pointing device is a 'mouse' or the keyboard. Support of specific 'mouse' devices is documented later in this publication. This TopView facility greatly enhances the movement of the selection pointer on TopView menus and allows the user additional flexibility in moving the pointer around the display screen. The pointer is used to point to an option on a TopView menu created using the TopView facilities (e.g., a list of functions that can be selected). The mouse pointing device is optional.

- DOS Services — Many Disk Operating System (DOS) functions can be executed in the TopView environment. Once this feature has been selected, pop-up menus aid you in selecting some of the more frequently used DOS functions such as copy, print, erase, and rename. Directories also can be sorted by name, extension, size and date/time to facilitate more flexible access to directory data. You can execute many DOS functions without leaving TopView and background applications can continue to run.

- Graphics Applications — TopView supports graphics applications by giving them control of the full screen when the graphics application is in the foreground. Both medium resolution color (320 x 200 four color pixels) and high 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. Several graphics applications can be started at the same time under TopView, but they are not allowed to run in the background. However, if a graphics application also uses text mode, it may run in the background while it is in text mode. Windowing functions are not supported for graphics applications when they are in graphics mode.

- Productivity Enhancement — TopView provides the user with an operating environment that improves productivity by:

  - Providing user control of the work flow on the PC so that interruptions are easier for the user to handle.

  - Reducing program start/stop time.

  - Overlapping program execution time to get more total throughput (work) when running multiple applications.

- Compatibility — TopView supports many existing IBM and non-IBM programs. Many of these programs are listed in the *TopView Application Guide*, available to all TopView licensees. In addition, a list will be distributed (and updated) to authorized IBM dealers and IBM marketing representatives.

The ability to use TopView features depends on how well the program interacts with TopView. Existing applications, in general, may not be able to use every TopView feature. However, the following features are always available:

- Program switching.
- Use of either the keyboard or a mouse pointing device to control and position the TopView pointer.
- Copy (in text mode).

## Terminology

The following terminology is associated with TopView:

- The **cursor** is the blinking underscore character that appears inside the window of the foreground application. The cursor usually marks the location where you enter data.
- The **pointer** is the solid square locating character used by TopView itself. This character is an arrow in graphics mode. You use the pointer to perform TopView functions such as selecting from a pop-up menu, marking areas to be sized or moved, and marking information to be transferred.
- A **pointing device** is any device that can be used to control the pointer on the screen. All pointing devices in TopView are interrupt driven; they require interrupt handlers to tell the system what they are doing. TopView handles pointing device interrupts using programs called pointing device drivers. In addition to handling interrupts, each pointing device driver also translates the unique characteristics of its pointing device into a standard format usable by TopView.

  TopView provides pointing device drivers for the keyboard and for the devices listed below.

- A **mouse** is one device that controls movement of the pointer. Moving the mouse on a flat surface such as a desk moves the pointer in the same direction on the screen. If you move the mouse away from you, the pointer moves up; toward you, the pointer moves down; similarly for left and right. You can move a mouse, and therefore the pointer, diagonally as well.

  TopView supports four specific mouse devices at date of announce:

  - PC Mouse, Part Number 900120-214, by Mouse Systems, Inc.
  - Microsoft Mouse for IBM Personal Computers, Model Number 037-099 (Parallel Interface)
  - Microsoft Mouse for IBM Personal Computers, Model Number 039-099 (Serial Interface)
  - Visi On Mouse, Part Number 69910-1011

  IBM has tested the use of these devices with TopView as of the announcement date of TopView. However, IBM does not endorse or recommend one non-IBM product over another and does not warrant these devices in any way. Other pointing devices can be used if a device driver exists or is written for the device according to the guidelines described in the TopView Programmer's ToolKit Reference Book. Your dealer may be able to identify additional pointing devices that work with TopView as they become available.

  The use of a mouse as a pointing device is optional. When using the keyboard as the TopView pointing device, the buttons on a mouse are replaced by the following keys: Home (Button 1), PgUp (Button 2), and Alt (Button 3). The arrow keys move the TopView pointer in single space increments. Holding down the Ctrl key while pressing an arrow key moves the TopView pointer in larger, user adjustable increments. The Ctrl key is also used as a toggle to allow the arrow keys to control either the TopView pointer or the application cursor.

## Menus

The first menu after the Welcome Window is the Start a Program menu. This menu lists the application programs you can run in your TopView environment. You control the choices that appear in this menu.

When you add the name of a program to the Start a Program menu, you can then run that application in the TopView environment. The Add a Program process provides TopView with certain information about your application. You can then start the application by selecting its title from the Start a Program menu. You can also delete the name of a program from the Start a Program menu, thereby removing it from your TopView environment.

You can display the main TopView menu one of three ways: pressing the Alt key on the keyboard, buttons 1 and 2 together on a two-button mouse, or button 3 on a three-button mouse.

The main menu, called the TopView menu, provides the following functions:

- **Scroll:** If an application window has been reduced in size, you can use the Scroll function to display portions of the application's screen that are outside the window. By invoking Scroll and moving the pointer, you can move the text into view. If the window contains an input cursor, TopView always makes sure that the cursor stays inside the window so that you always see what you are typing. For applications using the TopView programming interface, you can scroll within the full-screen. Many existing applications may provide a second level of scrolling; e.g., next page.

- **Window:** When you select the window function, another pop-up menu called the Window menu appears. Inside the Window menu are these choices:

  - **Move:** You can use the Move function to position a window to any convenient place on the screen.

  - **Size:** You can increase or decrease the size of a window. The normal size window for an application is the full screen, but the window can be reduced to any usable size, provided the application allows it through use of the Size function.

  - **Zoom:** A window that has been reduced in size can be expanded to the full screen by using the Zoom function.

  - **Unzoom:** After you have zoomed a window, it can be restored to its reduced size by using the Unzoom function.

  - **Hide:** You can remove a window from the screen by using the Hide function. When you hide a window, you do *not* stop the processing being done by the application in that window. To bring back the window you have hidden, you must bring up the TopView menu and switch to the application in the hidden window. The window then returns to the screen as the foreground application.

- **Scissors:** This function lets you transfer information between application windows. The Scissors menu includes these features:

  - **Copy:** With this function, you can copy information from the foreground window to a storage buffer maintained by TopView, while retaining the information in the foreground application.

  - **Cut:** This function is similar to Copy, except that the original information in the foreground application is deleted.

  - **Paste:** After a Copy or Cut has placed some information into the storage buffer, Paste brings that information out of the buffer and into the current foreground application.

  To use cut or paste in an existing application, that application must have a Filter Table (described later).

- **Help:** Help screens exist for many TopView functions. To use Help, bring up the TopView menu and select Help. TopView has a comprehensive set of help screens. When you select Help, TopView displays the particular help screen you need. You can then bring up additional help screens, or quit from the help screens and return to your application.

- **Suspend:** This function temporarily halts the processing of an application and removes that application's window from the screen. To resume processing, bring up the TopView menu and switch to the suspended application, which then becomes the foreground application.

- **Quit:** The Quit function lets you end an application. When you quit an application, that application's name is removed from the Switch menu. To restart the application, display the Start a Program menu and select that application. The Quit option may not be available from the TopView menu for many existing applications. It is intended primarily for applications written to the TopView interface. In general, it is recommended that you use the usual method to end an existing application. This helps to insure that all of the application's data is saved correctly.

- **Switch:** The Switch function allows you to move from program to program. The Switch menu contains a list of programs currently available in the system. From this list you can also determine if a program is suspended or hidden from view. A program selected from this list immediately begins running as the foreground application.

- **Programs:** This function returns you to the Start a Program menu. From the Start a Program menu you can load a program into storage and start that program running as the foreground application. Using this feature you also can add information about new programs so they can be loaded and started when you desire. Program information about applications also can be deleted or modified.

- **Exit:** With the Exit function, you leave TopView and return control to DOS. Unless a special

TopView option is used when you start TopView, you must first Quit all applications before you can Exit TopView.

### Managing Applications

TopView manages the applications that run in its environment. To do this, TopView requires information about the unique operating characteristics of an application prior to running it.

- **Program Information:** To run an application in the TopView environment, TopView needs certain information about the application: the name, where to find the program, the default drive and directory, parameters, memory requirements, and behavior characteristics.

  TopView keeps this information in a Consolidated Program Information File. Each application that runs in the TopView environment has its own record in the Consolidated Program Information File. The TopView diskette includes program information for most of the existing IBM software products that run with TopView. An application can have a small Program Information File included on its diskette. An individual Program Information File has an extension of .PIF and is identical in structure to one record in the Consolidated Program Information file maintained in the TopView directory. When the application is added to TopView, the PIF information is appended to the Consolidated Information file.

- **Filter Tables:** To use the Scissors function to transfer data from one application window to another or within a window, a filter table, in conjunction with the TopView Filter Program is required.

  To do this, you must first use either the Copy or Cut option to mark the data to be transferred. TopView then transfers the data from the sending window into an internal buffer. You then select the Paste option, move the pointer into the receiving window, and position it where you want to place the data. TopView then takes the character codes it finds in the internal buffer and writes them to the receiving application's keyboard buffer. By creating these keystrokes, TopView enters data into the receiving application just as if the data is entered through the keyboard.

  You can use the Scissors functions in an existing application only if the application's Filter Table supports them.

  All Filter Tables have a common format that the TopView Filter Program can read.

  Included on the TopView diskette are Filter Tables for some existing IBM software products. If you intend to add applications other than these to your TopView environment and you want your applications to perform functions that require filtering, a Filter Table must be available for each application. (Guidelines for writing application Filter Tables are given in the TopView Programmer's ToolKit.)

### Adding Your Applications

Before you can run an application under TopView, you must add the application's name to the Start a Program menu. You do this by bringing up the TopView menu and selecting Add a Program. If your application is an IBM application with program information included on the TopView diskette, or if your application includes a Program Information File, adding your program to TopView is easy. You need to provide TopView with the location (drive and directory) of your application and its associated Program Information File (if available). Let's assume that program information is not available for an application you wish to run in the TopView environment. TopView will not find the program information for your application, and you will be prompted for the minimum information required. The remainder of the information is defaulted by TopView.

You can review the contents of the program information that TopView creates for your new application and make changes by using the **Change Program Information** function.

### Online Tutorial

After you start TopView you see a welcome screen. This screen offers you the choice of taking the online tutorial that is supplied on a separate diskette (part of the TopView package).

You should take this tutorial before using TopView. The tutorial covers these subjects:

- Getting help when you need it
- Adding applications to your TopView environment
- Starting your applications
- Switching among applications that have been started
- Windowing
- Scissors
- Ending an application
- Using DOS Services

### Hardware Requirements

Following is a list of the computer hardware supported in the TopView environment:

- TopView runs on the IBM Personal Computer AT, The IBM Personal Computer XT, the IBM Personal Computer, the IBM Portable Personal Computer, and the IBM 3270 PC (Models 4 and 6) with the 3270 PC Control Program Version 1.2.
- TopView supports one of the following:
  - An IBM Monochrome Display and Parallel Printer Adapter
  - An IBM compatible 80-column color or black and white display with the IBM Color/Graphics Display Adapter
  - An IBM Enhanced Color Display with an IBM Enhanced Graphics Adapter that is configured for Enhanced Color (in Normal Color mode)
  - An IBM Professional Graphics Display with an IBM Professional Graphics Controller in emulator mode

    Note: When using the IBM Professional Graphics Adapter, TopView must be configured for the keyboard or the parallel mouse pointing device.
- A minimum of 256 KB of Random Access Memory. A larger amount of memory is necessary to execute several large programs at the same time, to execute programs with several windows, or to copy/cut/paste a large block of text.
- Two double-sided, double-density diskette drives. The second drive may be a fixed disk.

The above list constitutes the basic hardware requirements for TopView. However, the following hardware is recommended for optimal performance and usability:

- An IBM Personal Computer AT or XT with a fixed disk drive and one double-sided diskette drive.
- At least 512 KB of memory.
- An IBM Graphics Printer or other compatible printer.

## TopView Programmer's ToolKit

The TopView Programmer's ToolKit provides the system interfaces, programming tools, and guidelines for developing applications that take advantage of the TopView operating environment.

The interfaces and tools are intended to make it easier to write programs that run with TopView. These tools and interfaces help you produce system code to use in your application program. For example, if you want your program to do such things as windowing, multitasking, and data transfer, you can use the interfaces and the guidelines in the Programmer's ToolKit.

The Programmer's ToolKit includes the Window Design Aid for developing windows, menus, help windows, error windows, and forms to be used within an application.

The ToolKit also includes a language interface to the IBM Macro Assembler, a sample high-level language interface for the IBM Pascal Compiler; utilities for merging panel files and converting panels to linkable object modules; and the interface to use when writing a pointing device driver.

The documentation included with the Programmer's ToolKit explains the TopView programming environment, how to make use of the TopView programming interface, and how existing applications can best take advantage of the TopView environment.

### Window Design Aid

The Window Design Aid is an interactive tool that allows you to design, construct, store, and maintain panels in the TopView environment.

A panel is a stream of data that creates a window. All the information about your windows, such as display attributes, text, fields, and data format, is kept in panel files. When your foreground application program displays a window, the program actually calls for a panel file and gives that file to TopView. TopView then builds the window using the information in the panel file.

The Window Design Aid consists of a full-screen editor and several menus for creating panels. Using these menus you can select three kinds of functions. The Edit functions let you enter, move, and copy text, lines, and boxes. The Panel functions let you select the window border, title, placement, and display attributes (color, reverse video, etc.). The Field functions let you define fields within your panels.

### Edit Menu

When the Window Design Aid comes up on the screen, you see the Edit Menu. From this menu you can select and perform several edit functions for creating panels: Draw, Copy, Move, Auto Copy, Erase, Fill Text, and Fill Attribute. (Each function is explained below.) The Edit Menu also gives you access to the Panel Menu and the Field Menu.

The edit functions are:

- **Draw:** This option lets you create lines and boxes in a marked area of your panel. You can draw single lines or double lines. If you draw boxes, they can have either single-line or double-line borders. You can draw through text or around text, or you can draw lines or boxes and add text later.

- **Copy:** This option lets you duplicate a desired portion of text. You can duplicate in as many places on the screen as you desire. You must first mark the area around the text you want to duplicate. The text you copy remains in its original place.

- **Move:** This option lets you relocate a desired portion of text. You must first mark the text you want to relocate, then select the Move option and move the text to its new location.

- **Auto Copy:** This option lets you automatically duplicate a marked area wherever you move the pointer in the window. You can use Auto Copy to duplicate characters, lines, or boxes.

- **Erase:** This option lets you erase a marked area of text.

- **Fill Text:** This option lets you fill a marked area with any character you specify.

- **Fill Attribute:** This option lets you fill a marked area with any display attribute you specify. Depending on the type of attributes your application is using, you can choose from monochrome or color attributes. For example, if your program is intended to run on a color monitor, you can specify both foreground and background color attributes. You can also select a logical attribute value that will produce the desired effect whether the application is used with a color or a monochrome monitor.

### Panel Menu

The Panel Menu offers you a choice of reading or writing a panel, an overlay, or a template (explained later). It also gives you access to the Panel Options menu, where you can define your window's features and set its attributes.

The Panel Options Menu lets you specify your window's border, title, position (top or bottom), attributes, and display type.

The panel options are:

- **Window Border:** Provides a box around the window. You can turn the window border on or off and specify the border attributes.

- **Window Title:** If a window border appears, the Window Title is shown imbedded in the upper left corner of the border.

- **Window Position:** This determines how your window appears when your application is started. If you select Window on Top, the window becomes the topmost visible window. If you select Window Hidden, the window is initially hidden.

- **Window Attributes:** Define the display attributes for your window. The attribute options let you select either Logical Attributes or Physical Attributes.

  A physical attribute is passed unchanged to the display adapter currently in use. If you change from monochrome to color displays, or vice-versa, the physical attribute you specified for one display may be incorrect for the other display. Consequently, the concept of logical attributes was developed.

  A window in TopView can use 16 logical attributes. Each logical attribute is then assigned both a monochrome and color physical attribute. Eight of the attributes are defined by TopView. The other eight can be modified by the application.

  To be more specific, each window in TopView is defined by a logical attribute table that contains eight logical attribute settings for monochrome displays and another eight logical attribute settings for color displays. Each monochrome logical attribute setting is a pointer to the monochrome attribute table that contains actual hardware device codes. Likewise, each color logical attribute setting is a pointer to the color attribute table that contains actual device codes.

  The advantage of using logical attributes rather than physical attributes is that the attributes are automatically chosen to suit the display currently in use. If you have selected Logical Attributes in the Panel Options menu, you can then make

changes to the Logical Attribute table by invoking the Set Attribute option in the Panel Options Menu.

- **Display Type:** This option allows you to specify whether your panel is to be used with a monochrome or color display when using physical attributes.

- **Uses Standard Attrs:** This option allows you to specify that your program uses the TopView standard attributes. Using standard attributes allows the TopView user to change the colors for the logical attributes. Otherwise, the attributes you specify remain in effect.

In addition to the panel options you can access from the Panel Menu, you also can use the Panel Menu to write and read panels, overlays, and templates.

Both a panel and an overlay contain information describing the appearance of a window. The difference between a panel and an overlay is that a panel supplies the initial information for creating a new window and setting its size, screen location, and fields, whereas an overlay replaces the contents of an existing window.

A template has text, attributes, and a particular position on the screen. For example, if you create a window that has a business logo in the bottom right corner, and you want to duplicate that logo on every window in that application, you can store the logo in a template file. Then, whenever you read in that template file and write it to a window, the logo is placed in the same location, with the same attributes. You usually mark the beginning and ending points of the template area before writing a template to a file.

You can write or read a panel, overlay, or template by specifying a file name and extension. You can optionally specify a drive and path.

### Field Menu

The Field Menu lets you define, copy, move, and test fields in your panel. You can define fields as input, output, select, or inactive. You can determine how data in the fields will look; whether you want to see field numbers; and whether you want to see all the data a user enters or just new data. All of these functions are explained below.

The functions in the Field Menu are:

- **Field Options:** When you select Field Options, a menu appears. Using the Field Options menu, you can select how your program receives the information that a user supplies.

The functions in the Field Options Menu are:

- **Field Format:** If you select Modified Fields Only, your program receives only those fields that the user has changed. If you select All Fields, your program receives every field in your panel, whether or not the field has changed.

- **Field Numbers:** Whenever you define a field, it is given a number. If you select the Field Numbers option, your program receives the field numbers associated with the fields in your panel. If you select No Field Numbers, your program does not receive the field numbers.

- **Auto Reset:** Auto Reset controls which fields are returned to your program. If you select Auto Reset On, any fields that were returned to your program are reset, so your program does not receive those fields again. If you select Auto Reset Off, fields are returned to your program whether or not they have already been returned.

- **Select Field options:** Select fields are those that the user selects from menus, lists, tables, etc. You can choose how you want select field information to return to your program.

  If you choose Select Field Code, you receive a Boolean result — Y if the field was selected, N if not. This format is useful for determining which menu choice the user has selected. If you choose Select Field Data, you receive the actual text within the field. This format should be used when the application changes the contents of fields, such as a directory list.

- **Button Usage options:** Normally Button 1 is used to select a field (which may bring up another window), and Button 2 is used to end a window and return to the previous status. You can change these settings in the Field Options menu, although the recommended settings are the defaults.

- **Pointed-At Attribute:** When the pointer is in a particular field in a menu, that field is being pointed at. Each pointed at field has an attribute associated with it. For example, in the Window Design Aid, the default attributes of a pointed at field on a color display are light gray foreground and blue background.

When you select Set Point Attribute, you can choose how the field will look when it is being pointed at by the user.

- **Selected Attribute:** When you actually select a field, that field may take on a different set of attributes. In the Window Design Aid, the attributes of a selected field on a color display are blue foreground and cyan background. These attributes also are assigned to default selections, such as in the Panel Options menu, where the Window Design Aid gives you a predetermined selection.

- **Define Fields:** When you select Define Fields, a menu appears. The Define Fields menu lets you specify what kinds of fields you want in your panel. You can also move, copy, and adjust the sizes of fields. When you define a field, you must first mark the field on the screen.

  Functions in the Define Fields menu are:

  - **Field Number:** The system assigns a field number to each field you define. Field numbers are assigned in numeric order. You can reassign numbers if you wish.

  - **Field Type:** There are four field types — Input, Output, Select, and Inactive.

    — **Input** allows both the program and the user to enter data into the field. The data entered replaces any data already in the field.

    — **Output** allows only the program to write data into the field.

    — **Select** allows the user to make a choice by using a pointing device. An application can write data into a select field.

    — **Inactive** allows you to make a previously defined field inactive without deleting it, eliminating the need to renumber all fields.

  - **Program Output** attribute enables a group of fields to be written to in one operation at application runtime. If you select No Program Output, then the field is not written to during the group output operation.

  - **Copy Field** allows you to mark a field in your panel and make a copy of it within that same panel.

  - **Move Field** allows you to move a field anywhere within a panel.

  - **Adjust Field Size** allows you to change the size of a field in your panel.

- **Test Fields:** This function lets you simulate using the panel you are creating. You can read in a panel and enter data in its fields. The Test Fields function immediately shows you how the fields reacted, and it displays information about the field such as its length and the starting and ending bytes.

  Results are displayed in a window called the Field Test Input Data window. This window also contains three menu options: Reset Fields, Resume Test, and End Test. You must select End Test to quit the field test mode and return to the Field Menu.

- **Show or Hide Field Numbers:** If you want to see the assigned field number when you display the field, you should select Show Field Numbers; if not select Hide Field Numbers.

- **Renumber Fields:** This function lets you eliminate all reference to inactive fields by renumbering the remaining active fields in numerical order.

- **Delete All Fields:** This function allows you to erase every field in your panel.

### Using Windows When You Write a Program

When you write your program, you can use the windows you designed in the Window Design Aid by writing interrupt calls to TopView in your code. These calls tell TopView to perform the input and output that is necessary to read in and manage the displaying of your windows. The way you specify the call depends on the language you are using.

### Panel Utilities

The TopView Programmer's ToolKit contains two utilities that support panel usage in TopView applications.

The PANELLIB utility merges a group of panels into one file. This simplifies panel operations and reduces the storage required by panels in the application.

The PANELOBJ utility converts a panel or a merged panel file to an object module. This object module can be linked with an application so that the panel(s) reside in storage while the application is executing. This is advantageous in an application that

considers execution speed more important than the amount of memory required. This utility also makes it possible for an application to use panels without having to keep panel files online.

### Language Interfaces

Applications should use the TopView system functions to take full advantage of the facilities provided by the TopView environment. The Programmer's ToolKit provides you with an interface between TopView and the Macro Assembler. This interface provides you with access to the TopView windowing and multitasking facilities. The interface between TopView and an Assembler program is implemented using the Intel 8088 software interrupt mechanism.

The ToolKit also provides an example language interface for IBM Pascal. Programmers may use this Pascal example as a guide for creating a language interface.

### High-Level Interface Sample Package

Examples are provided for the IBM Macro Assembler and IBM Pascal to show how to code, and/or use, a high-level interface to TopView. The sample interface is composed of a set of routines which provide some of the more commonly used functions, such as applying panels, retrieving user input, processing field information, etc. This particular interface concentrates on the access and manipulation of Window and Panel objects. The reader is free to modify or extend the interface to handle the other objects in the system.

This sample package makes use of panels which have been created using the Window Design Aid. The high-level routines make use of single panels created by the Window Design Aid, as well as panels processed by the PANELOBJ and PANELLIB utilities. The assumption is made that the field options of the panels are the default options of the Window Design Aid. Unpredictable errors may occur if these options are modified unless corresponding modifications are made in the high-level routines.

Default Options:

Modified Fields Only
Field Numbers
Auto Reset On
Select Field Code
Button 1 Select
Button 2 Status

The interface routines are coded in Assembler and can be used by either Assembler or Pascal applications. An include file has been provided for the Assembler programmer, containing all the macros needed for the interface. Likewise, an include file has been provided for the Pascal programmer, containing all the procedure and function declarations needed. The Assembler macros have been designed to use the same parameter passing technique as is used by the Pascal procedures and functions; therefore, most of the routines have the same entry point regardless of whether they are called from Pascal or Assembler. The routines with different entry points are those which require the address and the length of a character string to be passed to them. The Pascal application will pass the parameter as a type LSTRING (first byte of the string denotes the string length), whereas the Assembler application will pass the parameter as a character string and string length.

Included in the sample package are two programs which use the interface, one coded in Assembler using the macros, and the other in Pascal using the procedure and function calls.

### Compatible Applications Types

Many applications that currently exist for the IBM PC are already compatible with TopView. The compatibility of an application with TopView can be classified as follows:

- **DOS Application:** No knowledge of TopView. The TopView features available are at least program switching and copy. Additional features may be available if the application follows certain guidelines. TopView can then provide resource sharing services, i.e., multitasking and windowing in addition to program switching and copy/paste.

- **TopView Aware:** These applications can run with TopView or in a standard PC-DOS environment. The applications use two TopView- provided Interrupt 10 BIOS calls to write to the screen instead of writing directly to the hardware video buffer. Refer to **Interrupt 10H Function Calls** for details. Multitasking, windowing, program switching and copy/paste are available.

- **TopView Specific:** The application was designed using the TopView API (as described in the ToolKit) and has access to all TopView features.

## Writing Compatible Applications

When you are programming for the TopView environment, you must be aware that several other things are happening internally. TopView intercepts system calls and maps them into specific TopView functions. Your application must follow certain guidelines to remain compatible with the TopView environment.

Certain programming techniques can cause problems for TopView when it tries to run your application. These techniques should be avoided, when possible, because they limit the TopView features that will be available to your application.

## Direct Video Memory Access

Programs that write directly to the hardware video buffer can run only when they are the foreground (interactive with the keyboard/ mouse device) application. These applications are suspended when their window is not the topmost window in the system, i.e., the application is suspended in the background. In the foreground, these applications take over the entire screen and cannot be windowed. That is, the user cannot use the Size, Move, or Zoom functions to change the size or location of the window.

## Interrupt 10H Function Calls

A new set of video BIOS function calls is available that allow an application to directly access video memory whether the application is running with TopView or without TopView. Use of these function calls allows your application to run concurrently with other applications in the TopView environment. If the application is running in a non-TopView environment, BIOS considers these calls as a no operation function. All other video BIOS function calls operate as described in the IBM Personal Computer Technical Reference Book.

All registers remain unchanged by these function calls unless otherwise noted.

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

Input:

AH = 0FEH

On entry, ES:DI contains the assumed address of the hardware video buffer. These values are as follows:

— For a monochrome display:

ES = 0B000H
DI = 00000H

— For a color display:

ES = 0B800H
DI = 00000H

Output:

On return, ES:DI contains the actual address of the video buffer. If the actual video buffer address is different from the assumed buffer, your program is running under TopView and must use the specified address to access the 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.

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

This call is used when the video buffer has been changed and the application wishes to insure that the user sees the changes.

Input:

AH = 0FFH

ES:DI = Pointer to the first character and attribute in the video buffer that has been modified.

CX = Number of sequential characters (and attributes) that have been modified.

Output:

On return, all registers are unchanged.

## Coding Tips For Using The Update Video Buffer Call

Use of the Update Video Buffer call (INT 10H, AH = 0FFH) under the existing DOS/BIOS environment has no effect. In fact, the call need not be performed if the video buffer address returned by the Get Video Buffer call (INT 10H, AH = 0FFH) is one of the standard hardware addresses (i.e. it is the same as the video buffer address supplied). However, when the video buffer is not a standard hardware address, the Update Video Buffer call must be used to insure correct program operation. The frequency and manner in which this program is used can have a significant impact on the application's performance.

The Update Video Buffer call does not need to be issued immediately after changing the video buffer.

It should be issued only at the point where the application must guarantee that the user can see the updates on the display screen.

This may require some trade-offs to be made when designing an application. For example, suppose ten characters are updated in non-contiguous, widely separated portions of the video buffer every time a function key is pressed. There are two acceptable solutions. 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 normally the preferred (and simplest) solution.

The second approach is to perform all updates to the video buffer and then issue a single Update Video Display call. This call specifies the address of the first character that was modified in the buffer and a length that includes all characters between (and including) the first character modified and the last character modified. Note that the length can wrap lines on the display, so a 25 row by 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 because the amount of time to perform an Update Video Display call is roughly proportional to the number of characters in the call (i.e., in CX). Ten calls of length 1 are probably faster than one call of length 200. However, there is a threshold at which this is no longer the case, So, 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 its user interaction. The following are a few simple rules of thumb:

— 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 scattered locations in the video buffer, issue single character requests to update each position rather than one large request.

— If a major update is to be performed, modify the buffer as required and then issue a single Update Video Display call which specifies the length of the entire buffer (or as much of it as needs to be updated).

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.

### Restrictions On Using The Get Video Buffer and Update Video Display Calls

The Get Video Buffer call and the Update Video Display call apply only to applications that run in text video mode. Incorrect operation will occur if these calls are issued from an application running in a graphics mode (i.e., video modes 4, 5 or 6).

In addition, 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 each time the video mode is changed and use the returned video buffer address.

### Graphics Support in TopView

TopView supports graphics applications by giving them control of the full screen at the time the application is the foreground task. Both medium resolution color (320 x 200 four color pixels) and high 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 the hardware video buffer, graphics applications can run only in the foreground with a full-screen window and cannot be moved, sized or scrolled. Several graphics applications can be started at the same time under TopView, but they are not allowed to run as background tasks.

When an application switches into a graphics video mode using the BIOS Interrupt 10H, TopView allocates a 16K image buffer for the application's logical window. If an application currently doing graphics switches back to a text video mode, TopView de-allocates the 16K image buffer and allocates a logical window buffer based upon the window's maximum size as defined in the application's Program Information File. Refer to the section **Restrictions On Using The Get Video Buffer and Update Video Display Calls** for additional information about changing video modes.

### Programming Guidelines

The following programming guidelines should be considered when writing applications compatible with TopView.

- Absolute Memory Loading

  TopView loads applications anywhere in free memory. An 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 system units with over 512 KB of system memory will work 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.

  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

  For TopView 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. If your application controls a hardware device, your program must allow other applications to share that device. TopView can provide multitasking services only when TopView controls the hardware devices. An application can control some special devices; however, all applications 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. Applications should explicitly open and close shared devices using standard system names and use DOS or BIOS calls to access the devices. This allows TopView to provide resource sharing for applications that use the standard system devices.

- Manipulation Of Interrupt Vectors

  Software interrupt vectors are an effective method of transferring control between sections of code in an application. 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 allows an application to use the user-assigned software vectors. For example, an application can share and transfer control between common code without worrying about whether or not other tasks in the system also use those interrupt vectors. However, 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), in a special way. TopView maintains these vectors separately from the range of software vectors saved for a particular task. 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 results. Since these three vectors are handled differently than others, your program's information file need not include them in the range of vectors to be swapped.

  TopView also maintains the Keyboard Break (1BH) and Timer Tick (1CH) interrupt vectors automatically for each task running in the system. If your application modifies these vectors, it does not need to specify them in the program's information file.

  TopView does not maintain the hardware vectors (08H to 0FH) on a per-task basis. The hardware vectors are not changed and remain in effect from one task to the next. This is because of the asynchronous nature of hardware interrupts; the vectors must remain in effect to service the hardware they are associated with. Applications should avoid controlling the hardware directly, so that two or more programs do not attempt to control the same hardware device.

  The keyboard hardware vector 09H is handled differently. If an application changes the keyboard hardware vector, TopView notes the change and uses the new vector whenever a keyboard interrupt occurs. This vector is in effect only when the application that changed the vector is the foreground application. In this way, each task running in the TopView environment can define its own keyboard interrupt handler. Like the Keyboard Break and Timer Tick vectors, if your application modifies the keyboard interrupt vector, it does not need to specify that in the program's information file.

- Multiple Applications / Load and Stay Resident Applications

  Problems may occur if your program loads another program that checks to see if it has been loaded in the same session, and then tries to use the previously loaded copy of the program. TopView may not be able to determine which task is running that code.

  Applications which use a load and stay resident structure to provide shared code are examples of this. A user may start a program that has a load and stay resident module (dialog manager, for example). A second copy of the program could be started. Since the load and stay resident module is outside of TopView's control, problems may occur because two programs are using the same code.

- Copy Protection

  If a DOS program has a copy protection scheme, it must not violate any of the TopView compatibility rules. Any application using one of the following copy protection methods may not work with TopView:

  - Requires booting a special diskette
  - Uses a modified DOS
  - Requires critical timing to read disk sectors or tracks

- Supporting DOS File I/O

  DOS 2.00 provides for multiple directory access, installable device drivers to handle nonstandard devices and redirected I/O for standard input devices. TopView provides multiple DOS tasks with the same environment as if they were single tasks running under DOS. In other words, 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. This 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 restarted, the original default values (from the application's PIF) are restored.

### DOS Restrictions

The DOS interface in the TopView environment is basically unchanged. There are a few restrictions with DOS:

- Batch File Support

  Batch files are not supported by TopView.

- 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. An application should avoid keeping files open the entire time the application is running to insure that handles are available for other applications that are running concurrently.

- File Locking

  An automatic file locking mechanism is provided in the TopView system that is intended to protect your task from other tasks updating the file you are using. However, multiple tasks can open the same file at the same time for read-only access. Refer to the chart for specific results of trying to open a file. In general, a lock is activated when a file is opened. The lock remains active until the file is closed or the task is ended. The task that opens the file can access that file as many times as required; however, if the file was opened for write or read/write or using FCB's, 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.

| | Current File State | | | | |
|---|---|---|---|---|---|
| | Closed | Opened | | | |
| **Request for Open** | | **Using FCB** | **Using ASCIIZ AL = Read** | **Using ASCIIZ AL = Write** | **Using ASCIIZ AL = Read/Write** |
| Using FCB | Open | Error | Error | Error | Error |
| Using ASCIIZ AL = Read | Open | Error | Open | Error | Error |
| Using ASCIIZ AL = Write | Open | Error | Error | Error | Error |
| Using ASCIIZ AL = Read/Write | Open | Error | Error | Error | Error |

**Figure 1. Results From Attempting To Open A File**

# Program Information

TopView must have access to certain information about the special operating characteristics of each application. This information, such as display mode, program name, program location, etc., is stored in a file called the Consolidated Program Information File. When an application is added to the Start a Program menu, TopView creates a record for the application in the Consolidated Program Information File. An application cannot be started by a user until the application has been added to TopView.

### Program Title

This is the name by which the user knows the application. This name can be up to 30 characters in length and is displayed in the Start a Program menu and the Switch menu. The name does not have to be the same as the name on the application diskette and can be changed by the user (using the Change Program Information function).

### Program Pathname

This is the full pathname and program name of the file that starts the program. This file should have an .EXE or .COM extension. The maximum length is 64 characters.

### Program Parameters

This is a string of characters that contains any additional variables that should be added to the invocation string of the program. If a question mark is entered as the only character of the string or as the last character of the string, TopView prompts the user for the program parameters when the application is started. All characters in the string (except the ending question mark) are displayed to the user in the prompt. The maximum length is 64 characters.

### Data Files Location

This is the default drive and directory in effect when the application is loaded. When a program is added using the Other option of Add a Program, the data files location defaults to the same drive and directory as the program location (program Pathname). The maximum length is 64 characters.

### Memory

These fields indicate amount of memory required to run the application. Three numbers must be specified: minimum, maximum and system memory.

System memory is the memory TopView uses to store your program's system control blocks and video buffer. The minimum system memory is 7KB. A program using graphics requires an additional 16KB of system memory.

When the application is started, TopView attempts to provide the maximum amount of memory specified. If that amount is not available, TopView attempts to allocate the minimum amount of memory specified. If more than the minimum but less than the maximum amount is available, the intermediate amount of memory is allocated. The

minimum and maximum specifications should not include memory required for DOS. Both fields are specified as K bytes of memory.

### Screen Type

This is a one-character field that indicates the video mode and/or monitor your application uses.

2 = 80 X 25 black and white
3 = 80 X 25 color
4 = 320 X 200 color
5 = 320 X 200 black and white
6 = 640 X 200 black and white
7 = 80 X 25 monochrome

X = Any display can be used. If the application changes the video mode, TopView accommodates the change.

D = Use the display mode in effect when TopView was started.

### Screen Pages

This is the number of screen pages that the application uses. Most applications use only one page; however, from 1 to 8 pages can be specified.

### Window Size

This is the size of the window initially created for the application. This is specified as rows and columns. For most existing applications that use the full screen, the window size is 80 columns by 25 rows.

### Initial Offsets

This is the initial location at which TopView is to position the window when the application is started. For applications that use the full screen (most existing applications), the offsets are row 0, column 0.

### Shared Program Pathname

This is the name of a program your application can share with other applications. If the application uses the filter program and a filter table, specify the filter program's name in this field. The filter program that is provided by TopView is FILTER.EXT. This field can be up to 64 characters in length.

### Shared Program Data

This is a character string that contains any data that should be passed to the shared program. It might be a file name or a string of data. If the application uses the TopView filter program, enter the name of the application's filter table file. A filter table file name has a suggested extension of .TBL. The maximum length of this field is 64 characters.

### Range of Software Interrupt Vectors Swapped

This is the range of software vector interrupts changed by the application. The maximum range is from 00 to FF. This information is saved by TopView when another application is dispatched and restored when the first application is dispatched. Changing standard interrupt vectors has two consequences: an increase in the TopView memory required to run the program and a slight decrease in the performance of the program. The amount of memory and the decrease in the performance is directly proportional to the number of interrupt vectors specified. The range is specified as two fields, a low value and a high value. If your application does not swap vectors, these fields should be blank.

### Does This Program Write Directly To The Screen

Valid input for this field is Y for yes or N for no. This information is used by TopView to determine if the program writes directly to the video buffer (address B0000H for a monochrome display or B8000H for a color display). Programs that write directly to the video buffer cannot be windowed or run in the background.

### Accessing System Keyboard Buffer

Valid input is Y or N. Programs that access the BIOS keyboard data areas cannot be allowed to run in the background.

### Should This Program Run Only In The Foreground

Valid input for this field is Y for yes and N for no. This information is used by TopView to determine if the program should only run in the foreground (when the program is interactive with the keyboard/pointing device). However, programs

that write directly to the video buffer using the TopView INT 10H calls are well behaved and can be windowed and run in the background.

**Does This Program Use The Math Co-processor**

Valid input for this field is Y for yes and N for no. This information is used by TopView to determine if the program uses the IBM 8087 or 80287 Math Co-processor to do arithmetic.

# Guidelines For Compatibility Testing 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. The following types of testing should be done before an application can be designated as compatible with TopView:

- Make sure the Program Information data is correct for the application, especially the memory requirements and the software interrupt vectors.

- Test the application (by itself) with TopView.
  - 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.

- Perform the same tests as above except with multiple copies of your application started.

- Test the application in combination with other applications and TopView.
  - Do the same testing as in the second bullet except with multiple applications started.
  - If the application runs in the background, test all functions that run in the background while another application is running in the foreground.
  - Test the application in the foreground with application(s) running in the background.
  - Suspend and hide the application.
  - Run the application with other applications suspended and hidden.
  - Run your application with other applications using different start orders (i.e., start your application first, start your application last, etc.)
  - Attempt to use the printer from your application (if appropriate).

- The testing must be done for each possible combination of the following system configuration variables that may be applicable to your application:
  - 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

IBM Corporation
Editor, IBM Personal Computer Seminar Proceedings
4629
Post Office Box 1328
Boca Raton FL 33432