Exchange

March/April 1986

Hardware

Software

Random Data

Departments

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.

Managing Editor	Michael Engelberg
Technical Editor	Bernard Penney
Associate Editor/ Design Director	Karen Porterfield
Writer/Automation Consultant	John Warnock
Editorial Assistant	Lowell Blackham
Production Automation Consultant	Greg Klipstein
	Sherry Reardon
Illustrator	Jeff Jamison
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 (4704)
P.O. Box 3022
Boca Raton, FL 33431-0922

To correspond with *Exchange*, please write to:

Editor, *Exchange*
IBM Corporation (4704)
P.O. Box 3022
Boca Raton, FL 33431-0922

POSTMASTER: send address changes to *Exchange of IBM PC Information*, IBM Corporation (4704), 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.

Principles of Disks and Diskettes

Robert A. Flavin
IBM Corporation

This article is about the principles of IBM Personal Computer disks, diskettes and their corresponding drives.

Disk Terminology

Before we look at the details of the IBM PC's disk system, let's review some of the basics of fixed disks and diskettes.

In a computer disk system, a smooth disk, whether a fixed disk or a diskette, is coated with a magnetic material and spun around like a phonograph record. A head, similar to a small, movable tape recorder head, sits on (or near) the disk. The head reads and writes data on the disk that spins underneath it. Figure 1 illustrates the various parts of a disk.

A phonograph record has one continuous spiral groove on each side, so the record player's arm moves toward the center of the record as it plays. In contrast, a computer disk consists of a series of concentric circles called tracks. The disk head reads from and writes to a single track at a time.

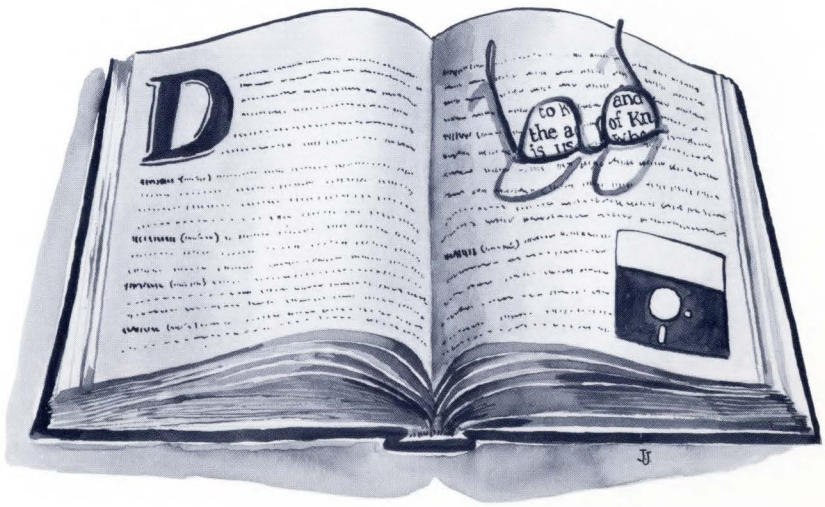
Because the amount of data that can be stored on a track is more than the computer can conveniently handle, tracks are partitioned into sectors. A sector is the smallest unit of storage on a disk.

If the disk head were not allowed to move, only one track of the disk could be used, and the disk would not be able to store much data. By moving the head in or out along a radius of the disk, additional tracks, physically smaller and larger, can be reached. This increases the amount of data that can be stored on the disk at the expense of the time needed to move the head to the desired track and the expense of mechanical complexity. Once the head reaches the right track, it remains in that position until another track must be accessed.

Both sides of a disk can be coated with magnetic material to double the storage capacity of the disk. Rather than requiring the disk to be flipped like a record, the disk drive can be outfitted with two heads, one on each side of the disk. The IBM Personal Computer family has diskette drives (360KB and 1.2MB) that accommodate double-sided diskettes.

On a fixed disk drive, one set of concentric tracks resides on a single "platter" that resembles a diskette. The fixed disk drive actually consists of two such platters stacked on a single spindle, with one head on each side of each platter, for a total of four heads. The fixed disk drives in the IBM Personal Computer family are capable of storing either 10MB, 20MB or 30MB of data.

Because the two platters in a fixed disk drive are stacked vertically, their tracks also are stacked. Similarly, a double-sided diskette has stacked tracks. In both cases, one stack of tracks, all of which have the same radius, is called a cylinder.



The concept of a cylinder is important for the sake of efficiency. One cylinder is the maximum amount of storage that the disk system can access without moving the heads.

The four heads in a fixed disk drive, or the two heads in a double-sided diskette drive, are connected to the same arm. Like the tracks, the heads are vertically stacked. The arm moves all the heads in or out at the same time, and all the heads together access one cylinder.

Numbering schemes are used to refer to each part of a disk system. The outermost (largest) cylinder is cylinder 0, and cylinders are numbered sequentially. Heads are numbered from 0, but the order in which they are numbered does not matter. Tracks are not numbered, because a track's identity is determined by the combination of its cylinder number and head number. Sectors are numbered starting with 1. For performance reasons, sectors are not physically arranged in numerical order around the disk (see the next section, *The Operating System's View*).

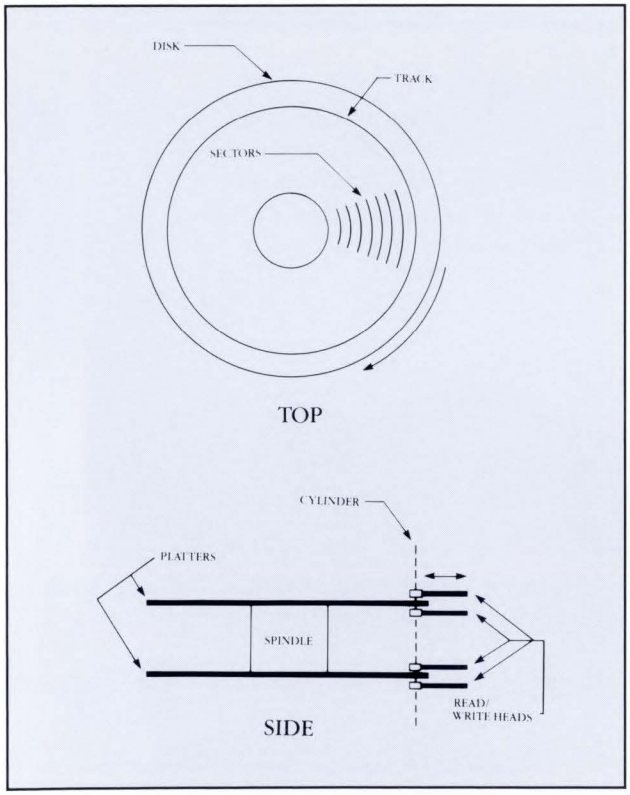


Figure 1. Disk Components

Disks come in a variety of configurations. All disks were once hard disks, with platters made of rigid aluminum coated with magnetic material. Diskettes, more recently developed, are made of flexible mylar, then coated. Because they flex, they are called floppy disks or floppies. The introduction of diskettes made it necessary to distinguish the old disks by calling them hard disks. A hard disk that is permanently mounted in a disk drive is called a fixed disk; otherwise it is called a removable hard disk.

The Operating System's View

One of the operating system's tasks is to organize the sectors on the disk into a file system that is convenient for programs to use. To mask the complexities of dealing with sectors, tracks and cylinders, the operating system uses sectors in the order of relative sector number. The operating system begins by using all sectors on one track; next, it uses all tracks (heads) in one cylinder; finally, it uses cylinders from the outermost to the innermost.

The operating system is responsible for managing the use of sectors, allocating the sectors to files, and remembering which sectors remain free for future use. Because it may be inconvenient for a microcomputer to manage the tens of thousands of sectors on a large disk, the operating system groups adjacent sectors into a unit of storage called an allocation unit or cluster.

In IBM Personal Computer DOS, the numbers of sectors in a cluster must be a power of 2. The fixed disk may have two, eight or 16 sectors per cluster, depending on the size of the disk, whereas a single-sided diskette has one sector per cluster. Cluster size is determined at the time the disk is formatted; in the case of fixed disks, cluster size is affected by the size of the DOS partition.

Data Recording Techniques

In some pursuits, getting the inside track is an advantage, but it doesn't work that way with disks. The outer tracks on a disk have larger circumferences than the inner tracks. Therefore the outer tracks provide more linear room in which to store data.

Some disk systems change the linear density of data on a track, depending on the circumference of the track. This technique squeezes more storage out of the diskette but increases the complexity of the hardware and software. The IBM Personal Computer maintains a constant amount of data per track, making the inner tracks more densely recorded than the outer ones.

The tracks on 5-1/4 inch diskettes are physically spaced at 48 tracks per inch. Each side of the diskette contains 40 tracks, so the 40 tracks form a band that takes up less than one inch of the 5-inch diameter. (The disk itself is 5 inches in diameter; its protective cover is 5-1/4 inches in diameter.) The band of 40 tracks is located near the outer edge of the disk because there the tracks can have the largest circumferences. Tracks with large circumferences make the least demands on the quality of the magnetic material and the precision of the head.

Recording density means the linear density of bits around a track. Early hobby computers recorded data at a density that has become known as single-density. The IBM Personal Computer was originally announced with single-sided diskettes that use double-density recording—twice the number of bits per track. Soon thereafter, IBM introduced double-sided diskettes for the Personal Computer family. The diskette capacity doubled, but the recording density remained the same, double-density.

The IBM 1.2MB High-Capacity Diskette Drive for the Personal Computer AT uses quadruple-density recording, which records at twice the linear density of the older diskette drives. The 1.2MB drive also has 80 tracks per side, spaced at 96 tracks per inch.

Disk Capacities

The following discussion of disk capacities assumes the use of DOS 1.10 or higher for formatting diskettes.

A double-sided diskette can be formatted to store 360KB of data. This storage is calculated as follows:

A double-sided diskette has 40 cylinders—40 positions to which the heads can move. Each cylinder has two tracks, one on each side of the diskette. Each track has nine sectors. Data can be stored so densely along a track that, in a single sector, 512 bytes can be stored with room left over.

Multiplying all these numbers together:

```
   40 cylinders per diskette
x   2 tracks per cylinder
x   9 sectors per track
x 512 bytes per sector
```

equals 368,640 bytes of data. Computer people define the number 1,024 (2 to the tenth power) as 1K, so 1,024 bytes equals 1K bytes, or 1KB. To reduce 368,640 bytes to a manageable number, divide 368,640 bytes/diskette by 1,024 bytes per KB. The result is 360KB of storage on a double-sided diskette.

A single-sided diskette has only one track per cylinder, so similar calculations result in 180KB of storage.

A 1.2MB high-capacity diskette has 96 tracks per inch and 80 tracks per side, twice as many as the older diskettes. Finally, a high-capacity diskette has 15 sectors per track.

Multiplying:

```
    2 sides/diskette
x  80 tracks/side
x  15 sectors/track
x 512 bytes/sector
```

yields 1,228,800 bytes/diskette. Dividing this by 1,024 bytes/K yields 1,200KB/diskette. Next, 1,024KB equals 1MB, so 1,200KB divided by 1,024K/MB yields 1.17MB as the amount of storage on a high-capacity diskette.

One of the operating system's tasks is to organize the sectors on the disk...

Calculations for the 10MB fixed disk drive are as follows:

```
  306 cylinders per disk
x   4 tracks per cylinder
x  17 sectors per track
x 512 bytes per sector
```

equals 10,653,696 bytes, or 10,404KB, or 10.16MB of data.

Calculations for the 20MB fixed disk drive are:

```
  615 cylinders per disk
x   4 tracks per cylinder
x  17 sectors per track
x 512 bytes per sector
```

equals 21,411,840 bytes, or 20,910KB, or 20.41MB of data.

Calculations for the 30MB fixed disk drive are:

```
  918 cylinders per disk
x   4 tracks per cylinder
x  17 sectors per track
x 512 bytes per sector
```

equals 31,961,088 bytes, or 31,212 KB, or 30.48MB of data.

Disk Overhead

There is a difference between the amount of data that theoretically can be stored on a track and the amount that can be stored during actual use. The following discussion is based on the characteristics of a 360KB, 5-1/4 inch diskette drive.

The IBM Personal Computer file system...repackages the basic marvels of magnetism into a form useful for storing data in computers.

Recall that a track on a diskette has nine sectors, and each sector can hold 512 bytes of data. Therefore, a track actually holds 9 times 512, or 4,608 bytes of data.

However, the unformatted capacity of that diskette track is 6,250 bytes, calculated as follows:

The characteristics of the magnetic material and the head are such that, on the innermost track of a diskette, where the recording density is highest, the drive can reliably record 250,000 bits per second. Also, the diskette rotates at 300 revolutions per minute, so it takes 0.2 seconds for the disk to rotate once (0.2 seconds equals 60 seconds per minute divided by 300 revolutions per minute).

The amount of data that can be stored on a track is the amount that can be written during one revolution of the diskette. In one revolution, which takes 0.2 seconds, the diskette drive can write 0.2 seconds/revolution times 250,000 bits per second, which equals 50,000 bits per revolution or per track. These 50,000 bits equal 6,250 bytes per track (50,000 bits/track divided by 8 bits/byte).

Why is there such a difference between theoretical capacity of the diskette and the amount of usable storage? The answer is overhead. Every sector has some overhead—some space used by the disk system itself—for control information and to permit the drive's electronics to work properly.

Each sector starts with a sector header that identifies the track, head and sector number. This sector header is written to the disk when it is formatted and is never changed (unless the disk is reformatted).

The sector header is used to confirm to the drive adapter—the interface electronics between the drive and the Personal Computer—that the head is on the right cylinder, that the correct head is active and that the correct sector is under the head. Immediately preceding the sector header are some synchronization signals that allow the drive adapter to get ready to read the header. Following the header is a Cyclic Redundancy Code (CRC), which is used to verify that the sector header was read correctly. All of these things contribute to sector overhead.

Following the header is a gap in the recording. This gap gives the electronics time to change between read mode and write mode. This gap is followed by another group of synchronization data. Next comes the actual data itself. Following the data is another CRC, which verifies that the data was read correctly, and another, larger gap before the header of the next sector.

On the IBM 10MB, 20MB and 30MB fixed disk drives, the second CRC is replaced with an Error Correcting Code (ECC). Although the ECC is somewhat longer, it is used not only to detect the occurrence of some errors, but also to correct many of the simple errors that can occur during reading.

The IBM Personal Computer file system is a complex combination of hardware and software. It repackages the basic marvels of magnetism into a form useful for storing data in computers.

Extended Memory or Expanded Memory

Dave Hoagland
Lawrence Livermore National Laboratory

Extended memory and expanded memory are two terms that often confuse computer users. This is understandable since both terms describe ways to append additional memory onto the 640KB maximum that DOS allows for active memory.

Extended Memory

Extended memory, limited at the moment to the IBM Personal Computer AT, is memory located above one megabyte (1MB). Extended memory can be addressed either directly (with a different command processor and the AT running in protected mode) or by using bank switching techniques.

Bank switching schemes fool the software into accessing extra "pages" or "banks" of memory just as if they resided within active memory. Direct addressing, on the other hand, allows the software to access extended memory at unique locations outside of active memory.

Expanded Memory

Expanded memory is constrained to bank switching. Because memory addressing in the PC and PC XT is limited to the first 640KB, any extra memory must be switched **in place of** a portion of active memory within the 640KB boundary.

Bank Switching Analogy

To explain the concept of bank switching, it may be helpful to visualize a stack of books. Each book represents 64KB of active memory. You are allowed to stack books on top of each other to a maximum of ten books, or 640KB. At this point, you cannot add any more books.

The only way you can use more books in your stack is by removing one, storing it somewhere else, and replacing it with another. The switching of the top book in the stack with other volumes (those for which there was no room on the stack) could be considered "bank switching."

In the expanded memory boards currently available, this bank switching occurs in blocks of either 64KB or 256KB, depending on the manufacturer.

...the PC takes very unkindly to attempts to infringe on its space.

Where the System Resides

Since the 8088 and 8086 processors in personal computers are capable of addressing memory up to 1MB, some of you may wonder what goes on in that never-never land between 640KB and 1MB. This is space reserved for the system itself, including such things as video memory. Suffice it to say that the PC takes very unkindly to attempts to infringe on its space. The table below helps clarify the way DOS utilizes the 1MB available in the PC and PC XT.

Address	Name	Function
000000-07FFFF	512KB system board	System board memory
080000-09FFFF	128KB	I/O channel memory - IBM Personal Computer AT 128KB Memory Expansion Option
0A0000-0BFFFF	128KB video RAM	Reserved for graphics display buffer
0C0000-0DFFFF	128KB I/O expansion ROM	Reserved for ROM on I/O adapters
0E0000-0EFFFF	64KB Reserved on system board	Duplicated code assignment address FE000
0F0000-0FFFFF	64KB ROM on system board	Duplicated code assignment at address FF0000
100000-FDFFFF	Maximum memory 15MB	I/O channel memory - IBM Personal AT 512KB Memory Expansion Option
FE0000-FEFFFF	64KB Reserved on system board	Duplicated code assignment at address 0E0000
FF0000-FFFFFF	64KB ROM on system board	Duplicated code assignment at address 0F0000

Table 1. PC System Memory Map

Electrostatic Discharge and the Computer Connection

Mark L. Weber
North East Indiana IBM PC Club

We have all walked across a carpeted room to turn on a light switch, and been startled by the stinging sensation of an electrostatic discharge between our fingers and the switch. The shock didn't cause us any harm, but if this type of discharge were to occur between pieces of your computer system, the chances are good that at least one of the components would be damaged.

You don't have to be a physics major to understand the basic principles of how static electricity can wreak havoc with your equipment. Indeed, computer users can take a few simple precautions that will protect their computer and associated peripherals from electrostatic discharge.

When objects are isolated from each other, they tend to gain or lose electrons. Even the friction of air passing over an object can strip electrons from the surface of that object. Likewise, people shuffling their feet across a rug can cause an excess of electrons to accumulate on their shoes. The more time that two objects are isolated from each other, the more likely it is there will be a difference in the electrostatic charge of the two objects. The difference in electrical potential between isolated objects can easily be as high as 30,000 electron volts. When differently charged objects come physically close to each other, electrons flow from the object with the most electrons to the object with the least electrons. This flow of electrons lasts only a split second, yet the rate of the current flow will be extremely high.

In a personal computer, the maximum current flow that microchips can handle is 15 to 30 amps, depending on the chip involved. You can imagine what would happen if an electrostatic discharge of 30,000 electron volts were to occur between two pieces of your computer system.

To prevent such a discharge from occurring the next time you open or re-connect your computer, simply plug **all** components into the power receptacle before connecting them to each other. This way, any difference in charge between the devices will result in a flow of current between the ground systems of the two components. Most connectors used in interfacing computer systems are designed to allow the ground connection to be made before the critical "signal" lines make contact. No damage will occur when you connect your components together because the electrostatic charge in each component will be the same. This one little rule will prevent electrostatic discharge from ruining any part of your computer system.

Don't Forget: Park Your Heads!

Don Scanlon
IBM Corporation

(Editor's note: Mr. Scanlon is a Service Planning Representative for IBM's ISG Service Business Product Planning.)

Often I find that PC users neglect to "park" the read/write heads of a fixed disk before they move their computers. I would like to emphasize that it is crucial to park the heads.

The read/write heads on a fixed disk remain where they were last positioned. Therefore, when you power off your PC, the heads are positioned over the last cylinder your program accessed. Shaking or jarring the heads while transporting the PC can damage the oxide surface of the fixed disk. This can result in disk errors and loss of your data.

To park your read/write heads, use the Diagnostics diskette that came with your Guide to Operations manual, or the Diagnostics diskette from the PC Hardware Maintenance and Service manuals. Insert the Diagnostics diskette and boot your system. After the program loads, you will see several options. Select option 3, Prepare System for Relocation. This option, which runs in only a few seconds, will park the read/write heads.

This short procedure will go a long way toward preventing fixed disk head crashes.

The IBM PC Network Analysis Program

Pat Tittizer
IBM Corporation

The Need for Network Management Tools

The IBM Personal Computer Network is a local area network (LAN) designed to make personal computer users more efficient and productive. By connecting many personal computers together, the LAN lets users send and receive files, have common disk files, and share printers with many users. This increases the resources available to each user at less overall cost.

As the size of a local area network increases, however, a component failure (adapter card or cable) or resource bottleneck (printer or disk constraint) within the network becomes more likely.

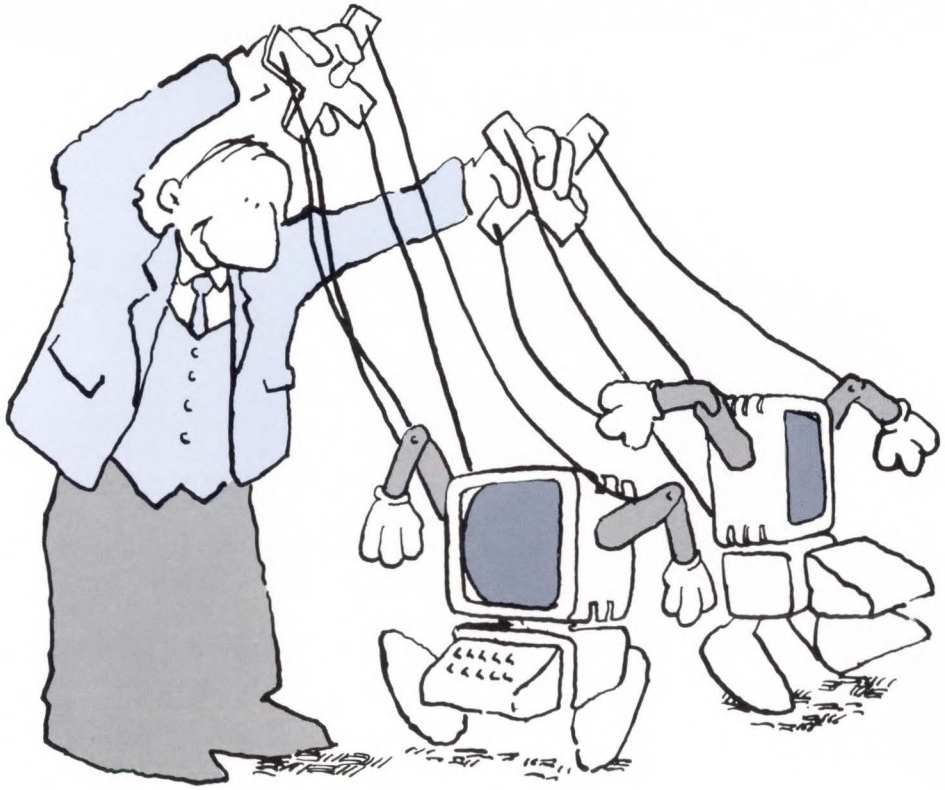
Supporting the users of the network and planning for expansion also becomes more difficult. Failure to correct these situations can result in inconvenience and expense.

In network management, it is important to avoid these problems, and keep the users as productive as possible. Overall, some of the major objectives of network management are to:

- Reduce the time and personnel required to locate problems.
- Improve the availability of the network.
- Maintain a list of end-users.
- Improve the end-user service.
- Provide data for preventative maintenance of the network.
- Maintain an inventory of equipment.
- Assist in planning for expansion of the network.
- Ensure that change in the network is orderly and controlled.

Overview of the IBM PC Network Analysis Program

The IBM PC Network Analysis Program provides a set of functions to help you manage your Network.



The Personal Computer Network Analysis Program can help you:

- Diagnose problem conditions.
- Locate performance bottlenecks.
- Install new PC Network Adapters on a network.
- Maintain an inventory list of adapters and users on a network.
- Check the function of a PC Network Adapter.
- Plan the growth of a network.

The program runs stand-alone on one of the PCs in the network. It monitors the other computers on the network independently of any software that they are running.

The PCN Analysis Program is completely menu-driven, with each menu uniquely numbered. Most of the menus have on-line help panels. Each menu is also explained in the *User's Guide*. The *User's Guide* index lists the menus by number and by name.

The PCN Analysis Program includes a demonstration mode to help you become familiar with the product. The sample data presented in demonstration mode typifies a network with a number of users, some of whom are experiencing problems. You can enter or leave demonstration mode at any time by selecting "Start or Stop Demonstration Mode" on the Main Function Selection menu, shown in Figure 1. A PC Network Adapter is not required for the demonstration mode.

```
IBM PC Network Analysis Program 1.0

    MAIN FUNCTION SELECTION

1 - Network Status Monitor

2 - View or Convert Logs

3 - Detail Network Activity Monitor

4 - Directory Maintenance

5 - Local Adapter Information

6 - Change Operating Options

7 - Start or Stop Demonstration Mode

         SELECT ===> 1
```

Figure 1. Main Function Selection Menu

Program Highlights

The PCN Analysis Program has five major functions. You select them from the Main Function Selection menu.

- **Network Status Monitor**
  The major job of the PCN Analysis Program is to monitor the activity on the network. The Network Status Monitor does this by polling individual adapters on the network and receiving their status. Polling is done at set intervals. You can select the polling interval, as well as which adapters to poll.

The IBM PC Network Analysis Program provides a set of functions to help you manage your network.

You can review the status collected from the adapters on-line and optionally log it for later review. The status is also checked against thresholds you specify. When these thresholds are exceeded, the program generates notices and optionally logs the information. These notices may alert you that some action is necessary to correct an existing problem, or avoid a potential future problem. You can monitor values including packet error counts, packet collision counts, adapters powering on/off the network, and overall network utilization.

While monitoring, the Network Status Monitor adds less than five percent of the traffic on the IBM PC Network. The actual traffic amount added to the network is determined by the number of network adapters monitored and the polling interval.

- **View or Convert Logs**
  This function lets you process status logs and notices outside of the network status monitor. You can view logs, or convert the logs to an ASCII file and process them with your own programs. The logs are helpful in tracing network trends and tracking network performance.

The program diskettes contain a sample program written in BASIC. You can use it as a model

when developing your own programs. The *User's Guide* contains the formats of the logs, the converted logs, and the directory.

- **Detail Network Activity Monitor**
  The Detail Network Activity Monitor investigates the amount of data traffic between pairs of adapters, giving an indication of the distribution of load on the network. This is done by monitoring a subset of adapters on the network for whatever length of time you specify. Data traffic between this subset of adapters is then detailed in an on-line report.

  The information provided by the Detail Network Activity Monitor can help you balance your disk or print server workload. For example, you can trace the usage of a disk server by monitoring it and all of its users. That way you can identify the end user who place the greatest demand on a disk server. If the disk server becomes overloaded, you have the information you need to solve the overload problem.

- **Directory Maintenance**
  The PCN Analysis Program provides you with tools to create and maintain directories of the users of the network. Each entry in a directory contains the identification number of the adapter and reference name of the personal computer user. Optional fields are provided for additional information such as the location and telephone number of the personal computer user. The format of a directory entry is in the *User's Guide*. An example of the information in a directory entry is shown in Figure 2.

  Four directory functions are available. The Update Directory function lets you access individual directory entries. You can add new entries, or change or delete existing ones. The Change Directory Files function lets you specify the directory that the PCN Analysis Program uses. Multiple directories can be useful for customizing monitoring functions and for analyzing logs generated on different networks. The Build Directory function builds an initial directory. The PCN Analysis Program polls all the powered-on adapters on the network for their adapter ID's and creates entries for them in a new directory. The Create Directory Listing function generates a directory file in a printable format.

- **Local Adapter Information**
  The local network adapter is the one installed in the PC that is running the PCN Analysis Program. This function checks the optional settings of an installed network adapter, confirming the correct usage. Other information is presented, including the names of the users who are communicating with the adapter.

The information provided by the Detail Network Activity Monitor can help you balance your disk or print server workload.

A special diskette, the Local Adapter Information diskette, can be built to obtain the local adapter information on any PC. It contains only this function of the program. It can be run on any PC in the network, not just the computer which has the PCN Analysis Program installed.

The Local Adapter Information function can be used to help install new adapters on the network. You also can use it to check the operation of a suspected malfunctioning adapter. If the Local Adapter Information diskette is used, you can test the adapter without removing it.

Using the PCN Analysis Program - A Sample Scenario

The following is a simple scenario which describes how to use some of the features of the PCN Analysis Program. Other, more complete scenarios are in the *PCN Analysis Program User's Guide* and in the *PCN Analysis Program Technical Guide*.

This scenario assumes that the PCN Analysis Program has been installed and started. Installation and starting instructions are in the *User's Guide*.

The sample menus in this article are some of the actual screens the PCN Analysis Program displays when run in demonstration mode. The menus presented here can be followed while referring to the actual product.

Sam is the network operator. Users call him whenever a problem occurs. He has the Main Function Selection menu displayed on the personal computer at his desk (see Figure 1).

Sam receives a call from Julie Baker, stating that she is unable to send a message to the Department 53 disk. Sam first verifies that both Julie Baker and Department 53 are on the network. If he has printed the directory, he can refer to it. Otherwise, he can select option 4, Directory Maintenance. Within directory maintenance, he can use option 1, Update Directory, to verify that Julie Baker and Department 53 are contained in the directory.

After verifying the reference names of the two adapters involved with the inquiry, Sam returns to the Main Function Selection menu and selects option 1, Network Status Monitoring. This leads to a menu containing the following three options:

- **Start Monitoring**
  This option does the actual status monitoring.

- **Notice Options**
  This option allows Sam to customize the limits that errors are tested against. He can also specify the name of a file to contain the notice log.

- **Adapters Monitored and Logged**
  This option lets Sam select which adapters are to be monitored and which of the monitored adapters are to have status statistics saved in a status log.

Sam first selects option 3 to make sure that the adapters involved in the inquiry are monitored. The Adapters Monitored and Logged screen, shown in Figure 3, is displayed. Both "BAKER, JULIE" and "DEPT53" are selected for monitoring.

Sam is then ready to begin the Network Status Monitor. Within this option, the program will collect adapter network statistics by polling the adapters selected for monitoring at a regular interval that Sam specifies. Adapter statistics can be logged to a status log. If the program detects that a network or adapter error count has exceeded its permitted threshold, it generates a notice. Notices can be logged for later review.

After the initial sample period has elapsed, the Current Network Status menu is displayed, as shown in Figure 4.

```
                                                IBM PC Network Analysis Program 1.0
Name: BAKER, JULIE
                                                                  Entry:   2 of   17
                                UPDATE DIRECTORY
Directory: NDDEMODR.DIR                                          Adapter ID: 001678
================================================================================

  TAB to those fields you wish to change and
  type a new value.  When finished, press ENTER.


      REFERENCE NAME ===> BAKER, JULIE

      OPTIONAL DATA  ===> Phone: 2242
                     ===> Office: 2A36
                     ===> Cable 1, Line 1


================================================================================
ENTER=Change              F7=Previous Entry     Alt+F5=Delete Entry    ESC=Return
F6=Search for Entry       F8=Next Entry         Alt+F6=Add Entry
```

Figure 2. Update Directory Screen

This panel is the main status monitor menu. From this panel, Sam has an overview of the state of the network, primarily in the utilization percentage and the notice count. From here, he also can select one of the three report types to obtain more detailed information regarding the adapters selected for monitoring:

- **Status Log**
  The status log contains the adapter statistics gathered by the Network Status Monitor. Sam can

```
                                                IBM PC Network Analysis Program 1.0

                  ADAPTERS MONITORED AND LOGGED                  Entry:    1 of   17

Display Criteria: Monitor = ANY    Log = ANY
Logging Option: YES  File Name: NDDEMOST.LOG
================================================================================
TAB to the Monitor or Log Option you wish to change and press the F2 key.

     Reference Name                      Monitor      Log
     ANDERSON, LARRY.............. ===> YES  ===> YES
     BAKER, JULIE................. ===> YES  ===> YES
     DAVIS, SUSAN................. ===> YES  ===> YES
     DEPT22....................... ===> YES  ===> YES
     DEPT53....................... ===> YES  ===> YES
     FILESERVER1.................. ===> YES  ===> YES
     FILESERVER2.................. ===> YES  ===> YES
     FISHER, DAVE................. ===> YES  ===> YES
     GOLDEN, ROBERT............... ===> YES  ===> YES
     HATHAWAY, MARTIN............. ===> YES  ===> YES
================================================================================
F6=Search for Entry    Alt+F1=Logging Options    Alt+f2=Change Display Criteria
```

Figure 3. Adapters Monitored and Logged Screen

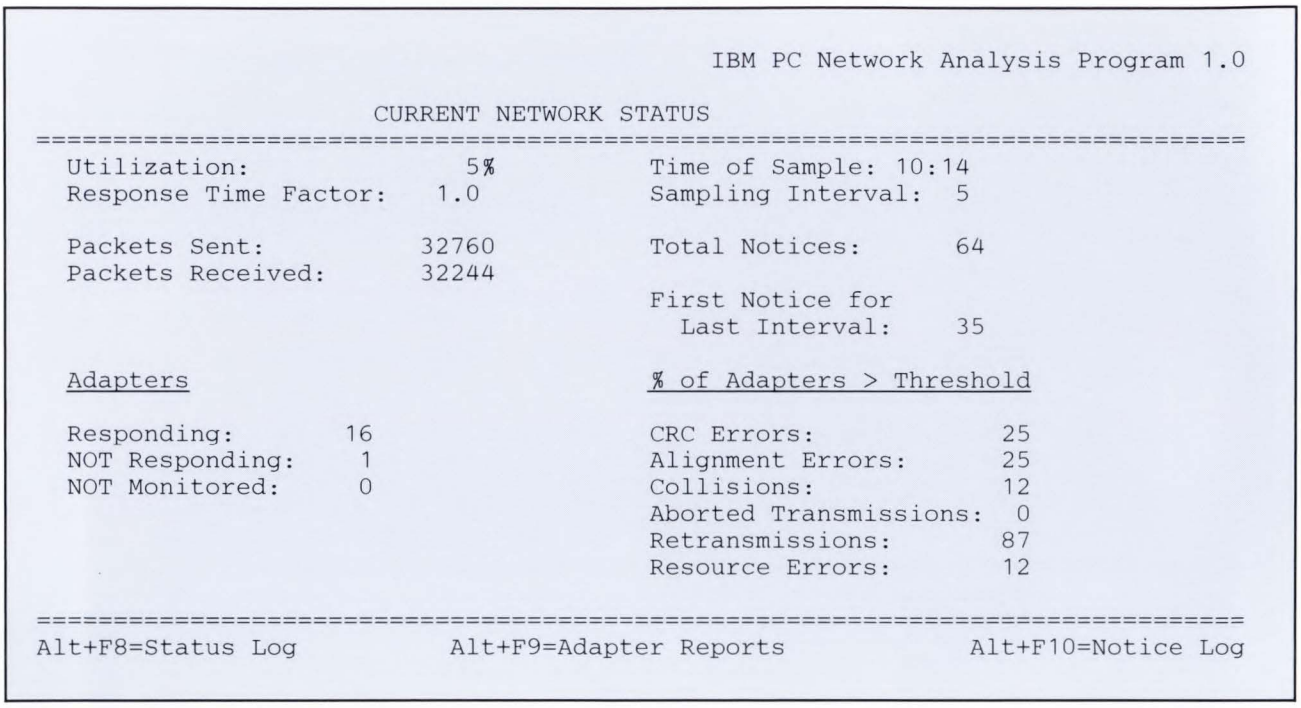


Figure 4. Current Network Status Screen

view the statistics for each individual adapter, sorted by time of occurrence. This log is useful in spotting trends, such as peak activity time or frequent downtime.

- **Adapter Reports**
  The adapter reports contain information related to the state of the network at the last time the Network Status Monitor polled the adapters. There are eleven different reports, each sorted by a significant network condition (such as an individual error rate).

- **Notice Log**
  The notice log contains a list of the notices generated by the Network Status Monitor. Notices are generated when the network is not performing within the limits specified by the network operator.

After checking the network utilization on the Current Network Status menu and seeing that the network has some traffic but is not too busy, Sam checks the state of the adapters he is interested in by viewing the first adapter report, All Adapters by Reference Name. This menu is shown in Figure 5.

Sam finds that the Department 53 adapter is responding, meaning that it answered the last poll request from the Network Status Monitor. He tabs to the line containing "DEPT53" and selects the Current Status function to find out more information about the adapter. The program polls the Department 53 adapter again and displays the Current Adapter Status Screen shown in Figure 6.

Sam sees that the adapter still appears to be working properly. He keys PgDn to look at the adapter's network name, DEPT53. Sam then calls Julie back to ask how she is trying to access the adapter. He discovers that the network name Julie is trying to use is DEPARTMENT53. He has her change the name to DEPT53 and send the message again. This time the message is sent successfully.

Since Julie Baker's problem has been solved, Sam exits the Network Status Monitor and returns to his other duties.

Other examples of network problems are simulated by the data presented in demonstration mode. These include an example of an overloaded server and an adapter turned off. Guidelines in solving these problems are described in the *PCN Network Analysis Program User's Guide*.

```
                                        IBM PC Network Analysis Program 1.0

                    ALL ADAPTERS BY REFERENCE NAME        Entry:    1 of   17

              Reference Name                          Status
================================================================================

              ANDERSON, LARRY.............     Responding
              BAKER, JULIE................     Responding
              DAVIS, SUSAN................     Responding
              DEPT22......................     Responding
              DEPT53......................     Responding
              FILESERVER1.................     Responding
              FILESERVER2.................     Responding
              FISHER, DAVE................     Not Responding
              GOLDEN, ROBERT..............     Responding
              HATHAWAY, MARTIN............     Responding
              JOHNSON, MARY...............     Responding
              MILLER, JOHN................     Responding
              THOMAS, BOB.................     Responding


================================================================================
F6=Search for Entry             F9=Last Status             F10=Current Status
```

Figure 5. All Adapters by Reference Name

```
(1.1.5)                                 IBM PC Network Analysis Program 1.0
Name: DEPT53
                            CURRENT ADAPTER STATUS              Screen 1 of 2
================================================================================

      Contact: Larry Anderson at 2120          Sample Time:       10:15
      Office: 2C10                             Time on Network:    1:14
      Cable 4, Line 2                          Status:       Responding

      CRC Errors:                 0            Adapter ROM Version:   01-23
      Alignment Errors:           1            Remote IPL:         Disabled
      Collisions:               101            Configured Sessions:      16
      Aborted Transmissions:      0            Configured Cmd Blks:       8
      Retransmissions:          528            Free Command Blocks:       5
      Resource Errors:            0            Max Packet Size:         750
                                               Number of Adapter Names:   1

      Packets Sent:            2532
      Packets Received:        2647

      Pending Sessions:           2


================================================================================
F10=Repeat Current Status
```

Figure 6. Current Adapter Status Screen

Software on the PC*jr*

Bob Kerns
Indiana IBM-PC Club

The PC*jr* with 128K and diskette drive will run a significant percentage of the commercial software made for the PC. The main restriction seems to be the memory space needed to run the application. If the PC*jr* is expanded beyond 128K, then perhaps 90-95% of the available commercial software will run. With expanded memory, the main restriction then becomes the location of the video memory and the logic of the software to find it on the PC*jr*.

The caveat for buying commercial software for the PC*jr* is the same as with any other computer: if possible, try it before you buy it.

One popular commercial program available today, Lotus 1-2-3, has a special PC*jr* version of the program using the cartridge system of the PC*jr*. This allows 1-2-3 to run on a 128K machine.

However, if you have 256K or more, you can run the diskette version as well. Everything will work fine except the graphics display on the screen. This is because the graphics board on the PC works differently than the one supplied on the PC*jr*. Lotus has made available a diskette which contains the drivers necessary to run graphics on the PC*jr*.

Another popular program is dBASE II. This program will run on the PC*jr* with no problems. If you have more memory than 128K, it may be advisable to put the program in RAM memory and use the diskette drive for data on the same diskette. If you are entering a lot of data over a long period of time, you may want to use the dBASE COLOR command. This allows you to change the screen colors. The command is:

```
SET COLOR TO n<foreground>,
             m<background>.
```

Changing colors every now and then helps prevent eye fatigue.

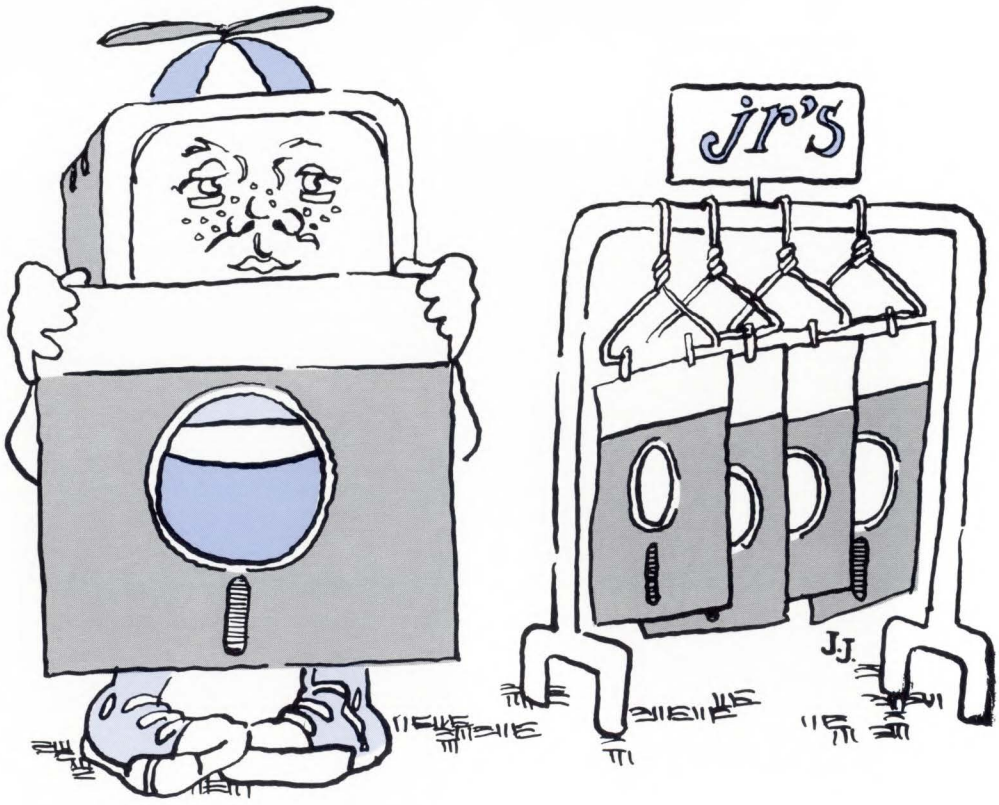
Multiplan is another popular spreadsheet that runs on the PC*jr*. Most of the commands are exactly the same for the PC*jr* as the PC. Any differences are noted in the screen help documentation.

If you enjoy games, the Flight Simulator runs very well on the PC*jr*. The program diskette contains a version just for the PC*jr*, and the documentation has a keyboard overlay for the PC*jr* keyboard. With the PC*jr*'s enhanced graphics, the 16-color mode may look a little better than on the PC.

Both word processors I've tried, Wordstar 3.3 and pfs: Write, work very well. These are only two of many word processors that will run on a PC*jr*.

There is one word of caution. If you work with both a PC and a PC*jr* using the same programs, you will notice that the applications run slower on the PC*jr*. This is because PC*jr* was not designed for a business environment and was not expected to perform the same job as a PC. The speed difference is small and does not interfere with the application. Be assured there is nothing wrong with the program or the computer if it takes a little longer to run on a PC*jr*.

(*Editor's note: For further information, please see* "Memory Intensive Programs on the PC*jr*" *in the January/February 1986 issue of* Exchange.)



Professional Graphics Software

John Warnock
IBM Corporation

Editor's note: This article is adapted from the IBM Personal Computer Seminar Proceedings.

IBM has a number of personal computer hardware and software products for the engineering and scientific communities.

The hardware products include:

- Professional Graphics Controller
- Professional Graphics Display
- Data Acquisition and Control (DAC) Adapter
- Data Acquisition and Control Adapter Distribution Panel
- General Purpose Interface Bus (GPIB) Adapter

To support these hardware products, IBM has a number of software products, including:

- Four professional graphics software development tools:
  - Graphical Kernel System (GKS)
  - Graphical File System (GFS)
  - Plotting System (PS)
  - Graphics Terminal Emulator
- A powerful programming language
  - Professional FORTRAN compiler
- Two data and laboratory information acquisition programs:
  - Data Acquisition and Control (DAC) Adapter Programming Support
  - General Purpose Interface Bus (GPIB) Adapter Programming Support

This article focuses on the graphics software products.

IBM Personal Computer Professional Graphics Software

The IBM PC graphics software products give application developers a fast, uniform method of creating graphics-based applications. This method uses a standard graphics interface that is consistent with graphics devices. Figure 1 shows the relationships of the four products listed above.

Note: Figure 1 does not imply that all four software products co-reside in memory.

IBM Professional Graphics Product Summary

The IBM Engineering/Scientific Series of graphics products has adopted the proposed graphics standards of the International Standards Organization (ISO) and the American National Standards Institute (ANSI). These standards allow graphics applications to support all hardware devices that accept the standards.

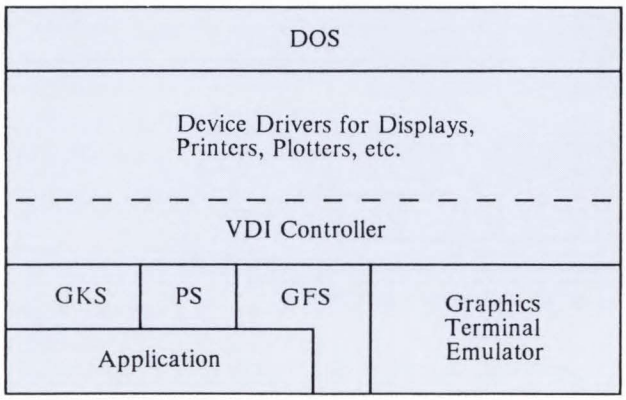
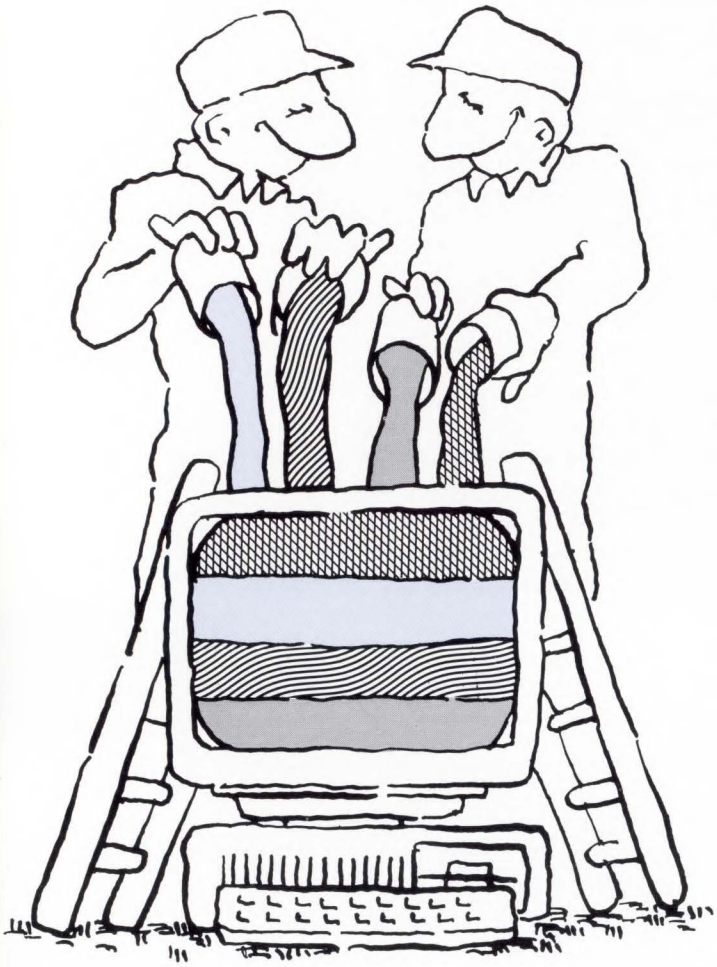


Figure 1. Relationship of PC Graphics Software

Thus, application programmers need not be concerned with specific hardware, and can write software that is easily migrated to other devices.

The **Virtual Device Interface** (VDI) is the foundation for all of the IBM PC professional graphics software products. The VDI includes a VDI Controller that follows the proposed ANSI X3H3 definition. The graphics software products use the VDI and its device drivers to communicate with a variety of displays, printers, plotters, etc. Should you need to, you can write your own device drivers for use with the VDI.

The **Graphical Kernel System** (GKS) is an implementation of the proposed ISO/ANSI GKS standard. It includes graphics functions that can be called by several popular languages.



The **Graphical File System** follows the proposed ANSI Virtual Device Metafile (VDM) standard for storing, retrieving and manipulating graphic images. It provides a program interface and an interactive interface for application developers.

The **Plotting System** is a VDI-based product for developing presentation graphics. It is a library of common charting and plotting routines.

The **Graphics Terminal Emulator** lets a personal computer emulate one of several popular graphics display terminals. It allows communication with a host computer.

Graphical Kernel System

The IBM PC Graphical Kernel System provides a set of functions that the majority of applications can use to produce computer-generated pictures.

The Graphical Kernel System is based on the Virtual Device Interface and benefits from the VDI's versatility. The GKS uses built-in device attributes of the VDI Controller and device drivers.

The Graphical Kernel System is based on proposed ISO/ANSI standards intended to aid graphics application programmers in understanding and using graphics methods. The resulting programs conform to ISO/ANSI standards and can run on any system that supports the GKS standards. Also, the standards give device manufacturers guidance about useful graphics capabilities.

Highlights

The Graphical Kernel System provides program portability between computer systems that support the GKS standard. Portability is accomplished by providing a consistent interface in high-level languages.

The Graphical Kernel System is designed to:

- Provide device independence to graphics applications.
- Conform to ANSI implementation with full 2B segmentation features of the GKS standard.
- Support segmentation functions. (A segment is a collection of graphics primitives that can be dealt with as a unit through a range of graphics manipulations, including transformation scaling, highlighting, visibility, and detectability.)
- Include a rich set of inquiry functions.
- Provide five different classes of input functions: locator, string, pick, choice, and valuator.
- Support 2D graphics primitives.
- Provide the following language bindings:
  - IBM PC BASIC Compiler
  - IBM PC FORTRAN 2.00
  - IBM PC Professional FORTRAN
  - Lattice C compiler by Lattice Corporation

- Support transformation between device coordinate, normalized device coordinate and world coordinate systems.

Workstations

In addition, the Graphical Kernel System supports four categories of workstations:

OUTPUT An example is a printer.

INPUT An example is a joystick.

OUTIN (Output and Input). An example is a display with a keyboard used to output graphical images and input keystrokes.

WISS (Workstation-Independent Segment Storage). WISS is a virtual device that allows segments to be stored independent of any particular physical workstation.

The IBM PC graphics software products give application developers a fast, uniform method of creating graphics-based applications.

GKS Routines

According to the tasks they perform, GKS routines are separated into nine categories:

- Control routines
- Output routines
- Attribute routines
- Transformation routines
- Segment routines
- Input routines
- Inquiry routines
- Utility routines
- Error handling and logging routines

A discussion of each of these nine routines follows.

- **Control** routines affect the state of the system, or the state of the workstation. These routines include:
  - Initializing and terminating the system.
  - Opening, activating, deactivating, closing, clearing and updating the workstation.
  - Redrawing all segments on the workstation and escaping. The escape routine lets you issue commands directly to the device, so you can take advantage of any non-standard features available on your particular graphics device.
- **Output** routines display the basic and generalized graphics primitives listed above.
- **Attribute** routines modify the appearance of the basic graphics primitives. The modifiable attributes include:

For this Primitive	These Attributes Can be Modified
Polyline	Color Width scale factor Type (minimum of six types of polylines)
Polymarker	Color Size scale factor Type (minimum of six types of polymarkers)
Fill Area	Color interior styles (hollow, solid, pattern, hatch) Style index, an index into the pattern or hatch table (minimum of six styles)
Text	Color font, string precision and character precision Character height Character up vector, the direction of the text string Alignment, including three horizontal and three vertical justifications

The Graphical Kernel System does not provide attributes for generalized graphics primitives. Each generalized graphics primitive assumes the current attributes of the basic GKS primitive it most closely resembles. For example, arcs use current polyline attributes; bars, pie slices, and circles use fill area attributes.

The number of available choices for an attribute is sometimes determined by the specific capability of a device. For example, the number of available text fonts is determined by how many hardware

fonts the device supports, and the number of available polyline colors is determined by how many colors the device supports.

- **Transformation** routines perform mapping between the three coordinate systems available in GKS.

  1. The world coordinate (WC) plane is a user-defined Cartesian coordinate system. You build your graphics image in world coordinates, then define the range of world coordinates with the Set Window transformation routine.

  2. The normalized device coordinates (NDC) plane is a standardized virtual plane that provides a uniform coordinate system for all workstations. You define the range of normalized device coordinates with the Set Viewpoint and Set Workstation Window routines.

  3. The device coordinate (DC) plane is the portion of the device surface that will be used to output graphics. You set the range of device coordinates with the Set Workstations Viewpoint routine.

  Normalization transformation maps WC to NDC, and workstation transformation maps NDC to DC. Eight normalization transformation numbers can be defined at one time in GKS. The Select Transformation Number routine is used to select the effective transformation number.

The Graphical Kernel System provides program portability between computer systems that support the GKS standard.

By default, the normalization transformation number 0 maps WC (0.0, 1.0) x (0.0, 1.0) to NDC (0.0, 1.0) x (0.0, 1.0). The default workstation transformation maps NDC (0.0, 1.0) x (0.0, 1.0) to DC using the largest square that fits on the device surface.

- **Segment** routines are primitives grouped together so you can operate on them as a single object. To create a segment, you call the Create Segment routine, then call the output routines that create the primitives for the segment. You call the Close Segment routine to define the end of the current segment.

  When a segment is created, it is automatically stored in all output workstations that are active. To store a segment into a workstation, the associate segment with workstation routines can be used.

  The Delete Segment routine removes the segment from all workstations. The Delete Segment From a Workstation routine is limited to a single workstation.

  The attribute routines can modify the following attributes of segments:

  Visibility
  Detectability
  Highlighting
  Segment priority
  Pick identifier

  Any existing segment can be transformed by the Set Segment transformation routine. For input parameters, this routine accepts fixed point, shift vector, rotation angle and scaling factors. For the output parameter, you specify a transformation matrix. The routine then calls on the Evaluate Segment Transformation routine to calculate this matrix.

- **Input** routines support five input classes for interactive graphics: locator, choice, string, pick and valuator.

  – With the **locator routine**, you move a graphics cursor on the screen to select an input position. The routine returns the value of the point in world coordinates, together with a normalization transformation number.

  – The **choice routine** usually asks you to choose among fixed alternatives by pressing a button or function key on a device. For example, the keyboard function keys act as a simulated device. By pressing the keys, you cause the routine to return a non-negative integer value that represents a selection from a number of choices.

- The **string routine** lets you enter a character string into your program. The character string typically comes from the keyboard.

- The **pick routine** moves the graphics input cursor over the display screen to pick a segment on the display surface. It returns a segment name, pick identifier and the status.

- The **valuator** is a routine used to enter a real number value. It sets a valuator device. For example, turning a dial to the position which represents the value you want.

Each of the five input routines can operate in one of two modes: request mode or sample mode.

1. In **request mode**, execution of the program is held until you respond with a request mode trigger. For example, when a locator function is invoked in request mode, the program stops until you move the cursor to the desired area on the screen and terminate the request. You may terminate the request by pressing the Enter key on the keyboard or by pressing the corresponding joystick buttons.

2. In **sample mode**, program execution is not held. Input routines under this mode return the values that are the current measure of the physical workstation at the time the routines are called. For example, the sample mode of the locator routine returns the current position of the graphics cursor at the time the routine is called.

- The Graphical Kernel System provides an extensive set of **inquiry** and **utility** routines as well. Using the inquiry routines, your application programs can ask for the current operating state, workstation information, current setting of the primitive attributes, segment attributes and device capability. The utility routines make it more convenient to compute transformation matrices and to handle packed data records.

- Finally, GKS includes **error handling** and **error logging** routines. For each routine listed above, a finite number of error situations is defined, and can be classed accordingly:

  - Class 1 errors result in a precisely defined reaction. For example, when the viewport rectangle set is invalid, GKS reacts by logging an error message in the error file of the application program and ignoring the function call.

  - Class 2 errors result in an attempt to save the results of previous operations. When a memory error occurs, or a workstation cannot be opened, GKS reacts by displaying the error message on the console and transferring control to the operating system console.

  - Class 3 errors are those that cause unpredictable results including the loss of information or control. A hardware failure is an example.

    During processing, GKS is always in one of the following five operating states:

    — Kernel system closed
    — Kernel system open
    — At least one workstation open
    — At least one workstation closed
    — Segment open

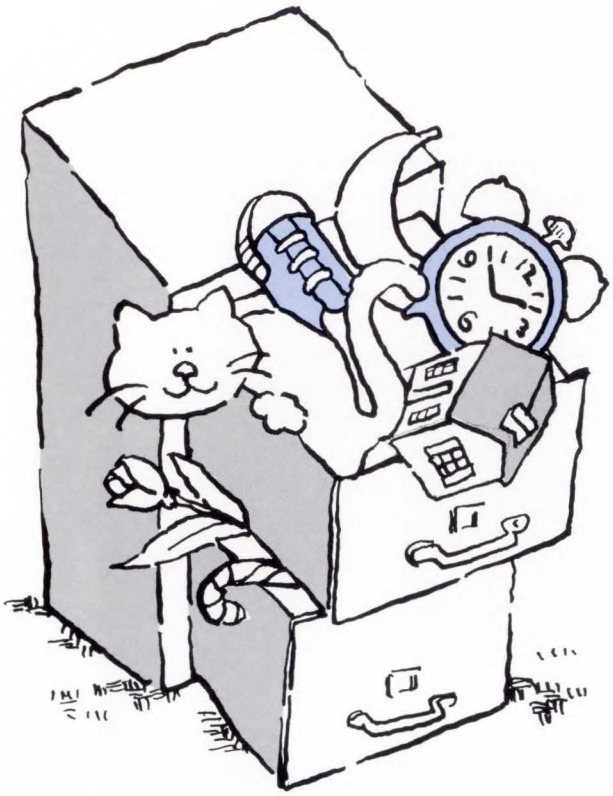
Each Graphical Kernel System routine requires the system to be in a certain operating state. If you make a call in the wrong operating state, you will receive a "Kernel System not in proper state" error message.

Fundamental to the Virtual Device Metafile notion is the ability to conveniently archive graphic images...

Graphical File System

The IBM PC Graphical File System is an implementation of the Virtual Device Metafile (VDM). Fundamental to the Virtual Device Metafile notion is the ability to conveniently archive graphic images on commonly available storage media.

A metafile consists of one or more pictures. Each picture has its own header and trailer data, data specific to the picture itself and a set of metafile elements or instructions made up of operation codes and parameters. The structure and definition of metafile elements are consistent with the proposed ANSI X3H33 standard.



The metafile is created by an appropriate device driver, operating under the Virtual Device Interface (VDI), which translates the graphics commands issued by an application program into the metafile instructions.

To recreate a picture, a metafile interpreter program reads the file, interprets the elements and reissues the VDI commands for execution by the output device driver. Output can be directed to a printer, plotter or any supported display, no matter what display was originally used.

Output also can be directed to a metafile to save as a new picture. Thus, a metafile containing separate pictures of three ships can be combined to show all three ships in one picture and the resulting picture can be saved.

Because pictures are stored in a device-independent manner, they can be ported to and shared with other computers and installations without recalculation. All that is needed is a metafile interpreter to read the file to the application.

Images can be composed from existing metafile images and some basic editing may be performed during the composition process.

Metafile Encoding

There are six classes of elements in a metafile:

- Descriptor
- Control
- Picture
- Graphical
- Attribute
- Escape

Depending on the number of parameters, the elements are characterized as short or long. The first 11 bits define the operation code, with bits 5 through 11 defining the operation ID. Bits 12 through 15 specify the class.

The two basic parameter types in a metafile are string and 16-bit integers. String types consist of a count of ASCII characters followed by the encoded list of one-byte ASCII characters. The character count is one byte in short form or a 16-bit integer in long form. The 16-bit parameter type is an integer that can handle data ranging from -32768 to 32767. All parameters end on 16-bit boundaries and strings are padded to even byte boundaries where necessary.

The programming interface lets you interrogate each element of the picture as it is read from the metafile. You can take action based on the class and ID.

The IBM Graphical File Programmer's/User's Guide provides a detailed description of the operation codes and their meaning.

Interfaces

The Graphical File System offers two user interfaces:

- Interactive interface
- Programming interface

The interactive interface lets you communicate with the Graphical File System through an interactive, icon-driven screen display. This interface provides the environment for:

- Composing and viewing an image. You can select the area of the screen for viewing and, if desired, combine multiple images on one screen.
- Creating a new metafile picture for later viewing.
- Directing the image to an output device such as a plotter or printer.

Other icons let you specify the size of the displayed image and erase components of an image.

The programming interface functions enable you to write programs that control and interpret metafiles. You can perform some rudimentary editing or even create your own metafile interpreter. A simple example is to create a presentation from a set of metafile pictures that were created separately by different applications.

Because pictures are stored in a device-independent manner, they can be ported to and shared with other computers...

These functions are available as a subroutine package to application programmers who create their programs using the following language compilers:

Compiler	Library Name
FORTRAN Version 2.00	FORMETA.LIB
Professional FORTRAN	PFMETA.LIB
Lattice C Version 2.00	CMETA.LIB
BASIC Compiler	BASMETA.LIB and MHEAP.OBJ

The language bindings provide 15 functions in the following categories:

1. Initialization and termination, consisting of four functions:
   - The Open Metafile function makes a specified metafile available for reading and returns an integer identifier to be used with other function calls.
   - The Close Metafile function is used when interpretation is complete.
   - The Open Workstation function prepares a workstation to receive graphic output, including clearing the display surface. The workstation is specified by using one of the logical workstation names ("DISPLAY", "PLOTTER", etc.), and the function returns an integer identifier (device handle) to be used on subsequent calls.
   - The Close Workstation function is the converse of the Open Workstation function.
2. Picture control.
   Once the metafile is opened, and prior to interpretation, an individual picture must be selected using the functions:
   - Open Picture, which identifies a specific picture by its integer.
   - Close Picture, which indicates that interpretation is complete.
3. Interpretation.
   Two classes of functions are available for interpreting:
   - The Interpret Picture function identifies a complete picture to be interpreted and also specifies the output device on which the picture should appear.
   - This class consists of three functions that let you proceed through the metafile item-by-item (or element-by-element) starting at a given picture:
     - Inquire Metafile Item (Element) Length function, which returns the length (in bytes) of the next element in the metafile before it is loaded into a buffer. (This is necessary because metafile elements vary greatly in length.)
     - Get Metafile Item function, which reads the next element into the buffer.
     - Interpret Metafile Item function, which interprets the element in the buffer and directs it to the specified output device.
4. Workstation metafile control, performed by the Clear Workstation function, which:
   - Clears CRT devices.
   - Displays all pending graphics on plotters and printers.
   - Prompts for new paper on plotters; advances to top-of-form on printers.
   - Initiates a new picture if a metafile is specified.

5. Cutting and pasting. Pictures are defined in a virtual device coordinate (VDC) space of 0-32767 on the x and y axes. They are portrayed on a viewing surface with device-dependent dimensions restricted to 0-32767 on each axis. (Thus the Color Graphics Adapter is 32767 by 22500.)

   The Set Window function in the Graphical File System lets you "cut" part of a picture by specifying a rectangular area or window in the VDC space. Using the Set Viewport function, this "cut" can be "pasted" to any rectangular area on the display surface. These functions provide the basic mechanism to superimpose several pictures on one image.

6. Error and Status Detection. You can determine the cause of an error by using the Inquire Metafile Error function. (The error codes are listed in appendix A, of the *IBM Graphical File System Programmer's/Users Guide*.) Also, the Inquire Metafile Version function identifies the current version and level of the Graphical File System.

APL Graphics Support, a Mystery Story

Anne Ross
Poughkeepsie IBM Microcomputer Club

When an APL program runs, output is usually written to the screen sequentially. When the screen is full, the information is scrolled up and the new line placed at the bottom.

It is also possible to display output in certain predefined areas of the screen called windows. In this case there is no scrolling. The output can be in the form of characters or images formed of individual dots (pixels). It may be in color and it is possible to read input from the screen as well as write output to it.

Interactive use of the screen as implemented in APL 1.0 is the topic of this article. Getting it to work has been quite an adventure! My first attempt to write characters in a screen window was successful but flashed on and off the screen so fast I didn't see it. A delay statement was needed.

The program SETSCR shows in Figure 1 how these problems were solved. The first section of the program shows the creation and use of a window in character mode. Lines 10 to 14 establish communication between an APL-defined function and the AP205 full-screen processor. Line 16 clears the screen and sets it to 40-character color mode. Lines 19 and 24 define windows and establish the new screen format. Then lines 25 to 30 write a vextor of characters to the window.

Next I decided to draw a graph using pixel graphics in high resolution mode. The first point of confusion was that the window is defined in terms of the number of characters it can contain, whereas the graphics image is formed of 8-by-8 pixels in each character position. In addition, the size of the window is limited by the fact that no APL object can have more than 32,767 elements. The definition of this window is shown on lines 32 and 37. A boolean vector is written to this window on lines 38 thru 41.

Printing APL Programs

John Warnock
IBM Corporation

The program listing in this article presented an interesting challenge: how to list a program written in APL and maintain quality output. Personal Computer APL supports the IBM Graphics Printer which is fine for daily use, but not for typesetting.

We solved this problem by using our IBM 3812 Pageprinter. Our 3812 Pageprinter has two APL fonts: a 10-pitch font and a 20-pitch font.
For these fonts, we printed a chart showing the ASCII values for the special APL characters. Next, using Personal Editor, we keyed in the programs substituting the appropriate ASCII values for special symbols. We then preceded the listing with an instruction to load the 10-pitch APL font, and to use character set 2. Finally, we sent the file to the printer.

```
      SETSCR[0]
     ∇SETSCR:T;RC;GRAF;C;D;FA
[1]   ⍝=================================================================
[2]   ⍝
[3]   ⍝   PROGRAMMER/DATE :   ANNE ROSS    8/6/85
[4]   ⍝
[5]   ⍝    SYSTEM        :   IBM PC   IBM APL    DOS 2.0
[6]   ⍝
[7]   ⍝=================================================================
[8]   ⍝ A REQUEST TO SHARE VARIABLES WITH AP205, THE FULL SCREEN AUXILIARY
[9]   ⍝ PROCESSOR.  SEE APL REFERENCE BOOK CHAPTER 3 PAGES 10 - 20
[10]   RC←205 ⎕SVO 2 1⍴'CD'
[11]   ⍎(∨/RC≠1 1)/'→OUT,(T←⎕DL 10),0⍴⎕←''OFFER FAILED'''
[12]  ⍝ A REQUEST TO SHARE VARIABLES C & D.
[13]   RC←⎕SVO 2 1⍴'CD'
[14]   ⍎(∨/RC≠2 2)/'→OUT,(T←⎕DL 10),0⍴⎕←''C & D NOT SHARED'''
[15]  ⍝ SET THE SCREEN TO 40 CHARACTER COLOR MODE VIA THE CONTROL VARIABLE.
[16]   C←0 4
[17]   ⍎(0≠RC←C)/'→OUT,(T←⎕DL 10),0⍴⎕←''SCREEN NOT SET TO COLOR 40'''
[18]  ⍝ DEFINE A WINDOW ON THE SCREEN.
[19]   FA←2 2 1 30 2 2
[20]  ⍝ ASSIGN A MATRIX OF FIELD DEFINITIONS TO THE DATA VARIABLE.
[21]   D←FA←1 6⍴FA
[22]  ⍝ ESTABLISH NEW SCREEN FORMAT.
[23]   C←1
[24]   ⍎(0≠RC←C)/'→OUT,(T←⎕DL 10),0⍴⎕←''FIELD DEFINITION N.G.'''
[25]   CHAR←'TEXT IN A WINDOW                '
[26]  ⍝ ASSIGN CHARACTER DATA VECTOR TO D.
[27]   D←CHAR
[28]  ⍝ WRITE CHARACTERS TO THE SCREEN.
[29]   C←2 1
[30]   ⍎(0≠RC←)/'→OUT,(T←⎕DL 10),0⍴⎕←''FIELD 1 N.G.'''
[31]   T←⎕DL 20
[32]   C←0 8
[33]   ⍎(0≠RC←C)/'→OUT,(T←⎕DL 10),0⍴⎕←''SCREEN NOT SET TO HI RES'''
[34]   FA←2 2 15 30 2 2
[35]   D←FA←1 6⍴FA
[36]   C←1
[37]   ⍎(0≠RC←C)/'→OUT,(T←⎕DL 10),0⍴⎕←''FIELD DEFINITION N.G.'''
[38]   GRAF←(240×64)⍴(240⍴1),((240×7)⍴0)
[39]   D←GRAF
[40]   C←2 1 1
[41]   ⍎(0≠RC←C)/'→OUT,(T←⎕DL 10),0⍴⎕←''GRAPHICS DATA N.G.'''
[42]   T←⎕DL 20
[43]   C←0 4
[44]   ⍎(0≠RC←C)/'→OUT,(T←⎕DL 10),0⍴⎕←''SCREEN NOT SET TO LO RES'''
[45]   FA←4 4 8 30 2 1
[46]   D←FA←1 6⍴FA
[47]   C←1
[48]   ⍎(0≠RC←C)/'→OUT,(T←⎕DL 10),0⍴⎕←''FIELD DEFINITION N.G.'''
[49]   GRAF←(240×64)⍴(240⍴1 0),((240×7)⍴0),(240⍴0 1),(240×7)⍴0
[50]   D←GRAF
[51]   C←2 1 1
[52]   ⍎(0≠RC←C)/'→OUT,(T←⎕DL 10),0⍴⎕←''GRAPHICS DATA N.G.'''
[53]   T←⎕DL 20
[54]  OUT:
[55]   0←0 8
[56]   RC←C
[57]   RC←⎕SVR 2 1⍴'CD'
[58]   RC←⎕EX 2 1⍴'CD'
```

Figure 1. Sample APL Program

Device-Independent Graphics for PCs

Jim Glass
Rocketdyne Microcomputer Users Group

Device independence means that a graphics system is programmed to handle the nitty details of pixel addressing, aspect ratio, etc. so the user does not have to know the exact characteristics of the hardware that will draw the graphics. By writing a device-independent graphics application, you write the program once and can then use it on a wide range of devices (screens, plotters, printers, etc.) without having to recode it.

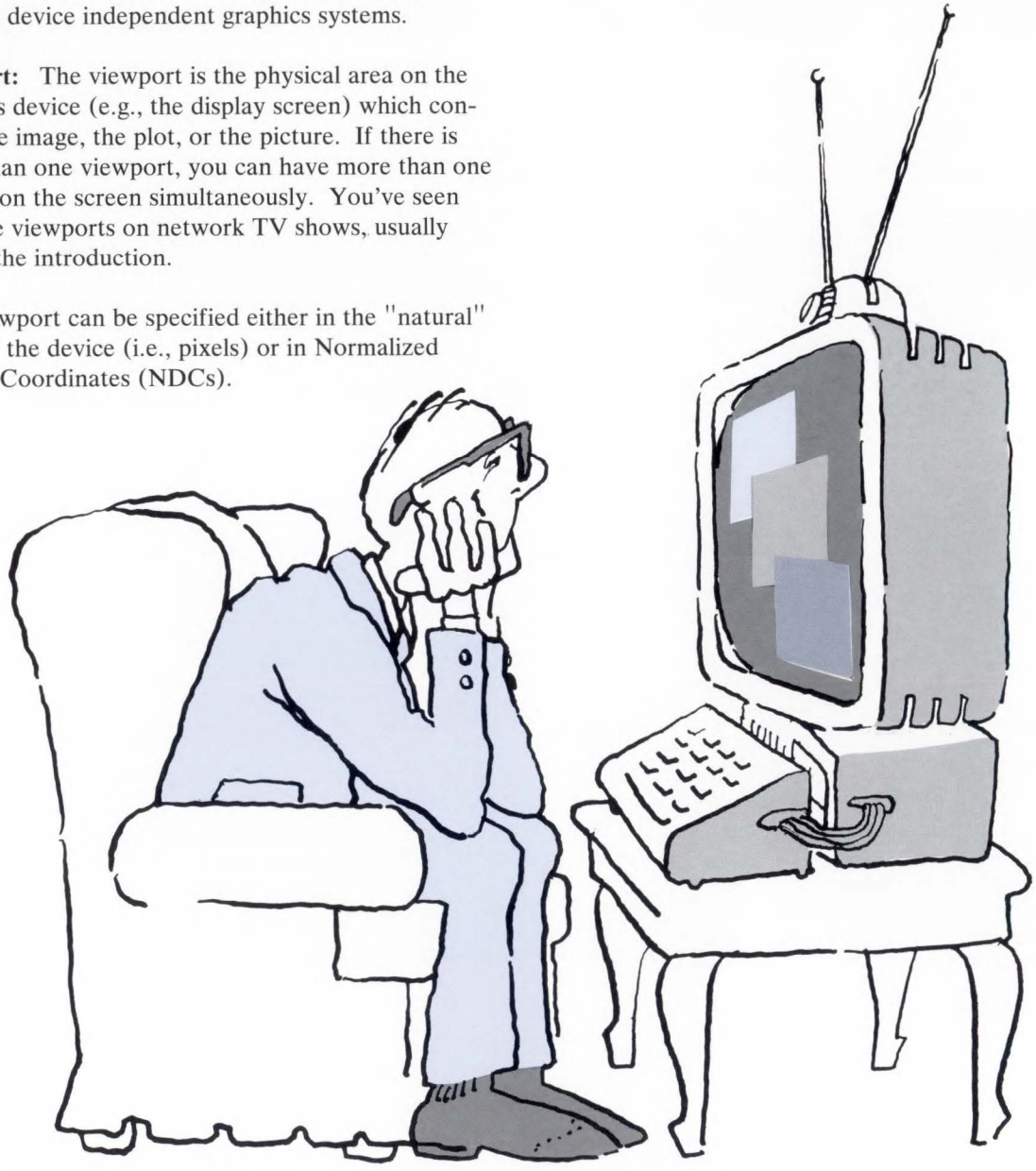
By programming applications with two concepts in mind, **window** and **viewport**, you can more easily develop device independent graphics systems.

Viewport: The viewport is the physical area on the graphics device (e.g., the display screen) which contains the image, the plot, or the picture. If there is more than one viewport, you can have more than one picture on the screen simultaneously. You've seen multiple viewports on network TV shows, usually during the introduction.

The viewport can be specified either in the "natural" units of the device (i.e., pixels) or in Normalized Device Coordinates (NDCs).

In NDC units, every device has a coordinate space that covers the range from 0 to 1 in both the X and Y axes. This means that all possible displayable points have X and Y coordinates less than one. Because there are an infinite number of points in the sub-plane bounded by {(0,0) (0,1) (1,0) (1,1)}, the entire (plane) universe can be mapped onto the NDC space.

This approach requires that an additional mapping be performed from NDCs to actual hardware pixels. Tektronix systems use GDUs (Graphic Device Units), which can be thought of as NDCs that do not use 1.0 as the limiting value of the data range. For instance, on many Tektronix systems, the GDU range is 0 to 130 in the X direction and 0 to 100 in the Y direction. This makes the aspect ratio of the display explicit. Given the ranges mentioned above, the apsect ratio would be 1.3.



Window: The window defines the range of values in X and Y that your plot or image will cover. Sometimes the window is called world space or world coordinates. Tektronix calls the units of the window User Data Units (UDUs).

If you define a viewport and window limits, you automatically define a mapping from your world onto the graphics device. You will be providing enough information for any competent graphics system to transform coordinates from your units onto the screen or other display.

Clipping

This feature (in most graphics systems) prevents segments of the image which fall outside the window from being displayed. Clipping is a very valuable feature, because together with the window it permits zooming in or out of an image to see more or less detail.

You can explore the use of device-independent graphics if you have an IBM PC with BASIC version 2.00 or later. BASIC 2.00 or later includes a WINDOW command and a VIEWport command.

If you're curious about how the graphic system (or BASIC) performs the mapping from your units onto the screen, you might try to derive the transformation equations yourself. If not, I'll present them here.

User Units to Device Coordinates

Let's assume that the graphics system has been initialized, and that it contains the correct values of the (internal) variables as shown in Figure 1.

Using these variables, the transformations (in FORTRAN) from user units into device coordinates are:

```
X=1.+((XSCR-1.)/XGDU)*((X0-WIND(1))*
(VIEW(2)-VIEW(1))/ & (WIND(2)-
WIND(1)))+VIEW(1)*XSCR/XGDU
```

and either

```
Y=YSCR-((YSCR-1.)/YGDU)*((Y0-WIND(3))*
(VIEW(4)-VIEW(3))/ & (WIND(4)-
WIND(3)))-VIEW(3)*YSCR/YGDU
```

or

```
Y=1.+((YSCR-1.)/YGDU)*((Y0-WIND(3))*
(VIEW(4)-VIEW(3))/ & (WIND(4)-
WIND(3)))+VIEW(3)*YSCR/YGDU
```

where X0 is the desired X coordinate in user units and Y0 is the desired Y coordinate in user units, both multiplied by the scale factor SCALE.

The reason for two Y transformations is that the first one assumes an "upside-down" device such as the IBM graphics screen, where the (1,1) pixel is at top left. The second assumes a "right side up" device such as a plotter.

The graphics system also should maintain internal variables LASTX, LASTX0, LASTY and LASTY0. These REAL variables hold the previous values of the transformed and untransformed coordinates. This information is used to perform relative (as opposed to absolute) point addressing.

Note: The transformations above do not perform clipping. The system must perform both hard and soft clipping. Hard clipping assures that pixel or device coordinates are never generated beyond the physical limits of the device:

```
X=AMIN1(X,XSCR) X=AMAX1(1.,X)
```

Soft clipping clips vectors against the window limits, assuring that vectors which extend beyond the window limits are drawn only to the edge of the window, and that vectors which lie entirely outside the window are not drawn at all.

A simple CLIP routine is shown in the following page. It assumes all variables are in COMMON. The flag ERRFLG, when true, causes other graphics routines to return without taking any action, so that the offending segment is not drawn. The X0 and Y0 variables again contain the current user coordinates to which a vector is to be drawn, or to which the graphic point is to be moved. For fans of structured code, the reason for all the GOTOs is that this routine was translated more or less verbatim from BASIC.

XGDU	the number of GDUs in the X direction
YGDU	the number of GDUs in the Y direction
XSCR	the number of pixels (or device coordinates) in the X direction
YSCR	the number of pixels (or device coordinates) in the Y direction
SCALE	the device aspect ratio
WIND(1)	the minimum X value of the user's data space (i.e., the lower X limit of the window)
WIND(2)	the maximum X value of the user's data space (i.e., the upper X limit of the window)
WIND(3)	the minimum Y value of the user's data space (i.e., the lower Y limit of the window)
WIND(4)	the maximum Y value of the user's data space (i.e., the upper Y limit of the window)
VIEW(1)	the minimum X value of the viewport, in GDUs (i.e., the lower X limit of the viewport)
VIEW(2)	the maximum X value of the viewport, in GDUs (i.e., the upper X limit of the viewport)
VIEW(3)	the minimum Y value of the viewport, in GDUs (i.e., the lower Y limit of the viewport)
VIEW(4)	the minimum Y value of the viewport, in GDUs (i.e., the upper Y limit of the viewport)

Figure 1. Variable Listing

```
      SUBROUTINE CLIP
$INCLUDE: 'COMMON.FOR'
      LOGICAL CFL
      DOUBLE PRECISION XM
      ERRFLG=.FALSE.
      ERRFLG=( X0 .LT. WIND(1) .AND.
     &  LASTX0 .LT. WIND(1))
      ERRFLG=((X0 .GT. WIND(2) .AND.
     &  LASTX0 .GT. WIND(2)) .OR. ERRFLG)
      ERRFLG=((Y0 .LT. WIND(3) .AND.
     &  LASTY0 .LT. WIND(3)) .OR. ERRFLG)
      ERRFLG=((Y0 .GT. WIND(4) .AND.
     &  LASTY0 .GT. WIND(4)) .OR. ERRFLG)
      YP=Y0
      XP=X0
      IF (ERRFLG .OR. .NOT. CLPFLG)
     &  GOTO 4225
      IF (X0 .NE. LASTX0) XM=
     &  (Y0-LASTY0)/(X0-LASTX0)
      IF (X0 .EQ. LASTX0) XM=
     &  ABS(WIND(4)-WIND(3))/1.E-30
      IF (XM .EQ. 0.D0) XM=
     &  ABS(WIND(2)-WIND(1))/1.E30
      IF (X0 .GE. WIND(1)) GOTO 4185
      X0=WIND(1)
      Y0=XM+(WIND(1)-LASTX0)+LASTY0
      CALL CHCKY0
      IF (ERRFLG) GOTO 4225
4185  IF (X0 .LE. WIND(2)) GOTO 4189
      X0=WIND(2)
      Y0=XM+(WIND(2)-LASTX0)+LASTY0
      CALL CHCKY0
      IF (ERRFLG) GOTO 4225
4189  IF (Y0 .GE. WIND(3)) GOTO 4192
      Y0=WIND(3)
      X0=(WIND(3)-LASTY0)/XM+LASTX0
      CALL CHCKX0
      IF (ERRFLG) GOTO 4225
4192  IF (Y0 .LE. WIND(4)) GOTO 4200
      Y0=WIND(4)
      X0=(WIND(4)-LASTY0)/XM+LASTX0
      CALL CHCKX0
      IF (ERRFLG) GOTO 4225
4200  CFL=.FALSE.
      IF (LASTX0 .GE. WIND(1)) GOTO 4205
      CFL=.TRUE.
      LASTY0=LASTY0+XM*(WIND(1)-LASTX0)
      LASTX0=WIND(1)
```

```
4205  IF (LASTX0 .LE. WIND(2)) GOTO 4209
      CFL=.TRUE.
      LASTY0=LASTY0+XM+(WIND(2)-LASTX0)
      LASTX0=WIND(1)
4209  IF (LASTY0 .GE. WIND(3)) GOTO 4213
      CFL=.TRUE.
      LASTX0=LASTX0+(WIND(3)-LASTY0)/XM
      LASTY0=WIND(3)
4213  IF (LASTY0 .LE. WIND(4)) GOTO 4217
      CFL=.TRUE.
      LASTX0=LASTX0+(WIND(4)-LASTY0)/XM
      LASTY0=WIND(4)
4217  CONTINUE
      IF (.NOT. CFL) GOTO 4225
      X=1.+((XSCR-1.)/XGDU)*((LASTX0-
     &  WIND(1))*(VIEW(2)-VIEW(1))/
     &  (WIND(2)-WIND(2)))+VIEW(1)*
     &  XSCR/XGDU
      X=AMIN1(X,XSCR)
      X=AMAX1(X,1.)
      Y=YSCR-1.)/YGDU)*((LASTY0-WIND(3))*
     &  (VIEW(4)-VIEW(3))/
     &  (WIND(4)-WIND(3)))-VIEW(3)*
     &  YSCR/YGDU
      Y=AMIN1(Y,YSCR)
      Y=AMAX1(Y,1.)
      I=X
      J=Y
4225  LASTY0=UP
      LASTX0=XP
      RETURN
      END
      SUBROUTINE CHCKY0
$INCLUDE: 'COMMON.FOR'
      IF (LASTY0 .GE. WIND(3) .AND.
     &  LASTY0 .LE. WIND(4)) RETURN
      IF (Y0 .GE. WIND(3) .AND. Y0
     &  .LE. WIND(4)) RETURN
      ERRFLG=.TRUE.
      RETURN
      END
      SUBROUTINE CHCKX0
$INCLUDE: 'COMMON.FOR'
      IF (LASTX0 .GE. WIND(1) .AND.
     &  LASTX0 .LE. WIND(2)) RETURN
      IF (X0 .GE. WIND(1) .AND. X0
     &  .LE. WIND(2)) RETURN
      ERRFLG=.TRUE.
      RETURN
      END
```

Graphics Development Toolkit

John Warnock
IBM Corporation

Editor's note: This article is adapted from the IBM Personal Computer Seminar Proceedings.

The IBM PC Graphics Development Toolkit (GDT) and its Virtual Device Interface (VDI) provide support for graphics applications on a personal computer. The VDI includes device drivers for many IBM graphics devices plus a VDI controller to communicate with the device drivers. The VDI controller includes a number of graphics drawing primitives and functions that simplify programming in a graphics environment. Graphics Development Toolkit completes the offering with language bindings that let your application programs use all the features of the VDI. Figure 1 shows the relationship of the GDT features.

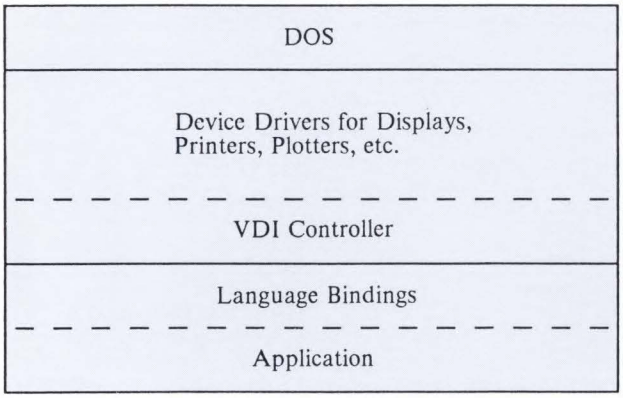


Figure 1. Graphics Development Toolkit Features

Using the Graphics Development Toolkit, your application might call for a circle via a language binding. The binding passes the request to the VDI controller, which recalls the specific steps to produce a circle and passes those steps to a device driver. The device driver takes the instructions and coordinates "as is" or translates the coordinates to fit the limitations of the device. The driver sends the data and control codes to the device, which then draws the circle.

Graphics Standards

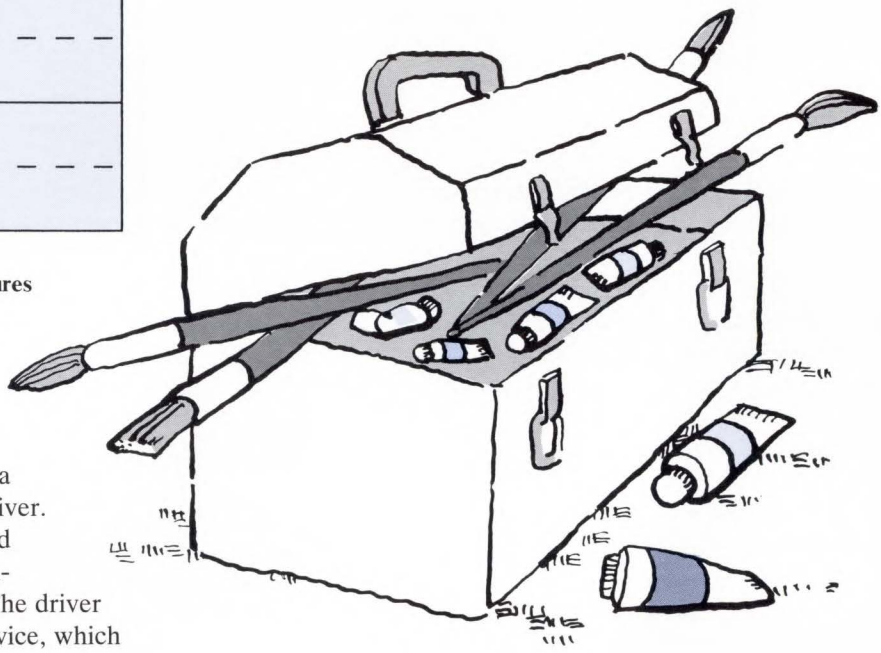
In order to define present and future needs for graphics applications, the International Standards Organization (ISO) and the American National Standards Institute (ANSI) have proposed graphics standards. These standards allow graphics applications to support all hardware devices that accept the standards. Thus, application programmers need not be concerned with specific hardware, and can write software that is easily migrated to other devices.

ANSI Technical Committee X3H33 established a definition of the ideal graphics device. Since this device does not really exist, and most existing devices emulate part of this "ideal," it is called a "virtual" device.

This virtual device uses a Normalized Device Coordinate (NDC) system. This is an area of 32,767 units by 32,767 units. These units can represent meters, inches or any required measurement. Actual physical devices use all or a subset of these coordinates or convert them to fit the device.

Virtual Device Interface

The Graphics Development Toolkit (GDT) includes a Virtual Device Interface (VDI) that follows the



proposed ANSI X3H33 definition. This and other IBM graphics software products use the VDI and its device drivers which:

- Provide a standard device interface.
- Enable programmers to design a graphics application without being concerned about the particulars of the many input/output graphics devices available.
- Can be used to create simple and complex graphics images.

Device Drivers

The VDI Controller allows communication between device-independent software and device-dependent drivers that serve as interfaces between the VDI controller and the graphics devices. Each specific device has its own driver that translates all the information exchanged between the device and the application program. To support a new device, you write a new device driver instead of changing the application.

The Graphics Development Toolkit includes a Virtual Device Interface that follows the proposed ANSI X3H33 definition.

The Graphics Development Toolkit provides a set of device drivers for a variety of IBM devices including:

- IBM Enhanced Graphics Display and Adapter
- IBM Color Graphics Display and Adapter
- IBM PC*jr* Video Subsystem
- IBM Graphics Printer
- IBM Color Printer
- IBM Compact Printer
- IBM Color Plotters
- IBM Game Adapter
- Metafile Device

The VDI Controller determines the size of the largest device driver you will be using and allocates space for it. As you use different devices, the VDI Controller swaps device drivers within the memory allocated for that class of driver.

Device drivers are called using the DOS CONFIG.SYS file. Drivers can be assigned to groups for swapping or made resident. Only one driver from each group will be resident in memory at a time. Memory is allocated according to the largest driver in each group. You can also describe paper and ribbon information to the driver at the time it is called.

Devices are referred to by generic names: PRINTER, PLOTTER, CAMERA, MOUSE, DISPLAY, etc. You can have multiple devices in the same category attached to your computer. The DOS SET command allows you to change between different devices. For example, your CONFIG.SYS file has drivers for two different graphics displays.

```
DEVICE=VDIDY004.SYS /GROUP:DISPLAY
DEVICE=VDIDY006.SYS /GROUP:DISPLAY
```

The first device is the medium-resolution driver. The second device is the high-resolution driver. The last driver in any category will become resident when your system starts, so the high resolution driver is loaded. To select the medium resolution driver, you type:

```
SET DISPLAY=VDIDY004
```

VDI Controller

The VDI Controller of the Graphics Development Toolkit provides many graphics routines that include:

- Control functions
- Output functions
- Attribute functions
- Input functions
- Inquiry functions
- Device specific PEL (Picture Element) functions.

Control functions allow the GDT to start, stop, and control workstations, devices, text, and graphics cursors. Your application can request device control functions without necessarily referring to a specific device. This lets your program handle a wide variety of devices.

Output functions draw basic graphics primitives such as polylines, polymarkers, fill areas and text, and generalized graphics primitives, including circles, arcs, pie slices and bars. This makes writing programs faster and easier because the most common routines already exist and can be called by the application program.

The polyline primitive draws lines between sequences of end points, while the polymarker primitive places a marker symbol at each point in the line.

Programming productivity is improved by the easy way functions are called.

For sophisticated workstations with native graphics functions, the VDI can access these capabilities through a Generalized Drawing Primitive (GDP). Circles, for example, are described by the location of the center and the radius.

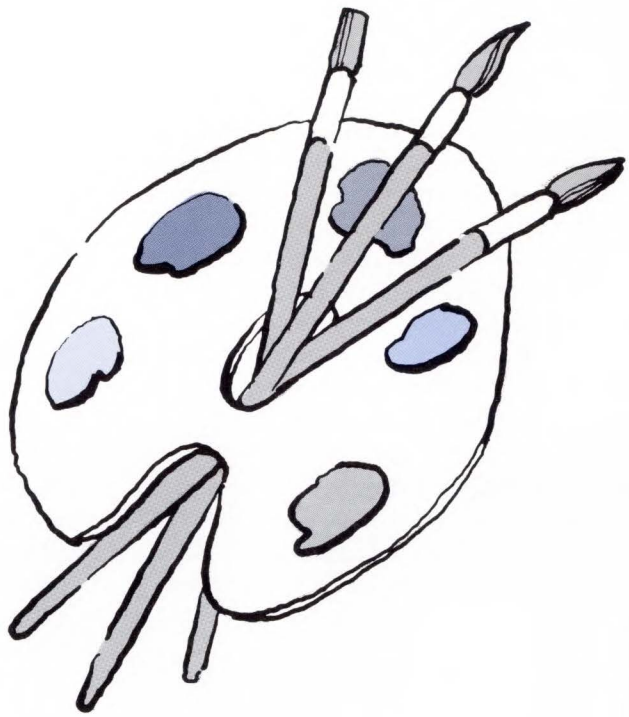
The text primitive displays text strings at any position with any orientation in various styles, sizes and colors. Three types of text models such as alpha, graphic and cursor are supported. Alpha text is used for chart labeling. Graphic text allows you to assign style, size, color, and rotation to device-dependent text. Alpha and graphic text is addressed by Normalized Device Coordinates (NDC). Cursor text places device-independent text on the screen that can be addressed by the cursor in row/column mode, or used to create menus and interactive input. Cursor text is mutually exclusive with alpha and graphic text.

In addition, the Virtual Device Interface Controller has bar, arc, and pie slice primitives. The VDI Controller can accommodate the differences in devices to keep the aspect ratio correct (preserved). This performs workstation-dependent calculations to keep the horizontal and vertical scales in proportion. Non-preserved mode accepts output to devices "as-is" and does not compensate coordinates.

The attribute functions assign color, style, size, and pattern to the lines, circles, polygons, and text. Each primitive (polyline, polymarker, etc.) has its own set of attributes associated with it. Text can have font, style, and rotation assigned (subject to device limitations), while lines can be given color, width, and style. Attributes can also be set for workstations when they are opened.

Input functions allow you to interact with your application. Input can be the cursor position, the dial setting from a device, a function key or keyboard entry. Input also can be in request mode where the application stops and waits for input from the device, or sample mode where the application will take input from a device if presented but does not wait.

The VDI also has inquiry operations that programmers can use in determining error conditions or device and attribute status. The application can also inquire as to whether a device can handle certain functions before requesting device output.



The VDI supports faster devices and faster operation or picture-element (PEL) functions. The PEL functions move one or more pels and provide animation and image generation. Certain paging functions are also available. These functions tend to be device specific and should be avoided to achieve true device independence.

Language Bindings

The GDT has a set of linkable libraries for graphics and text functions. GDT functions are grouped into eight major functional areas:

- Workstation functions
- Paging functions
- PEL functions
- Cursor control functions
- General graphic functions
- Graphic functions and attributes
- Text functions
- Input functions

The documentation includes specific language syntax for each of the supported functions.

Language interfaces are provided for:

- IBM PC BASIC Compiler version 1.00
- IBM PC FORTRAN Compiler 2.00
- IBM PC Professional FORTRAN

- IBM PC Pascal Compiler 2.00
- IBM PC Macro Assembler
- Lattice C 2.00 developed by Lattice, Inc.

Programming graphics is a simple process. Using the appropriate language binding, you initialize or open the device. Depending on the device, it may respond with information on the fonts, coordinates or colors it can handle. Then you set the attributes (width, size, style, color, font, etc.) for the particular primitive you want to use. You then call the graphics primitive to output to the device and input any status information. You can then repeat the attribute/output/input steps until you close or terminate the device.

Device Driver Builder's Toolkit

If you want to write your own device driver, you can order a Device Driver Builder's Toolkit (6277786) using the form that comes with the GDT Reference Manual.

The kit includes a reference manual that provides the information necessary to generate a device driver that will work with the Graphics Development Toolkit. In addition, the kit includes four diskettes that contain:

- Skeleton device drivers and the necessary code to be linked with new device drivers
- Four sample device drivers for a mouse, display, plotter and printer
- Utilities to build new device drivers
- Programs to test new device drivers

Hardware Requirements

The IBM Graphics Development Toolkit requires a member of the IBM Personal Computer family with:

- 128KB of memory
- A graphics display and appropriate adapter
- DOS 2.10 or higher
- A language compiler
- One 360KB diskette drive

Any applications using this product that you develop for redistribution require a redistribution license from IBM. You can write to IBM for information about redistributing the Graphics Development Toolkit and writing your own device drivers to interface with the VDI. The address is:

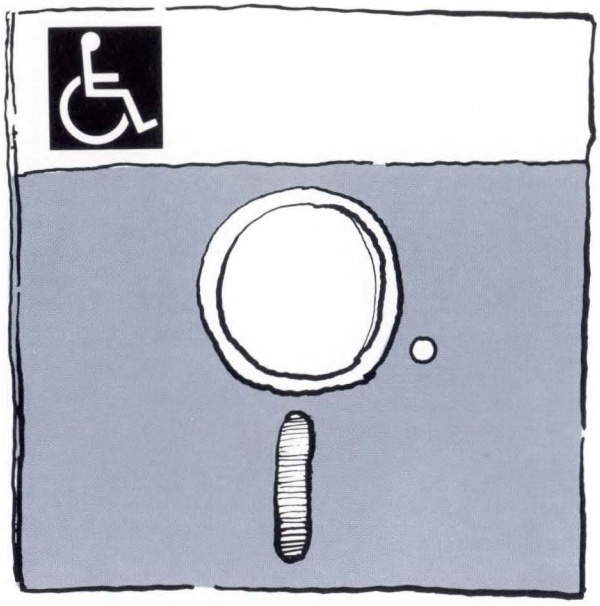
Graphics Development Toolkit
IBM Personal Computer
P. O. Box 1328-A
Boca Raton, FL 33432

The Software Handicap

Gary J. Bullard
Tulsa Computer Society

In the early days of microcomputing (less then ten years ago) there was little good software to be found anywhere. Indeed, little software of any kind was to be found. Computer clubs were formed to help answer this need, and members traded their home-written programs everywhere. Public-domain software libraries were accumulated and made available to club members for the cost of a cassette tape or diskette.

The evolution of microcomputer software is particularly interesting. If your computer club maintains a public domain library, try to get copies and explore the contributed programs. You will find some valuable programs, and a few that were first efforts—and look like it. Early computers were short on memory and used slow cassette tape storage. This forced programmers to develop tight coding techniques. This meant no remarks, multiple statements per



line (if their version of BASIC allowed it), very short variable names, and short or non-existent instructions and prompts. In other words, a lot of programs were written that may have been masterpieces. But they were difficult to use and nearly impossible to debug or maintain.

Today things are much better. Memory is cheap, disk drives are common, and software is abundant. There is no longer any excuse for writing programs that do not contain clear instructions and unambiguous prompts. Software houses that sell poorly documented programs that are difficult to use soon gain a well-deserved bad reputation. Thus, most commercially available software can be expected to perform as promised.

Of course, there are always exceptions. Don't buy a program without asking questions and receiving a demonstration. Caveat Emptor applies to software purchases as much as any other transaction.

This only points up the fact that things are better today—but not perfect. What can be done to improve software? Many things. I have a few suggestions below.

The majority of first programming efforts were games. Early games were keyboard-controlled due to the lack of any other method. Games that required directional controls used the numeric pad to signal directions, or some other combination of keys if there was no numeric keypad. Using "U" for up, "D" for down, "L" for left, and "R" for right soon fell out of favor. These controls worked fine for the "I take turn, you take turn" games, but when real-time games appeared ("arcade-style" games), these controls were insufficient—too slow and too hard to find in a hurry.

Putting the commands in a block (for instance, i,j,k,m, for up, left, right, down on the keyboard) helped, but even that was dropped in favor of game paddles and joysticks. Most software houses offered the option of paddle or keyboard control in case you didn't have the paddles or preferred the keyboard, which was an improvement.

Some games require joysticks to play and have so many features that the keyboard is frequently used to select between them. Again, an improvement, but why not devote some programming skills to reducing the number of controls for a game?

There is no longer any excuse for writing programs that do not contain clear instructions and unambiguous prompts.

I have a game program that uses the keyboard to enter directional commands. It uses the left and right arrow keys on the keyboard to signal left and right moves, and the "A" and "Z" keys to signal up and down. This is fine for me and I can play this game without getting confused. My left hand operates up and down and my right hand operates left and right.

But I know three people who have only one hand each. This game is all but impossible for them. It would have been so simple to program the game to use the ";" and "/" keys for up and down, making the game playable for the one-handed.

Why two buttons? My one-armed friends are more fortunate than many people. Another man I know is quadriplegic. He operates his computer with a stick held in his mouth. He has just enough control of one arm that he can hold the shift and control keys when necessary. Most programs are difficult or impossible to operate. But he persists, and does remarkably well. Some handicapped users cannot do even this well.

Recently a representative from the Tulsa Rehabilitation Center came to a Tulsa Computer Society meeting and asked the membership to help them write programs and modify computers for use by the handicapped. They have patients who have broken their necks, suffered brain injuries, or have severe arthritis. These people are so limited in their movements that some of them can only operate a computer by blowing or sucking through a rubber tube.

The Rehabilitation Center designed and built some switches that could be activated by sucking or blowing into a tube. These sip'n'puff control switches can be connected to a computer, for instance the game connector adapter. Sipping operates button zero and puffing operates button one. Both buttons cannot be "on" at the same time.

The experienced programmers reading this are already thinking of possibilities. Computers are binary machines; what more natural than binary controls? Consider how to make the

program you are currently working on controllable with a sip'n'puff control.

For instance, take Pacman. This game is usually played with a joystick control, often with a keyboard option. All it needs are four directional controls. There are no guns to fire, barrels to leap, or hyperspace drives to engage. The major problem is the speed of the action versus the speed of reaction of the player. Let's try this scenario: Game starts, giving a choice of options: keyboard, joystick, sip'n'puff. Player sips on his tube, signaling his choice. This puts the game in the "halt" mode—Pacman and ghosts move only while the sip switch is on. The game starts, with Pacman heading east. As long as the player keeps the sip switch on, Pacman continues in the same direction and the ghosts are in pursuit. Pacman reaches an intersection and the player releases the switch. All action on the screen stops, except that Pacman slowly rotates in ninety degree increments. As soon as Pacman is facing the direction he needs to go, the player sips on the tube again and action resumes.

Thus a popular arcade-style game is adapted for play with only one switch needed. Similar modifications can be made to other games. For instance, a pinball game could be played with two switches (and usually is) and possibly only one. The difficulty with pinball games is that the firing spring must be set with the game paddle or joystick. How do you get that degree of control with only on-off switches? Simple. When it is time to fire a ball, let the computer move the spring from its tightest to its loosest and back again. When the player sees the spring at the point he wants, he sips on the tube.

..there is an extreme lack of game software that bedridden people can play on their computers.

Another type of game that is very difficult for handicapped people to play is the adventure game. Typing is difficult or impossible for many people. The Tulsa Rehabilitation Center has several programs that allow text entry with the sip'n'puff switches. It is a slow, tedious process, but it is the only way some handicapped people have to communicate with the rest of the world. These same techniques can be applied to text input for the adventure game.

These text programs work in a variety of ways. Usually, the program presents an alphabet on the screen with the first letter printed in reverse or highlighted in some way. By sipping on the tube, the patient moves the marker down the alphabet to the chosen letter. Then the patient puffs into the tube and the computer accepts that letter and adds it to the word or sentence being built.

Sip'n'puff text routines can be added to adventure programs, or the programs can be written like the books that allow you to read several paths to the various endings. Instead of requiring the player to enter complete sentences in hopes of guessing the correct move, let the program print a menu of choices. The player may then sip to a likely choice and puff it into action.

I have concentrated on games for examples because there are plenty of programs for handicapped people that enable them to print messages, turn on lights, signal the nurse, and even change channels on the television. But there is an extreme lack of game software that bedridden people can play on their computers. When movement is so limited, there are few forms of entertainment in which the patient can participate. Computers are ideal for this purpose. I would like to see all software houses begin to offer sip'n'puff options for all future programs.

The Tulsa Computer Society is working with the Tulsa Rehabilitation Center to create standards for sip'n'puff controllers for all computers. If you have any questions or suggestions about standards for hardware or software, please write:

HUG (Handicapped Users' Group)
Tulsa Computer Society
P.O. Box 3211
Tulsa, Oklahoma 74101

Choosing Educational Software for Children

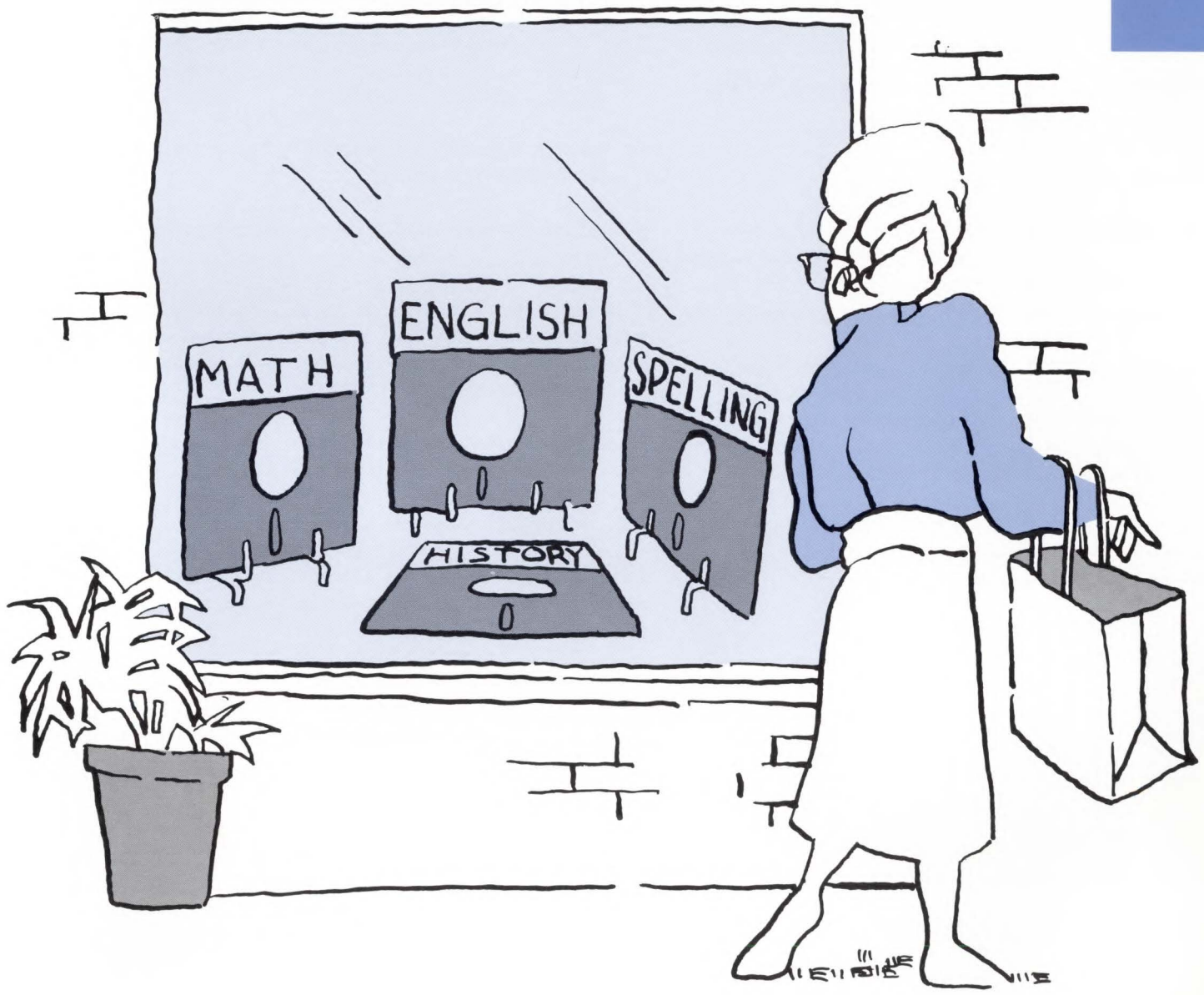
Carmen Wagner
Tucson IEEE Computer Society

It's reasonable to say that computers are, and will be, an integral part of your children's educational environment. You can influence the learning process of your children by helping them make better use of computers at home, and by choosing the appropriate software to accomplish different purposes.

Those of you who have high tolerance to frustration, and wish to experiment, are encouraged to develop your own learning programs or games (and a lot of children are doing just that). However, if that's not for you, commercial software is available at a very low cost from your user group library. Don't let the price of this software fool you; even though they may be inexpensive, some of these programs are quite good. At a higher price, you can purchase software from many of the electronic publishers. You can also contact any of the IBM Authorized Dealers or other software stores in the area.

To create your own learning materials, you can select among a range of choices:

1. High-level programming languages (BASIC, C, Pascal, FORTRAN, etc.);



2. Authoring languages, a language which facilitates the programming of interactive educational software (PC/Pilot, Logo, ZES, etc.);

3. Authoring systems, a special program created to facilitate the programming of educational materials (the Author+, Pass, Blocks, MHIAS, etc.)

Many programs written in BASIC are available in educational publications or other computer journals; they can provide ideas for programmers. Authoring systems and languages, on the other hand, are a nice alternative for parents and children who do not want to become expert programmers, but wish to experiment a little. Several authoring systems and languages are presently available; their prices vary according to their capabilities.

You can influence the learning process of your children by helping them make better use of computers at home...

For game creation, some LOGO-based programs allow you to work with graphics and create your own games with animation. The scope of this article does not allow us to cover all the programming alternatives.

As for commercial software, it is almost impossible to figure out its educational value just by reading the advertisement, or the instructions that come with it. The best way is to access the real thing and try it with your kids before you actually spend your precious dollars on it (these programs usually don't cost much, but it all adds up). There really is no substitute for hands-on experience when it comes to any kind of software, including educational packages.

The following list will help you ask the right questions when checking out software for educational applications:

1. First analyze your intentions. What skills do you want your kids to work on? What concepts do you want them to learn?

2. Talk to friends, educators, or to other parents in the club about programs that you are thinking of buying. They may have had experience with packages you are considering and may be willing to share their opinions.

3. As to the program itself, check on how easy it is to use; evaluate the documentation and instructions available; check on correctness of grammar, spelling, and content.

4. Observe your kids' reaction to it. Not all programs will be interesting to all kids. I believe that to be effective a program should be interesting and fun to use.

5. How much control do users have when using the program? Can users get help easily?

6. Can you modify the program? For instance, with spelling exercises, you may want to add or delete words to update the program.

7. How crash-proof is the program? How easily can you recover from errors?

8. What is the reputation of the supplier and publisher?

A few additional reminders. If you're considering word processors as learning tools for small kids, look at those programs in your club's library. Several adequate programs are in the public domain.

A number of publishers deal with educational software; also, many organizations will hold meetings and seminars which may be of interest to parents who want to help their kids with computer learning. Watch for those announcements in your club's newsletter or check with the local software dealers and community schools, as well as in your local newspaper.

Fooling with Boole: (IF To Be AND/OR/NOT To Be THEN . . .)

John Warnock
IBM Corporation

A mathematician named Boole
Developed some logical rules
For testing conditions
With new intuitions,
And gave us some wonderful tools.

George Boole was a mathematician who devised a set of operations to test conditions to see if they are true of false. This kind of testing is what gives programs logic, and the ability to "decide" what to do under certain circumstances. For example, we might write a checkbook balancing program and say:

```
IF CHECK > BALANCE THEN BEEP
```

If the check > (is greater than) the balance then the computer will beep to let us know. This is a simple test that produces a simple result. Boolean logic lets us test for more complex conditions by combining one or more simple tests together.

Logical Operators

BASIC has a series of Boolean logical operators to build more complex true/false logic, and alter numerical values. These operators work like conjunctions in sentences to build complex IF/THEN tests in BASIC. Logical operators can also work with bits to derive a true (1) or false (0) value. The logical operators are AND, EQV, IMP, NOT OR, and XOR. Given two values (THIS and THAT), we will see what each logical operand does to them.

AND

Suppose you want to make sure several things are true before proceeding. The AND operator performs a conjunction; that is, THIS *and* THAT must be true (1) in order for the result to be true (1). Given that requirement, the AND operator will produce the following results:

1 AND 1 = 1
1 AND 0 = 0
0 AND 1 = 0
0 AND 0 = 0

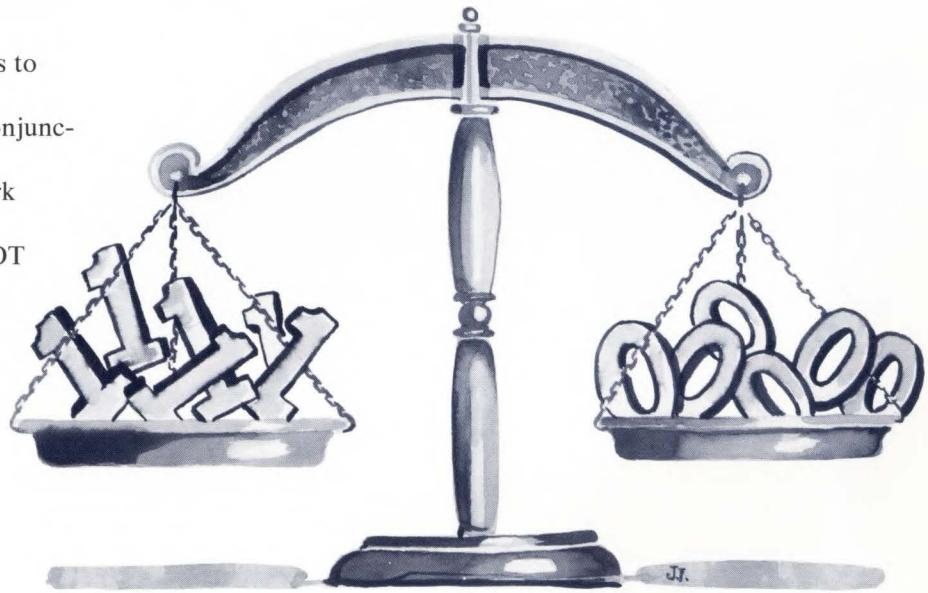
Let's apply this to our checkbook program again. Suppose we have a savings account that automatically transfers funds to checking if we run short. Our statement might read:

```
IF CHECK > BALANCE AND CHECK >
   SAVINGS THEN BEEP
```

This statement alerts us only if our check amount exceeds *both* the checking and savings balances.

EQV

Suppose we want two conditions to match, either both right or both wrong. The EQV (equivalence) oper-



ator is not concerned with truth as much as sameness. If THIS and THAT are the same, whether they are both true (1) or both false (0), then equivalence is satisfied. Here's how EQV works:

1 EQV 1 = 1
1 EQV 0 = 0
0 EQV 1 = 0
0 EQV 0 = 1

Let's get back to our checking program. Suppose we are reconciling our checkbook against our bank statement. If the amount of the check matches the amount on the statement, then the checkbook balance should match the statement balance. If the check amount doesn't match, the balances also shouldn't match.

This kind of testing is what gives programs logic, and the ability to 'decide' what to do under certain circumstances.

While this last condition isn't good, having a matching balance with mismatched checks or matching checks and a bad balance probably indicates it is time to find another bank or a new calculator. The BASIC might look like this:

```
IF CHECK = AMOUNT EQV CHECKBOOK
   = STATEMENT THEN PRINT
   "So far, you and the bank
       are about even"
```

IMP

Actually, you might want to test for one thing before doing a second test. The IMP (implication) operator determines the truth of the second value based on the truth of the first value. We look at the truth of THIS and what it implies about THAT. When THIS is true (1), then THAT is is tested to see if it is correct. If THIS isn't true (0), then we simply assume THAT is true (1). So, IMP produces:

1 IMP 1 = 1
1 IMP 0 = 0
0 IMP 1 = 1
0 IMP 0 = 1

Back to check reconciliation. Perhaps we can produce a better test. If the check and the amount on the statement aren't equal, we can assume we are out of balance and not compare our checkbook balance against our statement balance. We would just go ahead and print an out-of-balance warning. If the check matches the statement amount, then we check to see if the checkbook and statement are out of balance (not equal: <>), and if true, print the out-of-balance warning. Here's the BASIC:

```
IF CHECK = AMOUNT IMP CHECKBOOK
   <> STATEMENT THEN PRINT
   "Balance Error"
```

NOT

There may be another way to write that test. The NOT operator changes something into what it is *not*. In other words, not true (1) becomes false (0), and not false (0) becomes true (1). The not operator reverses existing conditions like this:

NOT 1 = 0
NOT 0 = 1

So, to test to see if two things are not equal could be written two ways. The second version of the statement looks like this:

```
IF CHECK = AMOUNT IMP NOT CHECKBOOK
   = STATEMENT THEN PRINT
   "Balance Error"
```

OR

The OR operator is concerned with true and false, but unlike the AND operator, it is much easier to please. Either THIS *or* THAT has to be true (1) to satisfy OR. If both THIS and THAT are true, so much the better. Only when THIS and THAT are both false (0) does OR become false. Here's how OR works:

1 OR 1 = 1
1 OR 0 = 1
0 OR 1 = 1
0 OR 0 = 0

We may really want to know when *either* our checks don't equal the statement amount *or* our checkbook balance is out of kilter with our statement. This is what you might write:

```
IF CHECK <> AMOUNT OR CHECKBOOK
   <> STATEMENT THEN PRINT
   "Balance Error"
```

XOR

There is another kind of OR operator we can use. The XOR (exclusive or) operand is like a fussy OR. It says THIS or THAT can be true, but they cannot *both* be true. So, XOR produces:

1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 1 = 1
0 XOR 0 = 0

Suppose we only want to print the balance if it is going to be a positive or negative value. Zero balances don't count, so we know the check can either be more or less than the balance.

```
IF CHECK = AMOUNT XOR CHECKBOOK
   = BALANCE THEN PRINT
   "Something's rotten in Denmark"
```

Bit Twiddlers Tweaking Bits Bag Big Bucks

What we have covered so far is Boolean operators in simple IF/THEN statements. Boolean operators can also work on single bytes of information to filter out or modify single bits. There are practical uses for this, as we shall see.

Boolean operators are designed to work with 1's and 0's to determine true or false. These are the same kinds of 1's and 0's that make up characters. Taking the letter "Z," there are eight bits that add up to 90, its ASCII value. The bit positions are set in powers of 2, so the whole thing looks like this:

POS	Z
1	0
2	1
4	0
8	1
16	1
32	0
64	1
128	0
ASCII	90

Excited yet? Well, let's compare the lowercase z.

POS	Z	z	
1	0	0	
2	1	1	
4	0	0	
8	1	1	
16	1	1	
32	0	1	<---
64	1	1	
128	0	0	
ASCII	90	122	

The only real difference between the two is that the bit in position 32 is turned on. If we compare all the uppercase letters with their lowercase counterparts, we see the same bit in position 32 making the difference.

Boolean operators can also work on single bytes of information to filter out or modify single bits.

Now we can really put these Boolean operators to work. Remember that while these operators work for 1's and 0's, they can also do groups of them as well. Let's take the lowercase "Z" and OR the value 32 to it, which looks like this:

POS	Z	OR " " =	z
1	0	OR 0 =	0
2	1	OR 0 =	1
4	0	OR 0 =	0
8	1	OR 0 =	1
16	1	OR 0 =	1
32	0	OR 1 =	1
64	1	OR 0 =	1
128	0	OR 0 =	0
ASCII	90	32	122

In BASIC it might look like this:

```
10 PRINT CHR$(ASC("Z") OR ASC(" "))
```

You can produce the same result by starting with the value 223, the bit-by-bit opposite of 32, and IMPing the character "Z" to it:

```
 APOS    ■   IMP  Z  =  z

   1     1   IMP  0  =  0
   2     1   IMP  1  =  1
   4     1   IMP  0  =  0
   8     1   IMP  1  =  1
  16     1   IMP  1  =  1
  32     0   IMP  0  =  1
  64     1   IMP  1  =  1
 128     1   IMP  0  =  0

ASCII 223       90    122
```

The BASIC statement might read:

```
10 PRINT CHR$(ASC("■") IMP ASC("Z"))
```

By the same token, we can convert the lowercase z to an uppercase Z in a couple of different ways. One method works by ANDing 223 to the character.

```
 POS     z   AND   ■  =   Z

   1     0   AND   1  =   0
   2     1   AND   1  =   1
   4     0   AND   1  =   0
   8     1   AND   1  =   1
  16     1   AND   1  =   1
  32     1   AND   0  =   0
  64     1   AND   1  =   1
 128     0   AND   1  =   0

ASCII 122       223     90
```

The BASIC statement might look like this:

```
10 PRINT CHR$(ASC("z") AND ASC("■"))
```

The nice thing about these three methods is that they only affect the characters you want to affect. If you are converting lowercase letters to uppercase, uppercase characters are left alone. The same is true when converting uppercase letters to upper case. Adding or subtracting the value of 32 won't work if the character is already the way you want it.

Booling Around with Memory

Boolean operators can do more than manipulate characters. You can use them to check or modify memory in your PC. Using the AND operator, you can PEEK memory and check if specific bits are turned on.

The byte at address 1047 stores the status of the keyboard shift keys. Each bit of the byte handles a different key. If the first bit is on, it means the right shift key is pressed. You could check to see if the other shift key is pressed by the following BASIC statement:

```
10  DEF SEG=0:IF PEEK (1047) AND 2
   THEN PRINT "Left Shift Pressed"
```

Taking this one step further, you can manipulate the keys yourself. Let's say you know you want input in all upper case, or want to allow numeric input from the numeric keypad. First use PEEK to get the value, use Boolean operators to change it, then POKE the value back out. You can use statements similar to those in Figure 1 to make that happen.

```
10  DEF SEG=0 'Sets memory address
20  POKE 1047, PEEK (1047) OR 64
    'Turns on Caps Lock
30  INPUT "Enter some characters:"
    ;A$ 'Sample input
40  POKE 1047, PEEK (1047) AND 191
    'Turns off Caps Lock
50  INPUT "Enter more characters:"
    ;A$ 'Sample input
60  POKE 1047, PEEK (1047) OR 32
    'Turns on Num Lock
70  INPUT "Use the keypad numbers: ";A
    'Sample numeric input
80  POKE 1047, PEEK (1047) XOR 32
    'Turns off Num Lock
90  INPUT "Use the keyboard numbers:";
    A'Sample numeric input
```

Figure 1. Sample Statements

In this case, the bit that represents the value 64 controls the Caps Lock, and the bit that represents 32 controls the Num Lock. We OR'ed the value to turn the bit on. We AND'ed all the bits but 64 to turn the Caps Lock off, and XOR'ed 32 to shut the Num Lock off. These were two different ways of doing the same thing.

As you can see, Boolean operators can do many useful things. They let you create better logic in your programs, and give you greater control over your PC. Just be very careful what you PEEK and POKE.

Passwords: A Serious Matter

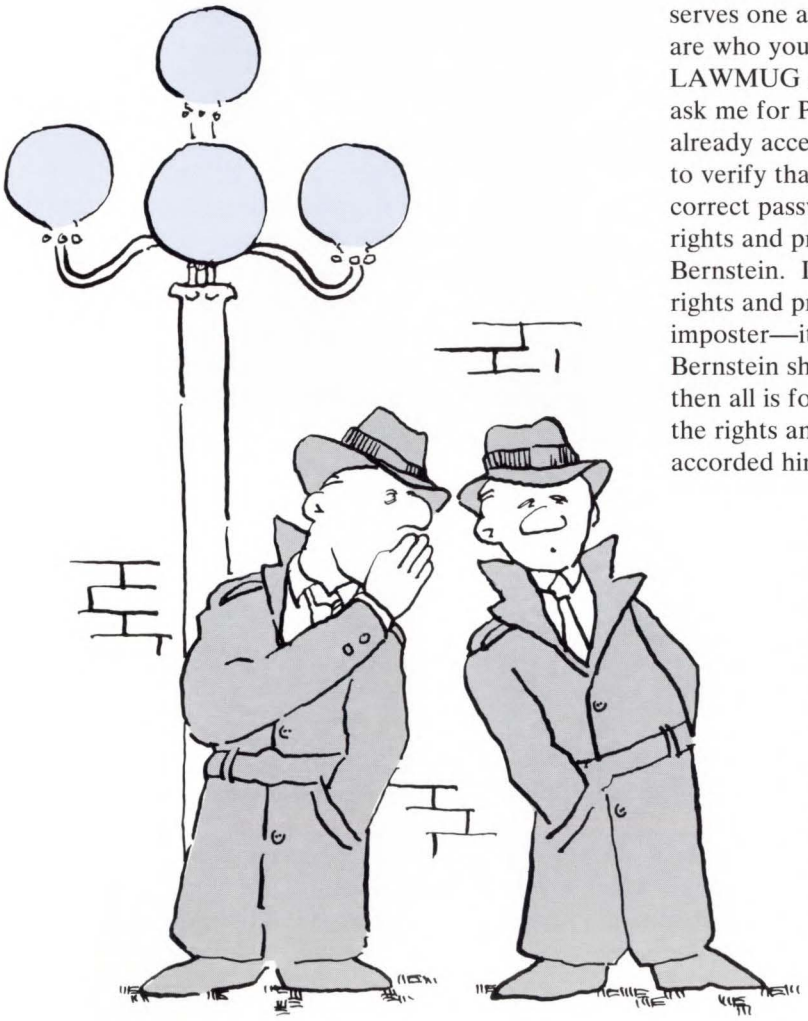
James J. Ayres
LAW MUG

Passwords and Computer Security

This article will examine some of the aspects of security that a computer user should be aware of. Security, as used in the context of a computer, should not be thought of as denying access to the computer or its resources, but instead considered as a measure of controlling who may access what and under which circumstances.

Super Users in the World of Computers

Systems are generally designed so that one user is capable of accessing all of the system's functions.



This "superuser" is usually referred to as the System's Operator. The system's operator is unrestricted in capabilities for a very simple reason: someone has to be able to access functions that may be otherwise blocked—either by a defective program or a defective security scheme. In the event of trouble, the system's operator needs to be able to access the troubled area.

As a simple analogy, consider a system to be your home. The locks on the doors are designed to limit access to your premises. If there were a fire in your home, you would need the key to get into your house and extinguish the fire. Unlike other "users" of your premises, you cannot maintain the safety and integrity of your home if you have to ring the door bell to get inside and put out a fire. A system's operator cannot maintain the safety and integrity of the data in the system if the restrictions placed on other users prevented access to the system.

As a user of a multiuser system, you are generally asked to supply or use a password. A password serves one and only one purpose—it verifies that you are who you say you are. If I were to sign on to LAWMUG as Paul Bernstein, the system would then ask me for Paul's password. Bear in mind that I have already accessed the system. The password is merely to verify that I am who I say I am. If I fail to give the correct password, the computer refuses to give me the rights and privileges granted to the user Paul Bernstein. Instead, the computer accords me the rights and privileges anyone would grant an imposter—it "slams the door in my face." If Paul Bernstein should call in and correctly identify himself, then all is forgiven and Paul Bernstein is granted all of the rights and privileges the system's operator has accorded him.

As you can see from the above example, the password is a simple yet fairly effective means of limiting access to the system. Once the user is beyond this level of verification, the system can only assume that the user is who he says he is. Although there are more sophisticated forms of security to verify users, the system's operator has to balance the amount of program overhead, hardware, and labor involved in implementing the more sophisticated measures against the desire to allow easiest access to the system.

The system's operator must make a trade-off between ease of access and security. The overriding concern is the level of the data's importance and confidentiality.

After a decision has been made to use password security on a system, it then becomes the user's burden to maintain that confidentiality. Unlike forged documents, someone accessing a computer with your identification does not leave "hard" evidence of the fraud.

A *password serves one and only one purpose—it verifies that you are who you say you are.*

The only records of the impersonation are the same records used to verify that the user is valid. Because these records are digital and contain no unique identifying features, it is almost impossible to prove that the impersonator wasn't the true user.

Suggested Rules for Computer Security

1. **Don't use a logical password that is easy to figure out**. Someone intent on impersonating you will try the easy password guesses first. For example, I would never use a password consisting of any part of my name or a close family member's name, my address, my auto license, etc. This information is too easy to obtain, and if an imposter has targeted you as his "doorway" to the system, he or she can probably get this information. Use a password that is a combination of letters and numbers that are only meaningful to you.

2. **Change your password often**. If your password remains the same for a long period of time, the odds that a persistent imposter will hit upon it are greatly increased. Again, don't get lazy and change your password to one that violates the first rule.

3. **Never give your password to another user or enter it into a system if you are uncertain as to the reason for the request**. Otherwise, you may have given someone else the irrevocable authority to act on your behalf. Furthermore, because of the nature of computer systems, you cannot prove that your "agent" was not you. You are initially responsible for everything that that person does while acting as you. There are several methods used by imposters to acquire a valid password directly from the user. One method is to use a system's communications mode to send a message to another user. This method causes some form of message to appear on the user's screen indicating that something technically meaningless has occurred and the user should re-enter the password. The imposter then watches what the user types. Another method involves setting up a program which follows the same technique as above, but the program then stores the password in a file and the imposter will check for a password later. Another method, recently used, is to set up a system to collect passwords. This happened in the Chicago area when a bulletin board was set up by imposters. It gave the appearance of legitimacy, but was later used by the imposters to access other systems because their users had the same password on several systems. Which leads to the last rule of password usage:

4. **Never use the same password on different computers**. Using the key analogy above, if all of the locks on your personal possessions have the same key, you wouldn't entrust that key to anyone. Why use the same password on several systems? If you do, you run the risk that someone will get your password and then use that information to access all of the systems you access. You will soon be unwelcome on several systems (if not a suspect in a computer crime case).

Final Caveat

Finally, be an observant user. Most systems indicate the last date you signed on or off. If that date isn't correct, you can begin to suspect that someone may be using your identification on the system. Immediately do two things: contact the system's operator and change your password.

Although the subject of computer security is generally the problem that concerns the system's operators and designers, they can't prevent the user from foolishly giving away the key. Users have to play an important role. Following the few suggestions above, you can help prevent others gaining access to systems by using your good name.

New Products

Hardware

IBM RT Personal Computer System Overview

The IBM RT Personal Computer System provides new, high-performance workstation and multi-user computing solutions to meet the requirements of the engineering/scientific, academic, and computer-aided design / computer-aided manufacturing (CAD/CAM) communities.

The IBM RT Personal Computer Advanced Interactive Executive (AIX) Operating System is a multi-user, multitasking, virtual memory operating system designed to take advantage of the IBM RT Personal Computer's 32-bit architecture. The system is complemented by the announcement of several system support programs, software development tools, and personal productivity applications.

Combined with the IBM 5085 Model 1 or Model 2 Graphic Processing Unit, the IBM RT Personal Computer forms a workstation with either standalone or host-connected CAD/CAM ability.

IBM 6150 RT Personal Computer Models 020, 025, and A25

IBM 6151 RT Personal Computer Model 010

IBM announces a powerful and extendible workstation-oriented system for the personal computing requirements of the technical professional. The IBM RT Personal Computer, an addition to the family of Personal Computer-related products, features a 32-bit reduced instruction set microprocessor with virtual memory, and optional Personal Computer compatibility for both programs and hardware attachment. The system is designed to satisfy computing needs typical of the academic, engineering/scientific, and CAD/CAM environments, with discipline-specific applications for personal-productivity.

IBM 6153 Advanced Monochrome Graphics Display

IBM 6154 Advanced Color Graphics Display

IBM 6155 Extended Monochrome Graphics Display

Three new displays are announced for the IBM RT Personal Computer. The 6153 is a 12-inch, medium-resolution, black and white display. The 6154 is a 14-inch, medium-resolution, color display and the 6155 is a 15-inch, high-resolution, black and white display. The three units can display both text and all-points-addressable graphics and provide a range of display functions.

IBM 6157 Streaming Tape Drive

The IBM 6157 Streaming Tape Drive is a 1/4-inch tape drive for attachment to the IBM RT Personal Computer and IBM System/36. The 6157 provides fast, convenient, save/restore, and data interchange capabilities for each of the product families. Data interchange between the RT Personal Computer and the System/36 is not supported.

IBM 5080 Graphics System Features for IBM RT Personal Computer Attachment

The 5080 Graphics System connected to a properly configured RT Personal Computer (IBM 6150 Model 020, 025, or A25) forms a high-function graphics workstation that operates in either a stand-alone or host-connected environment. The new 5080 features required for the connection include:

- RT Personal Computer Attachment (#6150) for the IBM 5085 Model 1 and S01
- Graphics Keyboard (#4651) for the IBM 5085 Model 1 and S01
- Serial Link Enhancement (#9150) on IBM 5088 Models 1, 2, and 1R with serial numbers below 4000.

The 5085 Model 1A and 5085 Model 2 also can be connected to the RT Personal Computer 6150 Models 020, 025, and A25 without the above features.

IBM 5085 Graphics Processor Model 2

The 5085 Graphics Processor Model 2 brings significant improvements and functional enhancements to the 5080 Graphics System. High performance CAD/CAM applications in both mainframe interactive and standalone (when attached to the IBM RT Personal Computer Model 20, 25, or A25) configurations may be processed by 5080 systems using the 5085 Model 2. Included in the 5085 Model 2 are more standard features: a larger base systems memory; larger systems memory increments; a full-screen tracking cursor; a 64-bit by 64-bit

programmable tracking cursor; the ability to visually detect overlapped geometry; faster pixel write and area fill speeds; 2D/3D performance improvements; and a range of other improvements.

IBM 5083 Tablet Model 12

The 5083 Tablet Model 12 is a thin, compact, flat surfaced unit for user interaction with the image on the IBM 5080 Graphics System's 5081 Display. With an 11.5-inch by 11.5-inch active area and 500-line-per-inch resolution, the Model 12 is an attractive peripheral addition to the 5080 system. It also can attach to the IBM RT Personal Computer.

An optional Stylus (#6351) and 4-button Cursor (#1511) are available for users who prefer a pen-like device or who require a hand-held unit with a fine crosshair for precise alignment and four buttons for application use.

IBM 5081 Display Model 11

The 5081 Display Model 11 provides a brighter monochrome display image for IBM 5080 Graphics System users. The new display uses new phosphors, a 60Hz screen refresh rate (non-interlaced), and circuit changes to provide enhanced black-and-white images.

IBM 6180 Color Plotter

The IBM 6180 Color Plotter is a versatile, high-quality, desktop output device that produces high-resolution presentation graphics charts, engineering drawings, graphs, and diagrams on paper and transparency film.

The IBM 6180 Color Plotter uses eight pens, and features automatic pen changing and faster plotting.

Software

IBM RT Personal Computer Advanced Interactive Executive Operating System and Virtual Resource Manager

The IBM RT Personal Computer Advanced Interactive Executive (AIX) Operating System is a multi-user, multitasking, virtual memory operating system. It can operate as a single-user or multiuser system for up to eight concurrent terminal users. A multiple virtual terminal interface allows task switching.

The AIX Operating System provides a broad range of capabilities, including a user interface with menus and a command bar, virtual memory management, device-independent input/output, menu-driven installation/configuration procedures, and an upwardly compatible UNIX System V environment.

The Virtual Resource Manager licensed program is designed to provide device independence and system expandability. It consists of a collection of processes, device drivers, and runtime routines that combine to support a virtual machine interface to an operating system such as the IBM RT Personal Computer AIX Operating System. The Virtual Resource Manager is included as part of the AIX Operating System, but it also is available as a separate licensed program. As a separate program, the Virtual Resource Manager is intended for use by customers who are planning to develop programs that will not use the IBM RT Personal Computer AIX Operating System.

IBM RT Personal Computer INmail/INnet/File Transfer Program

IBM RT Personal Computer INmail/INnet/File Transfer Program extends the capability of the IBM RT Personal Computer AIX Operating System INed editor by providing for the queued transfer of files and for interactive entering of commands to be executed on remote systems.

IBM RT Personal Computer/Personal Computer AT Coprocessor Services

IBM RT Personal Computer/Personal Computer AT Coprocessor Services provides the capability for many IBM Personal Computer AT programs to run on the IBM RT Personal Computer without modification or recompilation.

IBM RT Personal Computer 3278/79 Emulation

IBM RT Personal Computer 3278/79 Emulation increases the flexibility of the IBM RT Personal Computer system as an intelligent workstation for interactive use. It provides 3270 emulation and file transfer capability via attachment to an IBM 3274 Control Unit or to an IBM 4361 Display/Printer Adapter or Work Station Adapter. A subset of the functions of a 3278 Display Station Model 2 or a 3279 Color Display Station Model 2A or Model S2A are emulated.

IBM RT Personal Computer Structured Query Language/RT Data Base

IBM RT Personal Computer Structured Query Language/RT (SQL/RT) Data Base is a relational data-base management system with both programmer and end-user facilities. It includes an implementation of the SQL data language and provides interactive facilities to enter, retrieve, modify, and display or print relational data.

IBM RT Personal Computer Data Management Services

IBM RT Personal Computer Data Management Services extends the IBM Advanced Interactive Executive (AIX) Operating System file and directory system, providing a multi-user data storage structure and access

to a wide variety of applications. It provides record management services, field management services, multiple indexing of files, multiple-request data sharing across files, and file utilities.

IBM RT Personal Computer Languages - FORTRAN 77, BASIC, Pascal

IBM RT Personal Computer FORTRAN 77, an implementation of the FORTRAN 77 language with enhancements, was developed by IBM and INTERACTIVE Systems Corporation. It includes a compiler and runtime libraries. The Extended FORTRAN Language and Rational FORTRAN structured preprocessors for FORTRAN are also provided.

The IBM RT Personal Computer BASIC Compiler and Interpreter provides a level of function comparable to that of IBM Personal Computer BASIC 1.1 Interpreter Advanced Version plus extensions.

The IBM RT Personal Computer Pascal provides a level of function comparable to the IBM Personal Computer Pascal Compiler Version 1 plus extensions. The user can select a Personal Computer mode, which is similar to IBM Personal Computer BASIC or Pascal, or a native mode, which provides additional IBM RT Personal Computer capabilities.

IBM RT Personal Computer Professional Graphics Series

The IBM RT Personal Computer Professional Graphics Series consists of four licensed programs that provide support for graphics program development as well as for end-user interfaces.

IBM RT Personal Computer Graphics Development Toolkit provides a set of graphic device drivers for printers, plotters, and displays to provide device independence for programs. It also includes a set of graphics primitives that can be called by high-level languages to perform functions, such as displaying pie slices and drawing lines, polygons, and circles using the IBM RT Personal Computer virtual device interface.

IBM RT Personal Computer Graphics Terminal Emulator allows the IBM RT Personal Computer to emulate the Tektronix 4010 and 4100 protocols as well as the Lear Siegler ADM-3A protocol.

IBM RT Personal Computer Plotting System is a subroutine library of functions designed to assist the user in developing programs to produce various types of charts and to develop interactive graphics applications.

IBM RT Personal Computer Graphical File System is designed to facilitate the standardized retrieval, storage, and portability of two-dimensional graphics information. A metafile interpreter allows the retrieval of encoded graphic pictures that have been created in a virtual device metafile format and their routing to alternate devices.

Professional CADAM

Professional CADAM is a two and one-half dimensional, interactive computer-aided design system for designers, draftsmen, engineers and other technical professionals that runs on the IBM RT Personal Computer 5080 Graphics System. Professional CADAM functions are nearly identical to those of CADAM Interactive Design Version 2. This licensed program may be used for a variety of applications that require two and one-half dimensional drafting capabilities. It fulfills the requirements of a wide variety of industries that require drafting in their operations. Professional CADAM may be operated in a stand-alone mode or connected to a host computer with CADAM Release 19.2 or later installed.

IBM RT Personal Computer Personal GraPHIGS

The IBM RT Personal Computer Personal graPHIGS licensed program is an advanced graphics programming interface that is based upon the proposed ANSI standard for the Programmer's Hierarchical Interactive Graphics System (PHIGS) and is designed to simplify the programming of graphics applications, particularly for CAD/CAM.

Personal graPHIGS, with over 250 graphics functions, may be used with programs written in FORTRAN, Pascal, and C to create graphics applications for the IBM RT Personal Computer with a 5085 Graphics Processor.

The Personal graPHIGS application programming interface is the same, in most respects, as the GDDM/graPHIGS interface under IBM mainframe operating systems.

Computer-Integrated Electrical Design Series/Design Capture

Computer-Integrated Electrical Design Series/Design Capture for IBM Personal Computer AT

Computer-Integrated Electrical Design Series/Design Capture for IBM RT Personal Computer

IBM Computer-Integrated Electrical Design Series (CIEDS) is a set of application programs that assist users in performing computer-aided electrical engineering on IBM 30XX, 43XX, IBM RT Personal Computer, and IBM Personal Computer AT systems. CIEDS/Design Capture program supports hierarchical development of logic designs on a host-attached IBM 5080. It provides functions that allow engineers to create, edit, and list schematics, symbols, and information associated with a design. CIEDS/Design Capture supports multi-window interactive graphics, netlist manipulation, and hierarchical design expansion. Interfaces are available to other automated design tools, including the IBM Circuit Board Design System (CBDS).

The CIEDS family of programs offers a range of capability on a variety of IBM workstation products. Users of the IBM 5080 Graphics System, RT Personal Computer and Personal Computer AT workstations are supported with a consistent application interface and common data base capability. This can minimize training as users move from one design environment to another and also allows a balancing of performance, function, capacity, and cost.

UNIRAS for the IBM RT Personal Computer

UNIRAS, UNIversal RAster System, is the collective name of a set of software programs based on a raster technique for graphics display and manipulation of pictures in color. These programs consist of graphic utilities, FORTRAN subroutine libraries and interactive programs.

UNIRAS raster display technology provides mapping (contour, thematic and demographic), seismic plotting, image processing, business and scientific charting, hidden surface removal, and visible surface shading capabilities.

The individual products include:

- UNIRAS-RASPAK
- UNIRAS-RASPAK SOLIDS
- UNIRAS-UNIMAP
- UNIRAS-GEOPAK
- UNIRAS-GEOINT
- UNIRAS-KRIGPAK
- UNIRAS-GIMAGE
- UNIRAS-SEISPAK
- UNIRAS-UNIGRAPH
- UNIRAS-BIZPAK
- UNIRAS-UNIEDIT

Workstation Publishing Software for the IBM RT Personal Computer

Workstation Publishing Software by Interleaf is a document-preparation package. It consists of three main parts: text entry and edit, a diagramming and drawing tool, and a charting tool. Interleaf has combined these three tools into one package with a common user interface. Also included are interfaces for handling with external ASCII files and plot files.

SAMNA+ for the IBM RT Personal Computer

SAMNA+ is a text processing package that includes a spreadsheet function. It provides features found on dedicated word processors, including hyphenation, automatic pagination, math functions, column mode and left/right justify. It also includes Word Base Manager, a function that provides a word/phrase search/abstract creation capability sometimes found in host environments.

SAMNA+ addresses the word processing needs of technical, secretarial, and business professionals. It includes a mail list merge, spelling verification feature, math/Greek characters, three levels of HELP, and a user interface that simplifies operations.

SOLOMON III for the IBM RT Personal Computer

SOLOMON III is designed to be a high-function, integrated accounting package. It has a set of twelve applications. The individual products include:

- SOLOMON III General Ledger
- SOLOMON III Accounts Receivable
- SOLOMON III Accounts Payable
- SOLOMON III Payroll
- SOLOMON III Purchasing
- SOLOMON III Order Entry/Invoicing
- SOLOMON III Job Costing
- SOLOMON III Fixed Assets
- SOLOMON III Sales Analysis
- SOLOMON III Inventory
- SOLOMON III Address and Mail List
- SOLOMON III Database Reporter

Applix IA for the IBM RT Personal Computer

Applix IA is an integrated application package that combines text, graphics, spreadsheet, and database information within a single document. This document retains its ability to be edited. This package has been designed to accommodate information sharing by concurrent terminal users. Other key applications include Freehand DRAW, Business GRAPHICS, Personal DATABASE, Personal TIME MANAGER, Electronic MAIL/MESSAGES, and CALENDAR (meetings and resource scheduling).

IMSL Problem Solving Systems for the IBM RT Personal Computer

The IMSL Problem Solving Systems consist of the following five packages:

The IMSL Library is a collection of over 500 FORTRAN subprograms that offer diverse mathematical and statistical problem-solving capabilities. These subprograms permit users to select tested subprograms rather than writing their own.

The SFUN/LIBRARY is a collection of 180 subprograms that evaluate special functions that arise in applied mathematics, physics, engineering, and other technical fields.

MATH/PROTRAN is a problem-solving system that supports math programming efforts.

STAT/PROTRAN is a problem-solving system that supports linear programming efforts.

LP/PROTRAN is a problem solving system provided to support programming efforts for linear programming problems.

High-level PROTRAN procedures, help files, and error checking are provided to enhance productivity for many problems.

RS/1 for the IBM RT Personal Computer

RS/1 is an easy-to-use, versatile, and fully integrated software system designed to meet the unique data management and analysis needs of technical professionals. With RS/1, analysts have direct control of their data and analysis without writing programs.

RS/1 may be used in a wide variety of applications - from integrated circuit wafer mapping to environmental analysis, laboratory automation to product design, and from basic research to quality control.

IBM 3278 Emulation via the IBM Personal Computer for the IBM System/36

IBM 3278 Emulation via the IBM Personal Computer is a feature of System/36 System Support Programs. It consists of two components. One component executes on the System/36; the second executes on the IBM Personal Computer. 3278 Emulation via the IBM Personal Computer allows users of the IBM Personal Computer, IBM Personal Computer XT, IBM Portable Personal Computer, and IBM Personal Computer AT to emulate an IBM 3278 Display Station Model 2 or 3279 Color Display Station Model 2A or S2A when using the System/36 3270 Device Emulation feature in an SNA network.

The IBM Personal Computer can be attached locally via twinax cable to all models of the System/36 or attached remotely via the 5294 Remote Control Unit with Expansion Feature A, feature 3601, to the System/36 5360 and 5362 System Units.

The System/36 System Support Feature 608 Program supports the 5360, and 5362 processors.

The System/36 System Feature 609 Support Program supports the 5364 processor.

IBM Enhanced 5250 Emulation Program, Version 2.1

The Enhanced 5250 Emulation Program Version 2.1 enables attachment of the IBM Personal Computer, IBM Personal Computer XT, IBM Portable Personal Computer, or the IBM Personal Computer AT to the IBM System/34, IBM System/36, or IBM System/38.

Enhancements include Host Graphics Support, 5292-2 subset, table-driven printer support, keyboard enhancements, serial printer attachment support, Enhanced Graphics Adapter (EGA) support, and Professional Graphics Controller (PGA) support. Current licensees of the Enhanced 5250 Emulation Program will be offered an upgrade to the Enhanced 5250 Emulation Program Version 2.1 for an upgrade charge.

IBM 3812 Pageprinter Font Management System

The IBM 3812 Pageprinter Font Management System lets you customize the 3812 fonts shipped with the printer, change power-on option defaults, add fonts from other diskettes, and create and maintain exclusive fonts for the printer that include special logos, signatures, and graphics. Fonts and graphics are designed by turning on or off every picture element (pel) in a matrix of up to 600 by 600 pels. Other menu-driven options let you copy and maintain a library of fonts, and create or duplicate 3812 Pageprinter system diskettes.

The IBM 3812 Pageprinter Font Management System requires an IBM Personal Computer, IBM Personal Computer XT or IBM Personal Computer AT with 512KB memory, one double-sided diskette drive, a fixed disk drive, an IBM Color Display, IBM Enhanced Graphics Display, or IBM Professional Graphics display with the appropriate adapter. An IBM Personal Computer AT with the above features and a high capacity diskette drive is necessary to maintain the 3812 Pageprinter system diskette.

Sonoran Serif Typographic Fonts for the 3812 Pageprinter

Sonoran Sans Serif Typographic Fonts for the 3812 Pageprinter

Pi and Specials Fonts for the 3812 Pageprinter

The Sonoran Serif Typographic Fonts for the 3812 Pageprinter are the functional equivalent of Monotype Times New Roman (a trademark of the Monotype Corporation, Ltd.). The Sonoran Sans Serif Typographic Fonts for the 3812 Pageprinter are the functional equivalent of Monotype Arial (a trademark of the Monotype Corporation, Ltd.). Typographic fonts were digitized by the Monotype Corporation at 240-by-240 picture elements per square inch.

The Sonoran Serif and Sonoran Sans Serif Fonts are available in four type faces: Roman medium, Roman bold, Italic medium, and Italic bold. Each typeface is available in 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 24, 30, and 36 point sizes. Fifty six fonts are available in all. Each font contains 238 characters to support 11 national languages.

The Pi and Specials Fonts for the 3812 Pageprinter contain both serif and sans serif versions of Pi fonts in medium and bold typefaces, each in 6, 8, 10, and 12 point sizes. Each font contains 189 characters. 24 and 36 point Sonoran Display fonts are included with 81 characters. There is also a 4 point Sonoran Petite font with 133 characters.

The fonts require an IBM Personal Computer, IBM Personal Computer XT, IBM Portable Personal Computer, with DOS 2.00 or later, or an IBM Personal Computer AT with DOS 3.00 or later; the appropriate display and display adapter; and the IBM 3812 Pageprinter and appropriate serial adapter. The IBM 3812 Font Management System is strongly recommended for font downloading, otherwise you must supply functionally equivalent programming. In addition, the IBM Personal Computer AT is recommended if fonts are to be added to the 3812 Pageprinter system diskette.

IBM Personal Decision Series Reports+ Training Edition

The REPORTS+ Training Edition is a computer-based training package that introduces the REPORTS+ Edition of the IBM Personal Decision Series. It is an interactive educational workbook that teaches how to create a printed report with the REPORTS+ Edition.

The Directory Adds 14 Programs

The Directory of Personally Developed Software has developed 14 new low-cost programs. Ranging from $19.95 to $99.95 in price, most of these new programs cost $24.95 and less. Information about the new programs appears in the 1986 Spring Supplement (Volume 2, Number 1) of *The Directory*. You can order *The Directory* or the programs by calling:

1-800-IBM-PCSW

or writing to:

Personally Developed Software
P. O. Box 3280
Wallingford, CT 06494-3280

The 14 new programs are described below.

Communications Family

PC Network Remote Control
PC Network Remote Control lets you operate a remote personal computer as if it were your own. It lets you link multiple personal computers on a network to display a presentation on multiple PCs at the same time, or provide help to someone by watching what they do and typing the correct input from your system. PC Network Remote Control works on any network that is NETBIOS-compatible.

Productivity Family

Daily Organizer
Daily Organizer projects, tracks, displays, and prints your calendar and appointments on a daily, weekly, monthly, or yearly basis. It can store information for dates spanning more than half a century and remind you of appointments even when another program is running. It handles reminders for single and multiple dates, as well as periodic reminders such as mortgage payments. Daily Organizer also can run under TopView.

Introductory Editor
Introductory Editor lets you create, display, edit, and print ASCII files. Start-up is fast and simple because of the online help screen for each command. You can copy, move, and delete blocks of lines. Introductory Editor lets you split and join lines, and save and recall long sequences of keystrokes and information by pressing key combinations. You can search for and locate strings of characters.

JustEdit
JustEdit is a compact, yet flexible full-screen text file editor that resembles editors used on many mainframes. JustEdit shows an entire line on the screen, avoids horizontal scrolling, and automatically does vertical scrolling. It uses simple commands for quick text changes, and lets you reflow paragraphs easily. You can insert, split, join, and delete lines, rename files, and load a new file within an existing file. You can also search, replace, and repeat strings of text.

Programming Family

Source Maintenance System
Source Maintenance System helps you maintain and change programming source code across many files without duplicate effort. It lets you place single or multiple changes in Update Control files, which are then applied against existing source files to create new source files. Source Maintenance System produces a log of all changes to give you a history of your work.

Education Family

Aeromathics
Aeromathics combines the fun of a puzzle with the learning of flash cards to teach math to children ages 7 to 14. Each game begins with an aircraft picture hiding a math problem. You use cursor keys or a joystick to uncover the problem. When enough of the problem is visible, you race the clock to find the answer. You can select from six problem types at four levels of difficulty, and play for bonus points.

Countries, Capitals, and More!
Countries, Capitals, and More! teaches geography through three different games. The Select game lets you choose the correct name currency, or capital of a country or geographical feature. The Compare game lets you choose the country with the larger area, population, or population density. The Locate game tests your knowledge by having you point to specific countries on a map using a joystick or cursor keys.

Electronic Grammar (Parts of Speech)
Electronic Grammar (Parts of Speech) teaches eight different parts of speech (nouns, verbs, etc.) to children and adults. It lets you select which part of speech to study; shows you examples, definitions, and explanations; quizzes you on each topic; and keeps a running score of your progress. You can proceed at your own pace, and page backward for review of the material. You can also refer to help screens as needed.

Memory House
Memory House helps children ages 3 to 7 years of age improve their retention ability. The object of each game is to match shapes, colors, or animals in the doorway of the house with those shown in the four windows. Memory House uses simple menus that reduce the need for help from adults.

Multipurpose Authoring Language
Multipurpose Authoring Language lets you create unique teaching programs that track student responses. Multipurpose Authoring Language lets you use ASCII graphic characters to create pictures and animate objects as part of lessons. You can use text to create captions or ask questions. Multipurpose Authoring Language also lets you add optional equipment (cassette recorder, speech synthesizer, component monitor, and video laser disc player) to add new dimensions to lessons.

Lifestyle Family

Plan-A-Year
Plan-A-Year tabulates your income and expenses totals and shows how much you can spend and still reach your financial goal. It helps you stay within budget by displaying your total income and expenses, financial gain, spending money, and any money you have in special accounts. It lets you track 75 expense items, 15 income items, and 20 special accounts. Plan-A-Year is designed for the new computer user.

SOLITAIRE
SOLITAIRE is the classic card game you used to save for rainy days. Using function keys, you play the "Klondike" version of Solitaire. You even have an option to help you win in difficult situations. You can choose single or multi-pass options to go through the deck of cards.

1985 Tax Template
1985 Tax Template provides a fast and flexible way for most people to complete their 1985 tax returns. It contains nine 1040 schedules and ten other frequently used forms as Lotus 1-2-3 templates. All forms except the 1040 are currently accepted by the IRS as computer printouts. The templates reduce calculation errors and allow automatic transfer of amounts between forms as required. When finished, simply print and submit the forms to the IRS.

Entertainment Family

Side-Swipe
Side-Swipe requires quick thinking and reflexes as you guide your men through a maze of corridors with fixed and moving escape hatches and deadly creatures. You win points and men for each successful escape. You can adjust difficulty and speed levels to suit your skills. Joysticks or cursor keys let you control play.

Editor's Comments

How to Obtain *Exchange*

Frequently we are asked how to obtain *Exchange*.

Exchange is circulated primarily through IBM Personal Computer user groups. We distribute *Exchange* at no charge to several hundred PC user groups that have registered with us for our support. In turn, these user groups distribute *Exchange* at their group meetings. Therefore, one way to obtain *Exchange* is to become an active member of a PC user group. In addition to receiving *Exchange*, you will undoubtedly reap other benefits, because user groups provide focal points for PC information, support and camaraderie. If you are interested in locating an existing PC user group, registering your existing group with us, or forming a new user group, please contact us on 1-800-IBM-PCUG.

Exchange also is available at a nominal cost through the IBM Distribution Center in Mechanicsburg, Pennsylvania. To order through Mechanicsburg, readers who are not IBM employees should contact the librarian at an IBM branch office. IBM employees should submit their orders directly to Mechanicsburg.

Individual issues of *Exchange* should be ordered using the ITPS Publications Order Form. Form numbers for individual issues are:

Issue	Form Number
June 1985	G320-0842
July 1985	G320-0843
August 1985	G320-0844
September 1985	G320-0845
October 1985	G320-0846
Nov/Dec 1985	G320-0847
Jan/Feb 1986	G320-0848
Mar/Apr 1986	G320-0849

Subscriptions to *Exchange* should be ordered using the System Library Subscription Service (SLSS) form. Under Order Number, specify GBOF-1229. To receive all back issues, check the Ship Initial Library option.

Please note that we in Boca Raton are unable to handle requests for individual issues or subscriptions to *Exchange*. We distribute *Exchange* only in bulk to IBM PC user groups and the IBM Distribution Center.

I'd like to close by thanking all of you for your interest in *Exchange* and for your kind comments and encouragement.

Michael Engelberg
Managing Editor

Copyrights, Trademarks and Service Marks

"The IBM Personal Computer file system...repackages the basic marvels of magnetism into a form useful for storing data in computers. (page 4)

"...the PC takes very unkindly to attempts to infringe on its space. (page 5)

"The IBM PC Network Analysis Program provides a set of functions to help you manage your network. (page 7)

"Fundamental to the Virtual Device Metafile notion is the ability to conveniently archive graphic images... (page 19)

"Because pictures are stored in a device-independent manner, they can be ported to and shared with other computers... (page 20)

"The Graphics Development Toolkit includes a Virtual Device Interface that follows the proposed ANSI X3H33 definition. (page 27)

"You can influence the learning process of your children by helping them make better use of computers at home... (page 33)

"Boolean operators can also work on single bytes of information to filter out or modify single bits. (page 37)

G320-0849-00