Exchange

November/December 1985

Hardware

Software

Random Data

Departments

INSIDE: Pull-out Printer Code Reference Chart

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
User Group Editor	Steve Mahlum
Editorial Assistant	Wayne Taylor
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 (2900)
P.O. Box 3022
Boca Raton, FL 33431-0922

To correspond with *Exchange*, please write to:

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

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

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

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

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

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

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

IBM's New Proprinter: Better By Design

George Gynn
North East Indiana IBM PC Club

Design

The new IBM Proprinter is an excellent example of a printer that hasn't been trapped by the design criteria of its predecessors. It is the first low-cost printer that hasn't left me with the impression that the designer thought he was supposed to make an electric typewriter without keys. The Proprinter's design is characterized by the use of "engineered materials" that have particular physical properties. Where the properties of metal are needed, metals are used. Where the specific physical properties of plastics are needed, plastics are used. (The consideration here was not primarily cost savings, but the need for certain characteristics more appropriately explained in an applied plastics design magazine.)

Fasteners

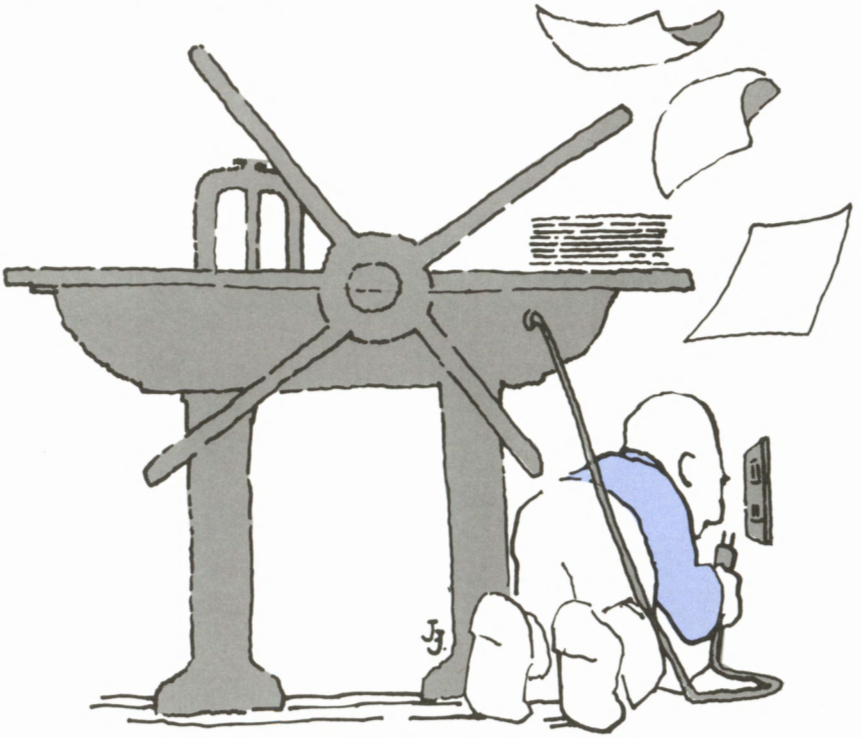
If anything characterizes this quality product, it can be summed up in the term "fasteners." Fasteners are all the little nuts, bolts, screws, washers, clips, and hundreds of other items that take lots of time (labor) to assemble—just to hold things together. Not only are parts assembled with fasteners full of holes, but all mating parts have to have tolerances to allow for the insertion of these fasteners. Then everything has to be adjusted. That takes time, and that is one reason why it is impossible to build a low-cost product with designs using lots of fasteners.

The Proprinter's designers minimized the use of fasteners by designing component parts that snap into each other. By using engineered material, they have slashed labor cost, slashed material costs, and improved performance and reliability in the process. They have incurred (one time only) higher initial development costs, product tooling costs, and engineering development costs in order to get long-term cost savings in assembly labor and materials. These savings have been passed on to the consumer.

To illustrate the advantages of the Proprinter's design, let me explain how easy it is to access the insides of the printer. The following steps are detailed in the *Technical Service Manual*, but not in the *Guide to Operations* manual that comes with the printer. I include these steps to illustrate my point, not to tell you how to service Proprinters.

To access the insides, I first remove the paper drive knob, and the two cables, cover, and ribbon. I then turn the unit on its back, insert a screwdriver into the slots, push it a little to the side (no twist please—no screws), then snap, snap, snap, and the cover lifts off.

Now I want to get to the printhead.



I unsnap the front printhead guide bar, unsnap the printhead cable, unsnap a little retainer from the rear guide roller (this step not included in the manual), and unsnap the rear guide bar. I then unsnap the printhead carrier from the drive spiral and unsnap the printhead from its carrier. (I've been a little loose in my use of unsnaps, but if you had watched, you wouldn't have argued.)

With the cover off, the printhead off, and the guide bars out, you can look around. See any screws, nuts, bolts, washers, and the usual collection of little snap rings that get lost or break? You might see a few, but only a few. To reassemble the printer, just reverse the process. It's a snap!

This extremely easy and quick disassembly and reassembly also should serve to minimize maintenance cost (should any ever be necessary), simply because the technicians are not wasting time with a bunch of fasteners and the problems they cause.

During your initial inspection of the Proprinter, have the person showing the printer to you remove the paper and the printer ribbon, unplug the power and data cables, and leave the printhead cover off the unit.

Immediately you will see that this isn't your ordinary dot matrix printer. Traditional dot matrix printers are characterized by a number of things. For one, the paper advance knob on the side of the printer rotated a large roller, one of the features carried over from typewriters. But the Proprinter doesn't use one. The tractor-feed paper is advanced by the tractor feed mechanism over the top of the printing platen.

Metal Platen

Metal? Yes, you read it right—the wires from the dot matrix print head work against a metal alloy platen. And why not? The Proprinter isn't mechanically throwing a massive type key at the paper like grandma's vintage typewriter, which needed a rubber roller to keep the type from perforating the paper.

A dot matrix wire is a delicate thing with very low mass, which accounts for its speed. Firing a little pin or needle at pieces of paper backed up by a rubber roller is fine for a single sheet at a time, but poor for two sheets or more.

Have you ever seen a fourth carbon copy from a dot matrix printer that used a rubber roller platen? Could you read it? But with a metal platen, where the only spongy material is the paper itself, four carbon copies come out clearly legible. I think the better quality is well worth the slight increase in noise that the metal platen makes.

No Wasted Sheets

Another delightful feature of the Proprinter is that the metal alloy print platen has the print columns cast into it. And because the printing starts about three lines down from the top of the platen, it acts as a nice tear bar when removing tractor-feed paper. I no longer have to reposition the paper after I've torn off printed copy, and at last I have a printer that doesn't waste an extra sheet of paper at the end of every print job. Even with the removable dust cover on the printer, I can still tear off the printed copy without losing a sheet.

Finally, the metal platen leaves everything out in the open, not hidden behind a massive, non-removable, rubber roller. This is particularly convenient when the tractor-feed paper inevitably jams. On printers with rollers, the paper can jam so tightly that you practically have to take the printer back to the shop to have it removed. With the Proprinter, however, the separate tractor-feed parts easily snap out from the top of the printer, leaving exposed the parts in which torn paper pieces can get stuck. It makes the printer extremely easy to clean and get back into service quickly.

Single-Sheet Feeding Made Easy

If you use tractor-feed paper for routine PC work and use fancy single-sheet paper for correspondence, you've got a thrill coming. Suppose your Proprinter is loaded with regular tractor-feed paper. Suddenly you want to print out a letter on 24-pound bond paper. Just insert that high quality sheet in the front, tilt the paper release lever forward, and slide in the sheet. It goes right over the top of the tractor-feed paper. After you're done printing, just roll the single sheet of tractor-feed paper back to the beginning. You've printed out your letter without having to remove and reload your tractor-feed paper. This method works beautifully for addressing an envelope, too.

Printing

Print Registration

One of the weaknesses of many small printers is the use of composition fabric, rubber cogs, or tooth belts to drive the print head and advance the printer ribbon. Belts stretch, are delicate, and age. The IBM Proprinter does not have this weakness. Everything is direct gear drive, including the printhead. If you look under the

printer ribbon, you will see the head drive helix. It's a spiral made of a plastic composite material. Watch it run the printhead back and forth, then look at the print registration, especially vertical lines. With the Proprinter, the vertical lines are perfectly registered, not wavy as is usually the case with a tooth-belt drive.

Print Speed

Many printers rave about their printhead printing rate, but did you ever notice how they seldom talk about "throughput rates"? My old dot matrix printer printed characters at an acceptable rate, but the paper feed just poked along. Nothing is more frustrating than having your printer take more time moving from one form to the next than it takes to print the form. Not so with the Proprinter. It can feed paper at three inches per second, which means it takes roughly four seconds to pump out an 8.5-by-11-inch sheet.

Print Quality

The print quality is as good or better than any other dot matrix printer I have ever seen at any price—even up to twice that of the Proprinter. But, the proof is in the printing, and you'll just have to see it to appreciate it.

Print Features

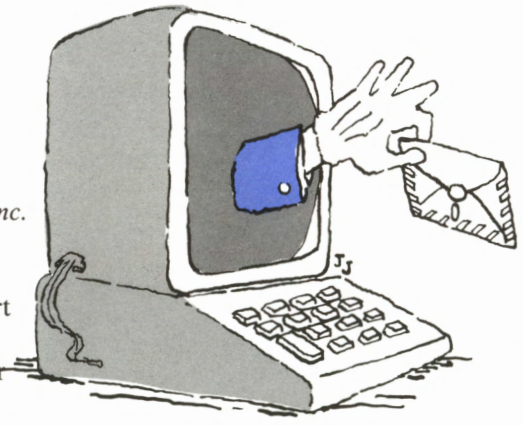
The Proprinter has all the usual print features and fonts you would expect from a dot matrix printer, including underlines, overscores, math symbols, international symbols, subscripts, and superscripts. Additionally, it comes with alternate fonts, and the ability to download user-defined characters. In the printer's memory, you can store up to 94 of your own characters or symbols. The Proprinter also does bit image graphics in 480, 960, and 1920 formats. (By "format" I mean dot columns on an eight-inch line.)

The Proprinter works fine with the codes for the IBM Graphics Printer, which the Proprinter is replacing. Simply run your software's printer installation programs and tell it you have an IBM Graphics Printer.

Finally, the best news always comes last. The suggested retail price for the IBM Proprinter is much below what you would expect to pay for all these features.

Disk Error Messages

Francis X. Bolton
New York Personal Computer, Inc.



Occasionally, your computer's fixed disk or diskette drives start behaving in a strange way, and you seem to have lost control or lost data. You also encounter error messages you've never seen before. These "greetings," as I call them, identify unwanted problems.

One such message is "Disk boot failure." Another is "Unrecoverable xxxxxx" (there are a variety of such messages, all signifying similar problems). Let's briefly examine some of these "greetings."

Note: No matter which error you receive, always try to make copies of the faulty diskette, and *never* try to write anything to a faulty diskette that may have files which are not backed up.

Disk Boot Failure

You may encounter the message "Disk boot failure" when you start your computer using either a cold boot (turning on the power switch) or a warm boot (restarting the computer using Ctrl + Alt + Del), or when the disk or diskette you have just been using becomes unreadable. Every disk and diskette formatted by the DOS FORMAT command contains a disk boot record. The disk boot record is the first thing read into memory.

A formatted disk may or may not contain the DOS operating system. If your disk does not contain the three DOS files— the two hidden files, IBMBIO.COM and IBMDOS.COM and the COMMAND.COM file— but you try to boot from it (i.e., load DOS from the disk into the computer's

memory), you will get the message "Non-system disk." However, if your disk does contain the DOS operating system, the boot process will continue, and DOS will be loaded from the disk into memory. If for some reason the DOS loading process fails, you will get the message "Disk boot failure."

This error message tells you, there may be an error in one of the DOS hidden files which prevents the operating system from loading. From my experience, if something is wrong with the hidden files, you won't get a message; instead, you will get gibberish on your screen. In either case, all other data on the disk is probably perfectly sound.

If you encounter the "Disk boot failure" message when booting from a diskette, make a copy of the diskette using the DISKCOPY command (two copies is not a bad idea), and then put the new copy in drive A and reboot. The boot record of the target diskette is created before the data is copied to it, so the new disk should boot normally, and all your data should be sound.

If you are interested in what this all means, read Chapter 1 of the *DOS Technical Reference*. Despite the manual's presumption that you should know 8088 architecture, interrupt mechanisms, and instruction sets, the first several pages referring to DOS structure and DOS initialization are relatively easy to understand.

Unable to write boot: If you encounter this message when formatting a diskette, it means that the first track of the diskette or DOS partition is bad, and the boot record cannot be created. The diskette or DOS partition is not usable. If you are using a new diskette, return it to your dealer to get another one; if you are using an old diskette, throw it away (I assume you didn't want any of the data on it since you were formatting it when the error occurred). If you encounter this problem on a fixed disk, create a new DOS partition because the current DOS partition is not usable.

Unrecoverable read error on source: This message can make your heart sink. It occurs only with the DISKCOPY command, and informs you that the data on your source diskette could not be read. Possibly your source diskette has some form of copy protection; if so, this message is the normal result, and there is nothing wrong with your diskette. However, if there is no copy protection, the files on your source diskette may have unrecoverable errors. Should you take this seriously? Yes. And you should probably take it more seriously if your files consist primarily of numbers instead of words, because errors within words are more readily recognizable than errors in long lists of numbers.

If there is an unrecoverable read error on the source diskette, its data probably was not copied onto the target diskette, or, if it was copied, the data on the target diskette may be incomplete. Occasionally, I have carefully checked all the data on the target diskette after encountering such a message, only to discover that nothing I could detect was awry. This does not mean that I can ignore the error message; it simply means that I was lucky enough not to have lost pieces of my files.

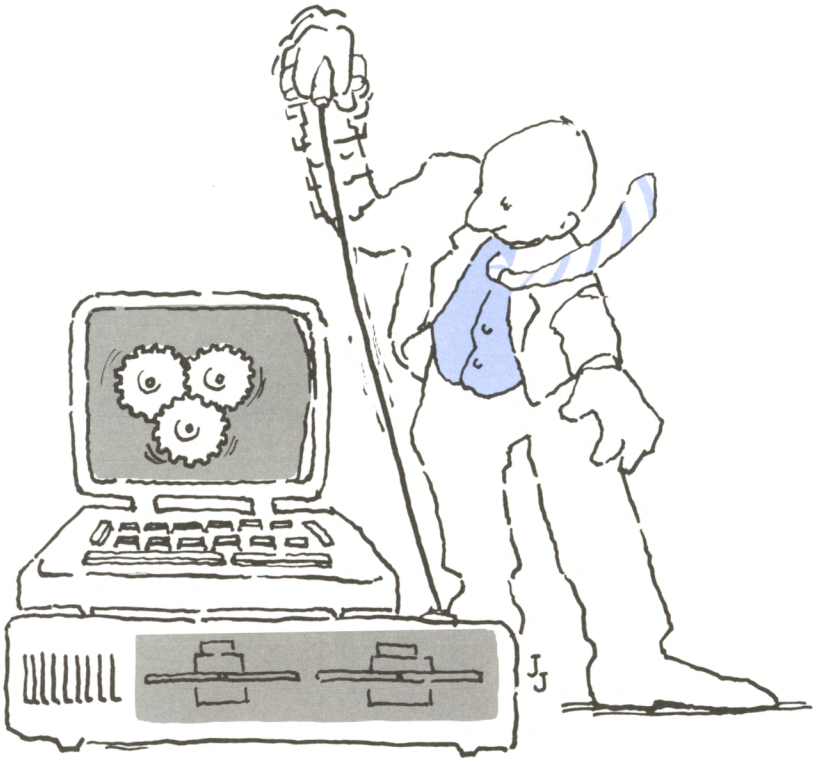
Other messages are "Unrecoverable format error on target" (encountered when using DISKCOPY in DOS 2.10; DOS 3.10 gives the message "Format failure"), and "Unrecoverable write error on target".

If you get an unrecoverable format error message when using DISKCOPY, the problem encountered is in formatting the target diskette, and the warning is given before your data is copied. Don't try to use the target diskette, since it contains no data. Instead, try formatting the target diskette using the FORMAT command, or insert another diskette and start DISKCOPY again.

If there is a write error on the target diskette when using DISKCOPY, all data will have been sent to the diskette, but some data may have been lost due to bad sectors on the target diskette. The message may say that DISKCOPY couldn't verify the write operation on the target diskette. This implies that the write operation took place, but the target diskette may contain incomplete files. When you make a copy using DISKCOPY, sectors are copied, not files, so you may lose pieces of data from several files. Thus, all the files may appear in your directory on the new diskette, but some may be incomplete.

DOS Startup

Bob Leder
Kentucky Indiana PC Users Group



When you flip the PC's power switch, the power supply unit verifies that it can supply voltage at the proper operating levels for the machine. Once the power has stabilized, the power supply provides a "power good" signal to the central processing unit (CPU) board. Only when the CPU receives the appropriate signal from the power supply will it begin executing a program called the basic input/output system (BIOS) that is stored in the read-only memory (ROM) on the CPU board.

This BIOS verifies that the main board is operating correctly before it checks for other ROM programs in the computer. If BIOS finds another ROM program, it reads it, and if instructed, will pass the control of the CPU to the program contained in the ROM chip.

Disk controller cards contain a ROM program that instructs the CPU to read from the fixed disk or the diskette. If the system does not contain any other ROM programs, the CPU will try first to read the diskette and then the fixed disk, looking for the boot record and the BIOS interface. If no diskette is in the A drive and the fixed disk has not been set up to load the Disk Operating System (DOS), IBM systems will load the BASIC language interpreter contained in ROM.

When you use the disk operating system (DOS) to format a diskette, DOS creates a boot record at track 0, sector 1; on a fixed disk, the boot record resides on the first sector of each DOS partition. The boot record tells the ROM BIOS whether the disk contains the BIOS interface (IBMBIO.COM), the DOS Program File (IBMDOS.COM), and the command processor (COMMAND.COM). Disks that do not contain these three files are called non-system disks. If you try to start your system using a non-system disk, the boot record will tell the ROM BIOS to send the error message "Non-system disk or disk error."

When the ROM BIOS reads the boot record and verifies that the disk is a system disk, it checks the disk's File Allocation Table (FAT) for the two hidden files mentioned above, IBMBIO.COM and IBMDOS.COM. When these two files have been loaded into RAM, control of the CPU is passed to IBMBIO.COM. This interface program knows how to talk to the devices in your computer. It determines which ones are present and resets all of the peripherals (printer, disk, video interface, etc.) in the system. This is the program that reads the CONFIG.SYS file. When all of the initial startup has been done—device handler loading and reserving space for disk buffers and file handlers—IBMBIO.COM moves IBMDOS.COM to overlay this initialization sequence and passes control of the CPU to

IBMDOS.COM, the DOS Program File.

DOS initializes its workspace and calls IBMBIO.COM to load the command processor (COMMAND.COM). The command processor is divided into four parts:

1. a resident portion for program termination and exception handling (disk error, Ctrl+Break, etc.)
2. an initialization portion
3. a transient portion for internal command handling
4. a portion containing the program loader.

In the command processor supplied under the IBM logo, the last two sections reside in high memory and the other sections reside in low memory. (In DOS 3.00 and 3.10, the program loader portion of COMMAND.COM is part of the resident portion of DOS and, as such it loads in low memory.) This may not be true for other versions of Microsoft's DOS. When the command processor starts up, it looks for a BATch file named AUTOEXEC.BAT. If this file is found, each BATch command in it is executed.

The initialization portion of DOS may be overlaid by the first program loaded into memory after the initialization process. When the program exits, DOS uses a checksum methodology to check whether the transient portion of DOS (the initialization portion) has been overlaid. If it has, COMMAND.COM is reloaded.

Locating Text Easily

Steve Mahlum
IBM Corporation

Editor's note: This article is adapted from Doug Chamberlain's article, "Easy Text Locating," which first appeared in PC Report, *a newsletter of the Boston Computer Society IBM PC Users' Group.*

You may be interested in a technique you can use to search all the files in a disk directory and report which files contain a specified text string up to eight words long. You can try it in two minutes or less. With a little modification, it can search a specified number of subdirectories without having to be entered more than once.

The technique uses the DOS FIND program from within a FOR statement in a BATch file. The BATch file, FINDER.BAT, is short, so it loads fast and runs fast, especially if you use a virtual disk (see Chuck Gaston's article, "The Virtual Disk Drive: Fleet and Silent" elsewhere in this issue). FINDER.BAT contains the following lines:

```
echo off
for %%f in (%1) do find
  %2 %3 %4 %5 %6 %7 %8 %9 %%f
```

The first line simply tells DOS not to display the BATch commands on the screen as they are processed. The second line contains the FOR command, which repeats the FIND command for every file in the set defined by the first parameter passed to the BATch file.

To execute the BATch file and search every file in a specified subdirectory, enter the command in the following format:

finder c:\text*.* "Search for this specific string of text."

where

finder — is the command that evokes the BATch file FINDER.BAT

c:\text*.* — is the path and name of the subdirectory you want to search. The *.* tells it to search every file in the subdirectory.

"Search ..." — is the string of text you are trying to find. The string *must* be enclosed in quotes. If a quote mark is part of the text you are trying to find, you must enter it using two quote marks in a row (""). This string of text may be up to eight words long.

If you would like to use any of the parameters of the FIND command (/c, /n, /v), you can enter them as the second variable in the command string. For example, if you wanted to display the relative line number of each matching line, you would enter the /n parameter as follows:

```
finder c:\text\*.* /n
  "Search for this
   specific string of text."
```

Or, to find all lines that do not contain the text string and list their relative line numbers, you would enter the parameters /n/v:

```
finder c:\text\*.* /n/v
  "Search for this
   specific string of text."
```

Note, however, that when you use the FIND parameters, you limit your text string to seven words. When you specify the subdirectory to search, you can use any combination of filenames and the global filename characters *, ?. For example, to search for a specific number in all the files in the DB subdirectory that have the file extension .dbf, and to display the relative line number of the matching lines, the command might be:

```
finder c:\db\*.dbf /n "123,479"
```

If you want to save the output of the search, you can redirect the output of the search to a printer or a file (or other device) using the redirection command: > or >>. You must enter the redirection command in the BATch file, not from the command line. To redirect the output to the printer, modify the FOR command in the BATch file as follows:

```
for %%f in (%1)
  do find %2 %3 %4 %5
  %6 %7 %8 %9 %%f >lpt1:
```

To redirect the output to a file, you will need to use the >> symbol so that the subsequent searches done using the FOR command will append to the searches it has already done. Modify the FOR command in the BATch file as follows:

```
for %%f in (%1) do
  find %2 %3 %4 %5 %6 %7 %8 %9 %%f
  >>d:\path\filename.ext
```

You may find that you search only within a few specific subdirectories where you store most of your text files. You can search more than one subdirectory, but you must have the paths and names of the subdirectories listed within the BATch file:

```
echo off
for %%f in (c:\text\*.*
  c:\info\*.*  c:\ws\*.doc) do
  find %1 %2 %3 %4 %5 %6 %7 %8
  %9 %%f
```

Because you have the names of the subdirectories entered in the BATch file, you no longer need to enter a subdirectory name when you enter the command at the DOS prompt. The first variable in the command could be part of the text string, allowing you to have a text string nine words long (or eight if you use any of the FIND parameters):

```
finder
  "If only I could find
   what I'm looking for"
```

If you are searching for a specific string of text among several diskettes, you could imbed a PAUSE and GOTO command in your BATch file:

```
echo off
:begin
for %%f in (%1) do
  find %2 %3 %4 %5 %6 %7 %8 %9 %%f
echo Press Ctrl-Break to end or
pause
goto begin
```

The first variable entered in the command would be a:*.* or b:*.*. If you enjoy tinkering with BATch files, you could even create one that would alternately look on the A drive and B drive for even faster searching on diskettes. After using this FINDER.BAT for a while, I found that sometimes I wanted to search a single subdirectory, while other times I wanted to search a series of subdirectories.

Rather than setting up several BATch files, I used the GETFUN.COM program* and a single BATch file to create a menu from which I could choose to search among several combinations of subdirectories, either one at a time or together.

Keep in mind that the DOS FIND command does not strip the high bit (eighth bit) off each byte, so if your text files were created using a word processor that uses the eighth bit, you may not be able to use the FINDER.BAT file to do searches.

***Note:** The GETFUN.COM program and information about setting up menus in your BATch files are found in Bill Weil's article "Keyboard Input for BATch Files" in the September 1985 *Exchange*.

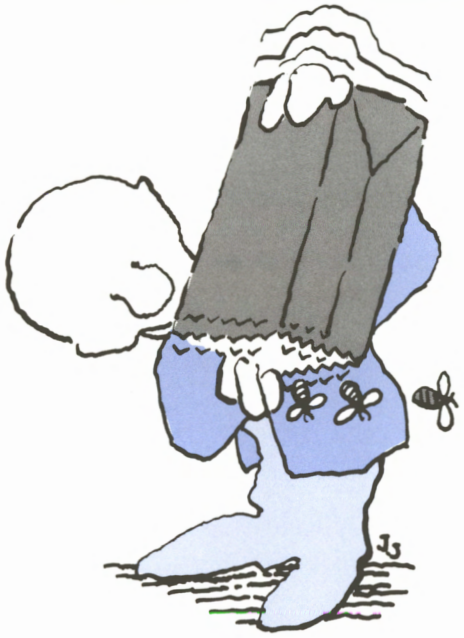
Exploring the Resident Debug Tool

John Warnock
IBM Corporation

The Resident Debug Tool (RDT), a major component of IBM's Professional Debug Facility, gives programmers the information and control they need to develop and test their programs. RDT, a full-screen interactive program, provides immediate, useful information for debugging assembly language programs and assembly language listings generated by the IBM language compilers.

When loaded, the RDT becomes a resident extension of DOS and acts as an interrupt handler that controls the execution of an application program. The program may issue certain interrupts that cause RDT to be invoked. You can then enter RDT commands to perform a desired debug function. RDT also handles hardware-based interrupts. The Professional Debug Facility comes with a Non-Maskable Interrupt (NMI) card that occupies a Personal Computer expansion slot and allows the programmer to activate the RDT in situations that would otherwise lock up the system.

Two major features of the RDT enhance programmer productivity. One feature is RDT's use of windows. Program information is continually displayed on the screen. Once you have set parameters such as memory locations, you need not issue new commands or reissue old ones to see your program's status. The second productivity feature of RDT is its ability to support both the application screen and the RDT screen at the same time. The two screens can alternate on the same display or be shown continuously on different displays.



Other functions of RDT include:

- Stop/start program execution
- Single/multiple step program execution
- Tracing program instructions to a buffer while stepping
- Displaying/altering processor registers, memory or I/O space
- Address stops (breakpoints)
- Searching memory contents to find an ASCII, EBCDIC or hexadecimal string
- Copying an area of memory from one location to another
- Comparing one area of memory with another
- Instruction disassembly
- Hexadecimal expression evaluation, with scratchpad space for evaluated results
- Stopping at program request
- Disk/diskette display, alter, find, verify
- Printing memory, instruction trace buffer, or disassembled instructions
- Printing the user program screen or debug tool screen

- Displaying Math Co-processor (8087 or 80287) state, if installed

RDT's user interface extends the capabilities of these basic functions. The RDT full-screen display format normally displays many items of value. Several valid commands and valid terms for an RDT expression are part of the normal display. The full-screen display is, to a large extent, self-prompting. Most RDT commands are two characters long in order to minimize keystrokes and to help users remember them.

Hardware and Software Requirements

You need the following hardware and software to operate RDT:

- An IBM Personal Computer AT, IBM Portable Personal Computer, IBM Personal Computer XT or IBM Personal Computer
- 128KB of RAM memory
- One single- or double-sided diskette drive
- An IBM Monochrome Display, IBM Color Display or compatible 80-column display plus the appropriate adapter
- IBM Personal Computer DOS version 1.10 or above

RDT also supports the following:

- Both an IBM Monochrome Display and an 80-column color monitor at the same time
- An IBM Matrix Printer, IBM Color Printer, IBM Proprinter, IBM Quietwriter Printer, or any compatible printer
- A Math Co-processor (8087 or 80287)

Invoking the Resident Debug Tool

After you load DOS, insert the diskette containing the RDT program into a diskette drive and enter the following command:

drive:RDT [options]

where

drive: is the diskette drive containing the RDT program diskette

RDT is the name of the Resident Debug Tool program

options may be any of the following:

a specifies that you want the RDT program to allocate 4KB of its memory space to save either the entire monochrome display buffer or, when you want both your own program and RDT to share the same display, the first 4KB of the color display buffer. The "a" option enables an alphanumeric display page to be saved before debugging and restored afterward. ("a" and "g" are mutually exclusive options.)

c specifies that initially you want the RDT program to use the color display as the RDT display screen. ("c" and "m" are mutually exclusive options.)

g specifies that you want the RDT program to allocate 16KB of its memory space to save either the entire monochrome display buffer or the entire color display buffer. You use "g" when you want both your own program and RDT to share the same display. The "g" option enables an alphanumeric or graphic display page to be saved before debugging and to be restored afterward. ("a" and "g" are mutually exclusive options.)

k Specifies that you want RDT to gain control whenever an INTerrupt 5 is issued. (INTerrupt 5 is normally the print screen interrupt issued from the BIOS keyboard handling routine.) RDT saves the current INTerrupt 5 vector and replaces it with one that causes entry into RDT. The "k" option lets you pass control to RDT when you use keyboard I/O that makes use of the BIOS code. You would therefore press Shift + PrtSc to give control to RDT. When you use option "k", you should never place an INT 5 instruction in your program, because the results will be unpredictable.

m specifies that initially you want the RDT program to use the monochrome display as the RDT display screen. ("c" and "m" are mutually exclusive options.)

n specifies that you want the RDT program to handle a soft INTerrupt 2 (non-maskable interrupt) that is caused by an INT 2 instruction or by a Math Co-processor exception condition.

p specifies that you want to use black and white attributes as the default color attributes when RDT uses a color display. Option "p" make RDT's normal color attribute white on black, its

highlight color attribute intense white on black, and its reverse color attribute black on white. This option is primarily intended for the IBM Portable Personal Computer.

t size

specifies the size of your RDT instruction trace buffer, where size is defined as the (hexadecimal) number of instructions. For DOS 1.10, the minimum number of instructions that you can specify is 20H (32 decimal), and the maximum is D0H (208 decimal). For DOS 2.00 and above, the maximum size changes to 924H (2340 decimal).

x specifies that you want to bypass pressing a key in order to continue past the RDT logo screen. Option "x" lets you invoke both RDT and your own program from within a BATch file.

When you write a program that you intend to debug with RDT, it is advisable to place an INT 3 instruction at the beginning of your program.

If you did not specify the "m" or "c" option when you loaded the RDT program, RDT uses the current display screen as the RDT display. If you want both RDT and your own program to share the same display, RDT swaps the display status as necessary while debugging the program. In addition, if you specified an "a" or "g" option, and your own program is in the corresponding mode (alphanumeric if "a", alphanumeric or graphics if "g"), RDT saves your program's display screen when it takes control and restores your program's screen before returning control to your program. If you did not specify an instruction trace buffer size, RDT creates a default buffer size of 32 (20H) instructions. Within RDT's instruction trace buffer, 28 bytes are required for each instruction. Therefore, a trace buffer capable of holding 32 instructions requires 896 bytes. Similarly, for DOS 2.00 and above, the maximum trace buffer size of 2340 (924H) instructions requires 65,520 bytes.

Entering the Resident Debug Tool

When RDT is loaded and all of your specified options are valid, an IBM logo screen appears on the selected (or default) RDT display. RDT then asks you to press any key to continue. After you press a key (or after a few seconds if you specified the "x" option), RDT is made a resident extension of DOS. From this point on, RDT resides in low memory just above DOS. Your own programs can then be loaded normally. DOS loads your programs into memory above RDT.

The amount of memory occupied by RDT is determined by the options you specify when you load RDT. If no options are specified, the size of the debug tool is approximately 42KB. The "m" option requires an additional 4KB; the "g" option requires an additional 16KB; and the "t size" requires from 0KB to 63KB additional memory, depending on the size requested. When all of these options are considered, the amount of memory occupied by the RDT ranges from approximately 42KB to 121KB.

When RDT becomes a resident extension of DOS, it saves your own program's INTerrupt 2 (non-maskable interrupt) vector. Next, RDT modifies the INTerrupt 3 (breakpoint) vector, the interrupt 2 vector, and (if the "k" option was specified) the INTerrupt 5 (print screen) vector so that they all point to RDT interrupt handlers. These vectors are now able to give control to RDT when one of these interrupts occurs in your own program. RDT therefore acts as an interrupt handler to control program execution.

An INTerrupt 3 is triggered either by a user-requested breakpoint set within RDT or by an INT 3 (hex CC) instruction inside your program when it is executing. An INTerrupt 2 can be classified as either hard or soft. A hard INTerrupt 2 occurs when there is a memory parity error or I/O channel check; it also occurs when you press the NMI button on the hardware card that accompanies RDT. A soft INTerrupt 2 is triggered either by a Math Co-processor exception condition or by an INT 2 (hex CD02) instruction inside your program when it is executing.

When you write a program that you intend to debug with RDT, it is advisable to place an INT 3 instruction at the beginning of your program. This will give control to RDT immediately after DOS has loaded it. You can then set breakpoints in the RDT program and can use any of the other RDT facilities. When you are not using RDT, you can leave the INT 3

instruction in your program, because DOS ignores the INT 3 instruction.

Resident Debug Tool Display Screen

The full-screen RDT display has five separate areas, as shown in Figure 1.

The command line is used for entering all RDT commands. The windowing area is used to display/alter memory, display/alter disassembled instructions, display the trace buffer, or display the state of the Math Co-processor (if installed). You can place the cursor anywhere in the input area of the command line, or you can move it to selected parts of the windowing area.

Display Line	Items Displayed
1	Release Number, Title, Release Date
2-9	Breakpoints and Scratchpad Variables
	Processor Registers and Current Instruction Disassembly
10	Command Line
11	Message Line
12-25	Windowing Area
	Memory, Disassemble, Trace Partial, Trace Full or Math Co-processor

Figure 1. RDT Display Screen Layout

Figure 2 shows how the RDT display may appear when RDT first gains control after it encounters an INT 3 instruction while executing a program. All processor registers, address stops (Sn) and variables (Vn) are displayed in the top portion of the screen. The 8088 processor flag register is displayed in both hexadecimal and binary forms. The 8088 segment registers are displayed in 20-bit "real address" representations. A segment register is only 16 bits long, corresponding to the first four characters of the segment register's displayed value. However, most calculations using segment registers involve their full 20-bit values; therefore, a "0" (zero) is appended to the contents of the 16-bit segment registers.

The center of the screen contains the command line, followed by the status (message) line.

Initially, RDT is in memory windowing mode. In this mode, the bottom portion of the screen contains the memory window area. In this area, line numbers, addresses and hexadecimal representations appear on the left side, while character (ASCII or EBCDIC) representations appear on the right side.

When RDT is first entered, there is only one memory window, 14 lines long, with each line displaying 16 bytes of the memory occupied by your program. Since no windowing activity has yet taken place, the first line of the memory windowing area contains the address 0000H. Also, no breakpoints or variables have been set, so they appear simply as dots. The values displayed in the 8088 processor general purpose registers are those that were present when your program issued the INT 3 instruction. The EX field is a pseudo 20-bit program counter maintained by RDT. It contains the sum of the CS and IP registers, followed by a one- to seven-byte hexadecimal memory display of the next instruction to be executed.

Above the hexadecimal display is a disassembled representation of the instruction. If the instruction has an operand that references memory, the 20-bit memory address is calculated (segment register + displacement base/index register(s)) and displayed as the OP variable. If the instruction has an operand that references a particular byte or word in memory, this byte or word is displayed under the OP location.

The undisplayed variable IL (Instruction Length) contains the length of the instruction. The Flag Register (FL) has several bits that are architecturally undefined. However, the OF, DF, TF, SF, AF, and CF bits are zero and the IF, ZF, and PF bits are one, and these bits are displayed in both binary and 16-bit (F246) forms. The default step count is set to one.

The "D1" in the upper right corner of the display denotes RDT display screen number 1. RDT has nine debug display screens. Some variables are common to all nine displays, and some are unique to individual displays. For example, the processor registers and memory contents are common in all display screens, whereas the displayed variables (V1-V9,

```
REL 1.00          IBM PERSONAL COMPUTER RDT   D1        07/01/84
V1:..... V2:..... V3:..... V4:..... V5:..... V6:..... V7:..... V8:..... V9:.....
S1:..... S2:..... S3:..... S4:..... S5:..... S6:..... S7:..... S8:..... S9:.....
                                          DISPLAY: ASCII     WINDOW: MEMORY
AX:  0000   BX:  0000   CX:  00FF   DX:  0000            TR:00  .....-.....
SP:  0200   BP:  0000   SI:  0000   DI:  0100     FL:F246 OF:0 DF:0 IF:1 TF:0
CS: 112E0   DS: 111E0   SS: 112F0   ES: 111E0        SF:0 ZF:1 AF:0 PF:1 CF:0
    LC:....          INT     3                                      OP:.....
    IP:0000     EX:112E0    CC              STEP CT: 0001  CO:.....
==> _

L1   * 00000   4331E300   3F017000   71040E06   C3040E06     *C1....p.q.......*
L2     00010   3F017000   CC040E06   23FF00F0   23FF00F0     *..p.............*
L3     00020   A5FE00F0   96070E06   23FF00F0   23FF00F0     *................*
L4     00030   23FF00F0   600700C8   57EF00F0   3F017000     *........W..p....*
L5     00040   65F000F0   4DF800F0   41F800F0   560200C8     *e...M...A...V...*
L6     00050   39E700F0   59F800F0   2EE800F0   D2EF00F0     *9...Y...........*
L7     00060   000000F6   860100C8   6EFE00F0   38017000     *.......n...8..p.*
L8     00070   4BFF00F0   A4F000F0   22050000   000000F0     *K...............*
L9     00080   FB0BE300   80014205   8C024205   99024205     *.....B...B...B..*
M1     00090   E2044205   D414E300   2115E300   E727E300     *...B............*
M2     000A0   070CE300   26017000   00000000   00000000     *......P.........*
M3     000B0   00000000   00000000   6D034205   00000000     *........m.B.....*
M4     000C0   EA080CE3   00000000   00000000   00000000     *................*
M5     000D0   00000000   00000000   00000000   00000000     *................*
```

Figure 2. Sample RDT Initial Entry Screen

S1-S9, L1-L9, M1-M5) are unique to each of the nine displays.

A useful debug technique is to use a different display screen for each section of the program you are debugging.

Resident Debug Tool Command Line Input

When RDT is first loaded, all command line alphabetic input is in upper case, regardless of the Shift key position. Numeric and special character keys always have dual case capability. The F3 key may be used to change from single-case alphabetic input to dual-case alphabetic input and back again. The cursor must be on the command line in order to make this change.

Some special key functions are:

F2 Performs a Find ASCII or Find EBCDIC (depending on the setting of the Display variable). Input parameters for the Find command, located on the command line, are arranged in a string that begins at the left side of the command line and ends one character before the cursor position. If the cursor is at the beginning of the command line, F2 searches memory for the search argument used in the previous Find command.

F3 Switches between single- and dual-case alphabetic input modes.

F5 Erases the entire command line and places the cursor at the beginning of the command line.

F6 Erases the command line beginning at the cursor position and continuing to the end of the command line.

F7, F8, F9
Restores the user command that was originally saved with F7, F8 or F9 to the command line. If no command was saved, this key has no effect.

F10 Retrieves previously entered commands to the command line. The first time you press the F10 key, the most recently entered command returns; the second time, the next most recently entered command returns; and so on. The ten most recently entered commands can be retrieved.

Command Processing

When you press the Enter key in RDT, the command line is processed. Command processing begins with the leftmost command on the command line and continues with the remaining commands. Commands must be separated by semicolons. Command line processing stops when a command brings back a return code greater than zero, indicating an error condition.

The command line can contain as many commands as will physically fit. Commands may be optionally preceded as well as followed by one or more blanks, although no blanks are required. Only one restriction prevents certain commands from being strung together in a sequence: If the F7, F8, or F9 (Store User Command) command appears in the command line, it will be the last command executed on that line, because the rest of the line is treated as the user commands to be stored.

Command Parsing

The format of the command line is:

```
/ cmd = opnd ; cmd = opnd ;
  cmd = opnd ; ...
```

The beginning slash is optional. If the slash is present, the command line will not be erased after it is processed.

Individual commands are always two characters in length. The last command can be followed by a semicolon, which permits using the Tab key to position the cursor after the last command in order to add a subsequent command.

The blanks and the equal signs are optional. You can save keystroke by using the format

```
cmdopnd;cmdopnd;cmdopnd; ...
```

The operand begins with the first non-blank character other than "=" following the command, and continues either to (and including) the last non-blank character preceding the semicolon or to the end of the command line. If no non-blank characters follow the command, the operand is considered to be null.

Expression Evaluation

The operand of most commands is an expression, defined as one or more terms. A term may be a self-defining hexadecimal, one of the processor registers, or one of the RDT variables.

Terms are always added together. The unary signs "+" and "-" precede individual terms. The unary "+" sign performs no function, but improves readability. The unary "-" sign converts the term to a negative before the addition is performed. Multiple unary signs may precede a term if desired.

Blanks may be used freely between terms in an expression and also can appear between a unary sign and its term.

Terms that are processor registers or RDT variables are always two characters long.

Self-defining hexadecimal terms vary in length. A blank, a unary sign, or the end of an operand must follow a hexadecimal term in order to delimit the end of the hexadecimal term. There is no similar requirement for register or variable terms.

The following are examples of expression syntax:

```
AX = V1 - BX + 4A + AX
AX=V1-BX+4A+AX
AX V1-BX 4A AX
AXV1-BX4A AX
```

All of these commands are equivalent. The RDT variable V1, the negative value of processor register BX, the hexadecimal value 4A, and the processor register AX are added together, and the result is placed into the AX processor register. The first example illustrates optimum readability. The removes the optional blanks following the command and between terms. The third example removes the optional '=' sign and '+' unary signs. The fourth example removes optional blanks. (The blank after the 4A is required to delimit the end of the hexadecimal term 4A.) Although the fourth example is not very readable, you may prefer it because it saves many keystrokes.

The valid processor register terms and RDT variable terms that may be used in an expression are:

```
AH   CH   DI   IL      SS
AL   CL   DL   LC      S1-S9
AX   CO   DS   L1-L9   V1-V9
BH   CS   DX   M1-M9   W1-W9
BL   CT   ES   OP      X1-X9
BP   CX   EX   SI      Y1-Y9
BX   DH   FX   SP      Z1-Z9
```

The variables OP and IL may be used as command operands. For example, the command L1=OP displays the contents of memory at the instruction operand address. The command /IP=IP+IL disassembles a program line-by-line without actually executing the program. (Since the program is not actually executing, the processor registers are not being updated and the OP variable is probably not useful.)

The value of the variables L1-L9 and M1-M5 depend on which RDT windowing mode you are in. In disassemble windowing mode, these variables take on the sum of their respective disassemble windows' CS and IP values. This is an especially useful feature when setting breakpoints. In all other RDT windowing modes, the variables L1-L9 and M1-M5 take on the values they assume in the memory windowing mode.

Comparing the Resident Debug Tool with DOS Debug

John Warnock
IBM Corporation

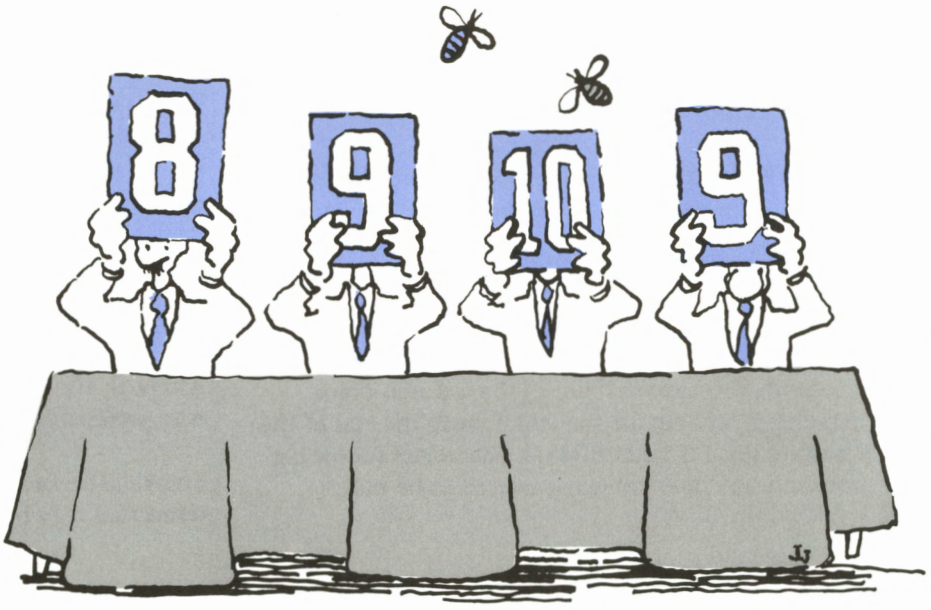
The Resident Debug Tool, a part of IBM's Professional Debug Facility (PDF), lets programmers interactively test and modify programs. The DOS Debug program provides a similar capability. This article compares these two offerings.

Professional Debug Facility Highlights

Greater display flexibility
PDF supports multiple displays. You can run your application on one display while using PDF on a second display. This lets you program and Debug without interruption.

If you have only one display, PDF lets you switch between your application screens and PDF screens. You may also define up to nine PDF screens that can address different segments of your program, and you can set different program breakpoints for each screen. You may allocate memory for screen storage, set attributes for the color display, and tell PDF which display to use when starting.

Screen-oriented displays
PDF provides improved information and usability through its screen-oriented displays. You can switch among four PDF "windows." The Memory window shows system and memory information as your program executes. The Trace window shows the step-by-step instructions executed. The Disassemble window shows your program translated into assembly language statements. The Math Co-processor window displays the state of the Math Co-processor. You can skip to different fields within these windows to change memory values and to enter new commands. Each window presents all of its information in a single screen, eliminating the need to scroll the display.



Greater printer control
PDF gives you greater control of the printer for producing hard copies of your debug sessions. You may print screens, traces, memory contents and disassembled program instructions.

Non-Maskable Interrupt Card
The Professional Debug Facility comes with a Non-Maskable Interrupt card. This card occupies one feature slot in the system unit. It lets you recover from a system lock-up due to a programming error. By pressing the button on the card, you interrupt the system, eliminate the lock-up and return control to PDF.

Support of Math Co-processor
PDF provides detailed information about the activity of the Math Co-processor as your application program executes.

Additional disk service facility
Professional Debug Facility includes a tool called Disk Repair, which lets you examine and modify the contents of a disk or diskette.

User-defined variables for memory addressing, storage, and program breakpoints
You can define, display, and modify up to 14 different memory locations simultaneously. You can describe as many as 18 breakpoints for use with each screen that displays memory or trace options.

In addition, 41 variables are available for scratch storage. They can be assigned the value of another variable or the address of a location in memory. They also can be set as the result of a hexadecimal arithmetic operation.

Greater control through more specific commands
Professional Debug Facility has many specific commands that give you much greater control of the debug process. For example, PDF lets you set both the high and low values of a register instead of only the entire value. It also lets you set breakpoints and memory display locations before you start your application or activate a trace.

Professional Debug Facility has many specific commands that give you much greater control of the debug process.

DOS Debug Program Highlights

Identify and load files by name
The Name command lets you specify a file by drive, path and filename. Debug then creates a file control block for use by the program and by the Load and Write commands.

Move blocks of memory
DOS Debug allows you to copy memory segments using the Move command. By specifying the range and address, you may duplicate memory segments repeatedly. Overlapping moves are permitted.

Load assembly language statements directly into memory
For making program changes while debugging, the DOS Debug program offers a very easy way to make changes. After you type assembly language statements, Debug assembles the statements and loads them into memory at the starting address you specify. Debug also checks for errors.

Perform hexadecimal addition and subtraction
For convenience, the Hexarithmetic instruction displays the sum and difference of any two hexadecimal values you enter.

Comparison of Commands

Table 1 compares commands within DOS Debug and the Resident Debug Tool portion of PDF.

Resident Debug Tool Command	DOS Debug Command	Comments
AF	Register	Sets auxiliary carry flag (AF).
AH	n/a	Alters contents of AH register.
AL	n/a	Alters contents of AL register.
AS	n/a	Displays all characters in memory windows in ASCII format. (DOS Debug displays hexadecimal format only.)
n/a	Assemble	Assembles instructions.
AX	Register	Alters contents of AX register.
BH	n/a	Alters contents of BH register.
BL	n/a	Alters contents of BL register.
BP	n/a	Alters contents of Base Pointer register.
BX	Register	Alters contents of BX register.
CA	n/a	Sets attributes for color display.
CD	n/a	Forces RDT to use color display.
CF	Register	Alters contents of carry flag.
CH	n/a	Alters contents of CH register.
CL	n/a	Alters contents of CL register.
CM	Compare	Compares two memory areas and reports differences.
CO	n/a	Tells RDT the code origin of your program.
CP	n/a	Copies memory from one location to another.
CS	Register	Alters contents of CS register.
CT	n/a	Sets step count for number of instructions to be executed.
CW	n/a	Switches RDT to Math Co-processor window mode.
CX	Register	Alters contents of CX register.
Dn	n/a	Selects active screen.
Dn Dm	n/a	Saves screen m as screen n. Useful for saving several "current" Debug environments.
Dn ON	n/a	Activates breakpoints in screen n.
Dn OFF	n/a	Deactivates breakpoints in screen n.
DF	Register	Alters contents of Direction Flag.
DH	n/a	Alters contents of DH register.
DI	Register	Alters contents of DI register.
DL	n/a	Alters contents of DL register.
DS	Register	Alters contents of DS register.
DW	Unassemble	Switches to Disassemble Window mode.
DX	Register	Alters contents of DX register.
EB	n/a	Displays all characters in memory windows in EBCDIC format. (DOS Debug displays hexadecimal format only.)
ES	Register	Alters contents of ES register.
EX	Go	Transfers control to your program. (RDT allows transfer to DOS when no program is loaded.)
7-F9	n/a	Assigns commands to those function keys.
FA	Search	Finds an ASCII string in memory.
FE	n/a	Finds an EBCDIC string in memory.
FL	Register	RDT alters contents of entire Flag register. DOS requires each individual flag to be set.
FX	n/a	Finds a hexadecimal string in memory.
n/a	Hexarithmetic	Hexadecimal addition and subtraction. (RDT uses variables to accomplish this.)
HP	n/a	Help command, lists all two-character RDT commands.
IB	Input	Reads one byte from designated port.
IF	Register	Alters contents of interrupt flag.
IP	Register	Alters contents of instruction pointer register.

Table 1. Comparison of DOS Debug and RDT Commands

Resident Debug Tool Command	DOS Debug Command	Comments
IW	n/a	Reads one word from the designated port. (DOS Debug reads an entire byte.)
L0	n/a	Sets or clears variables L1-L9.
L1-L9	n/a	Memory address variables for Memory and Disassemble windows. (DOS Debug does not support multiple windows and addresses.)
LC	n/a	Sets pseudo-Location Counter variable.
M0	n/a	Sets or clears M1-M9 variables.
M1-M5	n/a	Memory address variables for Memory and Disassemble windows. (DOS Debug does not support multiple windows and addresses.)
M6-M9	n/a	Scratchpad variables.
MD	n/a	Forces RDT to use Monochrome Display.
n/a	Move	Moves memory block.
MS	n/a	Returns available memory.
MW	Dump	Switches RDT to Memory Window Mode; DOS Debug enters memory display mode.
MW	Enter	Changes memory.
MW	Fill	Changes memory blocks.
n/a	Name	Defines file and parameters.
OB	Output	Writes one byte to designated port.
OF	Register	Alters contents of Overflow flag.
OW	n/a	Writes word to designated port. (DOS Debug writes an entire byte.)
PD	n/a	Prints disassembled instructions from memory to printer.
PF	Register	Alters contents of Parity flag.
PM	n/a	Prints memory in hexadecimal and character format.
PR	n/a	Prints the screen. (Not the same as the DOS Shift + PrtSc.)
PT	n/a	Prints RDT trace buffer.
RD	Load	Reads diskette into memory.
RDT	Debug	Starts the program. Once RDT is resident, you can load a program normally. DOS Debug requires you to name the file or program when you start Debug.
RK	n/a	Restores interrupt 5 vector to low memory vector position.
RN	n/a	Restores interrupt 2 vector to low memory vector position.
S0	Go	Sets or clears S1-S9 breakpoints.
S1-S9	n/a	Program breakpoint variables. RDT allows up to 18 breakpoints in each of 9 screens.
Sn ON	Go	Activates Sn as a breakpoint.
Sn OFF	Go	Deactivates Sn as a breakpoint.
SC	n/a	Replaces the RDT screen with your program screen.
SF	Register	Alters contents of the Sign flag.
SI	Register	Alters contents of Source Index register.
SK	n/a	Sets interrupt 5 vector to give RDT control when an interrupt 5 occurs.
SN	n/a	Sets interrupt 2 vector to give RDT control when an interrupt 2 occurs.
SP	Register	Alters contents of Stack Pointer register.
SR	Quit	Transfers control to system reset entry point in BIOS. In RDT, this causes the system to reboot.
SS	Register	Alters contents of Stack Segment register.

Table 1. Comparison of DOS Debug and RDT Commands (cont.)

Resident Debug Tool Command	DOS Debug Command	Comments
ST	n/a	Executes a designated number of user program instructions.
TC	n/a	Clears RDT trace buffer.
TF	n/a	Alters contents of Trap flag.
TR	Trace	Defines trace activity to be done.
TW	Trace	Switches RDT to Trace Window mode.
UM	n/a	Displays and sets user-defined interrupt mask.
V0	n/a	Sets or clears variables V1-V9.
V1-V9	n/a	Variables for temporary breakpoint storage.
Vn ON	Go	Activates Vn as a breakpoint.
Vn OFF	Go	Deactivates Vn as a breakpoint.
W0	n/a	Sets or clears variables W1-W9.
W1-W9	n/a	Scratchpad variables.
WD	Write	Writes sectors from memory to diskette.
X0	n/a	Sets or clears variables X1-X9.
X1-X9	n/a	Scratchpad variables.
Y0	n/a	Sets or clears variables Y1-Y9.
Y1-Y9	n/a	Scratchpad variables.
Z0	n/a	Sets or clears variables Z1-Z9.
Z1-Z9	n/a	Scratchpad variables.
ZF	Register	Alters contents of Zero flag.

Table 1. Comparison of DOS Debug and RDT Commands (cont.)

Making Music with IBM Personal Computers

Bret Whissel
Florida State University Users Group

Making Your Own Music

If you feel that BASIC's PLAY and SOUND commands do not offer enough flexibility, you can write your own program that accepts music data in any form you desire. You can write programs that will allow you to experiment with different tuning scales or invent your own scale patterns. If you are so inclined, you can even generate random music. (Of course, no one can guarantee the aesthetic value of such work.)

This article discusses how to write programs that play musical melodies on IBM Personal Computers. Making music is a two-step process. First you create a table of musical notes. This table includes note names, note numbers, frequencies and counter values. Second, you write programs that use the counter values to generate musical tones on the Personal Computer.

Before explaining how to create tables and programs, let's review a few concepts about frequencies of musical tones and the relationship between tones, as well as the Personal Computer components that generate musical tones.

Square Waves

On the IBM Personal Computer, music applications use the 8253 Programmable Interval Timer chip. The 8253 has three independent timers, each of which can be programmed for various types of output. Creating music involves only the third timer and the square-wave output.

A square wave is a series of instant state changes (from high to low to high, etc.), as opposed to gradual state changes as seen in a sine wave. When a square wave is applied to a speaker, it produces vibrations or tones.

Frequencies

The pitch of the tone, how high or low it sounds, is determined by how often the square wave changes state. The span of time from the beginning of one peak to the beginning of the next peak is known as one cycle. The term hertz (Hz) refers to the number of cycles per second. For example, 60 Hz means 60 cycles per second. This is known as the frequency of the tone.

Square wave output cannot be heard at very low frequencies. A human ear can hear vibrations as low as 20 Hz (and sometimes lower), but the IBM Personal Computer will produce a discernable tone somewhere around 55 Hz minimum.

The frequency 55 Hz is convenient because the international tuning pitch A[440] has 440 Hz, which is a multiple of 55 Hz. This means you can use 55 Hz, a nice easy number, as the basis of your tuning scale.

Relationship Between Musical Tones

A tuning scale defines the relationship between musical tones. Since the days of Pythagoras there have been several tuning scales, including the mean-tone scale, the well-tempered scale, and the equal-tempered scale. The discussion here is limited to the equal-tempered scale, the scale used by most modern-era composers.

In music an octave represents a change in frequency by a factor of two. Therefore, one octave above A[55] is A[110], and one octave below A[55] is A[27.5]. The factor of two applies throughout the range of audible and inaudible tones. Octaves of A, for example, have frequencies 55, 110, 220, 440, 880, etc.

In the equal-tempered scale, an octave is divided into twelve equal-tempered notes, known as half- or semitones. Each pitch has a logarithmic relationship to the base pitch. The relationship is a fraction of a power of two.

Calculating the Frequency of Any Note

Here's a formula for calculating the frequency of any note. First, assign numbers to every tone of the scale. It is easiest to assign note number 0 (zero) to the base pitch, A[55]. The next pitch, A# (read as A-sharp), is note number 1; B is note 2; and so on. The A that is one octave above A[55] gets note number 12.

We are now ready to look at our formula for calculating the frequency of any note. The formula is:

$$\text{frequency} = 55 \times 2^{n/12}$$

where

55 is the base pitch of 55 Hz

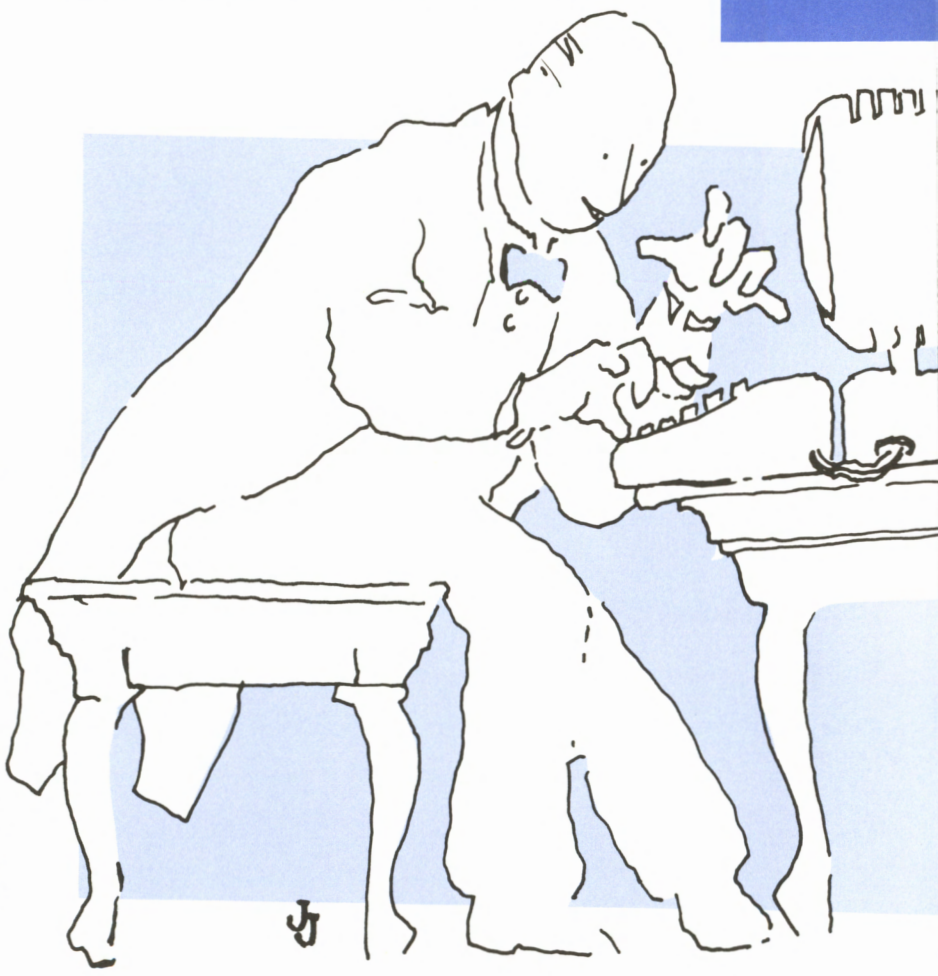
n stands for the number assigned to a particular note.

To see how this formula works for the A one octave above A[55], simply substitute 12 for n:

$$55 \times 2^{12/12} = 55 \times 2^{1} = 55 \times 2 = 110\text{Hz}$$

which we know is the frequency of that particular A.

This calculation can be repeated for any note whose note number is known. Obviously, for most notes,



the result will be a frequency that contains a fraction. For example, the frequency of note 2, a B, is 61.73542. The frequency of note 14, the B one octave higher, is exactly double: 123.4708.

To calculate the frequencies of notes below A[55], simply assign negative numbers to those notes. The note Ab (read as A-flat) below A[55] gets note number -1, the G below that gets note number -2, and so on. The calculation for the A one octave below A[55] is:

$$55 \times 2^{-12/12}$$
$$= 55 \times 2^{-1} = 55 \times 1/2 = 27.5 \text{ Hz.}$$

Programming the 8253 Interval Timer

Once you know what frequencies you want your Personal Computer to produce, you'll need to move from the realm of Pythagoras and Pascal to that of Mozart. First, however, you must perform a few more calculations.

The timers (which are really counters) in the 8253 Programmable Interval Timer chip can produce a change of state as rapidly as 1.19 million times a second. This rate is determined by the clock input pulse.

Using the third timer, which is attached to the speaker, you will divide the clock input pulse to produce a desired frequency. You do this by sending the third timer a counter value **Y**. The timer will then produce one change of state for every **Y** clock input value.

The following formula shows that the clock input pulse divided by the counter value will produce the desired frequency:

frequency =
clock input pulse / counter value

Therefore, using some algebra,

counter value =
clock input pulse / frequency

Creating a Table of Notes

Figure 1 shows a BASIC program that creates and prints a table showing note numbers, note names, frequencies and counter values for the audible range of notes that an IBM Personal Computer can produce.

```
 10  DIM NOTE$(12)  'array to hold
     note names
 20  FOR A% = 1 TO 12
 30  READ NOTE$(A%)  'fill the array
     with note names
 40  NEXT A%
 50  DATA A,A# or Bb,B,C,C# or Db,D,
     D# or Eb,E,F,F# or Gb,G,G#or Ab
 60  LPRINT "Note";TAB(10);"Note";
     TAB(35);"Counter"
 70  LPRINT "Number";TAB(10);"Name";
     TAB(21);"Frequency";
     TAB(35);"Value"
 80  LPRINT "------";TAB(10):"----";
     TAB(21);"---------";
     TAB(35);"-----"
 90  LPRINT
100  FOR A% = 0 TO 65  'generate
     table for notes 0 through 65
110  LPRINT TAB(3);A%;TAB(10);
     NOTE$(A% MOD 12 + 1);TAB(21);
     55*(2¬(A%/12));TAB(37);
     INT(1190000!/
     (55*(2¬(A%/12)))+.5)
120  IF A% = 60 THEN LPRINT:LPRINT:
     LPRINT:LPRINT:LPRINT
130  NEXT A%
140  END
```

Figure 1. BASIC Program to Generate Table of Notes

Table 1 is generated by the program shown in Figure 1. In Table 1, the only required column is the list of counter values, which are used in a later step to create music on the Personal Computer. The other entries in the table are for reference.

IBM Printer Code Reference Chart

Compiled by John Warnock
IBM Corporation

This reference chart may be removed and used for future reference. It includes the following information:

- A description of the print code function
- The ASCII codes needed to generate the function
- A list of printers that recognize and accept the ASCII codes

IBM Printer Code Reference Chart

Description of Print Function	ASCII code	Graphics Printer	Color Printer	Pro-Printer	Color Jet	Quiet Writer	Wheel Printer	Compact Printer	3812 Printer
5 CPI off (by line)	20	x	x	x	x	x		x	x
5 CPI off	27+87+0	x	x	x	x	x		x	x
5 CPI on (by line)	14	x	x	x	x	x		x	
5 CPI on	27+87+1	x	x	x	x	x		x	x
8 CPI off	27+87+0+18	x	x	x	x			x	
8 CPI on	27+87+1+15	x	x	x	x			x	
10 CPI (normal)	18	x	x	x	x	1	x	x	x
12 CPI	27+58		x	x		1	x		
15 CPI (condensed)	15						x		
15 CPI (condensed)	27+15						x		
17.1 CPI (condensed)	15	x	x	x	x			x	x
Alarm	7	x	x	x		x	x		x
Aspect ratio set	27+110+r		x			2			
Auto justify off	27+77+0		x						
Auto justify on	27+77+1		x						
Auto LF off	27+53+0		x	x	x	x	x	x	
Auto LF on	27+53+1		x	x	x	x	x	x	
Auto perf. skip n lines	27+78+n	x	x	x	x	x		x	x
Auto perf. skip off	27+79	x	x	x	x	x		x	x
Auto ribbon-band shift	27+97		x						
Backspace (BKSP)	8		x	x	x	x	x	x	
BKSP l+h*256/120th "	27+101+l+h		x			x			
Cancel	24	x	x	x	x		x	x	
Carriage return	13	x	x	x	x	x	x	x	x
Character set 1	27+55	x	x	x	x	x	x		x
Character set 2	27+54	x	x	x	x	x	x		x
Clear tabs (reset)	27+82		x	x		x	x	x	
Command end	0	x	x	x	x	x	x	x	x
Command start	27	x	x	x	x	x	x	x	x
Condensed print	15	x	x	x	x			x	x
DP quality print	27+73+1		x		x				
Deselect printer n	27+81+n		x	x	x		x		
Deselect printer	19		x	x	x		x		
Double strike off	27+72	x	x	3	x				x
Double strike on	27+71	x	x	3	x				x

Description of Print Function	ASCII code	Graphics Printer	Color Printer	Pro-Printer	Color Jet	Quiet Writer	Wheel Printer	Compact Printer	3812 Printer
Normal download font	27+73+4			x					
Normal resident font	27+73+0			x					
Normal width	18	x	x	x	x	x	x	x	x
Null	0	x	x	x	x	x	x	x	x
Overscore off	27+95+0			x					
Overscore on	27+95+1			x					
Print char. n	27+94+n		x	x		x	x		
Print next l+(h*256) chars.	27+92+l+h+...		x	x		x	x		
Proportional off	27+80+0		x			1	x		
Proportional on	27+80+1		x			1	x		
Read printer status	27+91+115+l+h+m+n						x		
Reverse line feed	27+93		x			x	x		
Ribbon band 1	27+121		x						
Ribbon band 2	27+109		x						
Ribbon band 3	27+99		x						
Ribbon band 4	27+98		x						
Select font holder *	27+73+1					x			
Select font holder A	27+73+0					x			
Select printer	17		x	x	x		x		
Set 6/72nd LPI	27+49		x						
Set 7/72nd LPI	27+49	x		x					x
Set 6 LPI	27+50	x	x	x	x	x	x		x
Set 8 LPI	27+48	x	x	x	x	x	x		x
Set9/96 LPI	27+49					x	x		
Set bckgnd color	27+91+78+1+0+c				x				
Set bit image mode	27+75+l+h+d				x				
Set control values = ASCII	27+64+1		x						
Set control values = binary	27+64+0		x						
Set exception char. n	27+91+72+n						x		
Set foreground color	27+91+77+1+0+c				x				
Set global char. set n	27+91+84+5+0+n						x		
Set horiz. motion index	27+62+n						x		
Set l/r margins	27+88+l+r		x			x	x		

Dbl. width off (by line)	20	x	x	x	x	x		x	x
Double width on (by line)	14	x	x	x	x	x		x	x
Double width off	27+87+0	x	x	x	x	x		x	x
Double width on	27+87+1	x	x	x	x	x		x	x
Download char. font	27+61+n...			x					
Download printwheel table	27+61+n...						x		
Emphasized off	27+70	x	x	x					x
Emphasized on	27+69	x	x	x					x
Form feed	12	x	x	x	x	x	x	x	x
Forward l+h*256/120th "	27+100+l+h		x			x			
Graphics 120 DPI	27+89+l+h+n...	x		x		2			x
Graphics 120 DPI (1/2 spd)	27+76+l+h+n...	x		x		2			x
Graphics, 140 or 168 DPI	27+89+l+h+n...		x						
Graphics 140 or 168 DPI (1/2 spd)	27+76+l+h+n...		x						
Graphics 240 DPI image (1/2 spd)	27+90+l+h+n...	x		x		2			x
Graphics 280 or 336 DPI (1/2 spd)	27+90+l+h+n...		x						
Graphics 60 DPI	27+75+l+h+n...	x		x		2		x	x
Graphics, 70 or 84 DPI	27+75+l+h+n...		x						
Half line feed fwd.	27+104						x		
Half line feed rev.	27+105						x		
Home print head	27+60	x	x					x	
Horizontal Tab	9	x	x	x	x	x	x	x	x
Ignore paper end off	27+57	x				x			
Ignore paper end on	27+56	x				x			
Initialize function off	27+63+0		x						
Initialize function on	27+63+1		x						
Line feed	10	x	x		x	x	x	x	x
Near letter quality (NLQ)	27+73+3		x						
NLQ download	27+73+6			x					
NLQ resident font	27+73+2			x					

1. Prints regular characters 1/2 line below writing line rather than 1/2 height.
2. Prints regular characters 1/2 line above writing line rather than 1/2 height.
3. Character spacing is a function of the font selected.

Set line length = 13.2 "	15					x			
Set line length = 8.0 "	18					x			
Set n vertical units	27+91+92+1+0+n					x			
Set n line perf. skip	27+78+n	x	x	x	x	x		x	x
Set n/144 LPI	27+51+n		x						
Set n/216 LPI	27+51+n	x		x					x
Set n/48 LPI	27+65+n						x		
Set n/72 LPI	27+65+n	x	x	x		x			x
Set page len. = n "	27+67+0+n	x	x	x	x	x	x		x
Set page len. = n lines	27+67+n	x	x	x	x	x	x	x	x
Set surface color c	27+91+76+2+0+c				x				
Set print mode n	7+91+86+n						x		
Set print resolution r	27+91+48+1+0+r				x				
Set initial values = n	27+91+75+1+0+n				x		x		
Set image resolution	27+91+79+l+h+f +hr+vr+d				x				
Set tabs (horiz.) = n...	27+68+n...+0	x	x	x	x	x	x	x	x
Set tabs (vertical) = n...	27+66+n...+0	x	x	x	x	x			
Set top of form	27+52		x	x		x	x		
Single n/144 LPI	27+74+n		x						
Single n/216 LPI	27+74+n	x		x					x
Start new line spacing	27+50	x	x	x		x	x	x	x
Stop printer	27+106					x			
Subscript on	27+83+1	x	x	x	x	4	4		x
Sub/superscript off	27+84	x	x	x	x	x	x		x
Superscript on	27+83+0	x	x	x	x	5	5		x
Swap foreground background off	27+91+93+1+0+0				x	x			
Swap foreground background on	27+91+93+1+0+1				x				
Text quality print	27+73+2		x		x				
Underline off	27+45+0	x	x	x	x	x	x	x	x
Underline off (color)	27+91+69+0+0				x				
Underline on	27+45+1	x	x	x	x	x	x	x	x
Underline on (color)	27+91+66+2+0+b+c				x				
Unidirect. print off	27+85+0	x	x	x			x		
Unidirect. print on	27+85+1	x	x	x			x		
Vertical skip	27+91+85+1+0+d				x				
Vertical Tab	11	x	x	x	x	x	x	x	x

4. Available on the Quietwriter Printer model 2 only.
5. Near letter quality print.

Note Number	Note Name	Frequency	Counter Value
0	A	55	21636
1	A# or Bb	58.27047	20422
2	B	61.73542	19276
3	C	65.40639	18194
4	C# or Db	69.29566	17173
5	D	73.4162	16209
6	D# or Eb	77.78174	15299
7	E	82.40688	14441
8	F	87.30705	13630
9	F# or Gb	92.49859	12865
10	G	97.99884	12143
11	G# or Ab	103.8262	11461
12	A	110	10818
13	A# or Bb	116.5409	10211
14	B	123.4708	9638
15	C	130.8128	9097
16	C# or Db	138.5913	8586
17	D	146.8324	8104
18	D# or Eb	155.5635	7650
19	E	164.8138	7220
20	F	174.6141	6815
21	F# or Gb	184.9972	6433
22	G	195.9977	6072
23	G# or Ab	207.6523	5731
24	A	220	5409
25	A# or Bb	233.0818	5106
26	B	246.9416	4819
27	C	261.6255	4548
28	C# or Db	277.1825	4293
29	D	293.6647	4052
30	D# or Eb	311.1269	3825
31	E	329.6275	3610
32	F	349.2282	3408
33	F# or Gb	369.9943	3216
34	G	391.9953	3036
35	G# or Ab	415.3045	2865
36	A	440	2705
37	A# or Bb	466.1636	2553
38	B	493.8832	2409
39	C	523.251	2274
40	C# or Db	554.365	2147
41	D	587.3293	2026
42	D# or Eb	622.2537	1912
43	E	659.2548	1805
44	F	698.4561	1704
45	F# or Gb	739.9886	1608
46	G	783.9905	1518
47	G# or Ab	830.609	1433
48	A	880	1352
49	A# or Bb	932.327	1276
50	B	987.7659	1205
51	C	1046.502	1137
52	C# or Db	1108.73	1073
53	D	1174.658	1013
54	D# or Eb	1244.507	956
55	E	1318.51	903
56	F	1396.912	852
57	F# or Gb	1479.977	804
58	G	1567.981	759
59	G# or Ab	1661.218	716
60	A	1760	676
61	A# or Bb	1864.654	638
62	B	1975.532	602
63	C	2093.003	569
64	C# or Db	2217.46	537
65	D	2349.317	507

Table 1. Table of Musical Notes

This concludes the prerequisite data. All that remains is to tell the Personal Computer what to do.

Of the various input and output options that the 8253 chip provides, you need to 1) select square-wave output and two-byte binary input, and 2) enable the third timer to accept data. When this is done, send timer three the value for the pitch you want to produce.

The IBM PCjr architecture includes an advanced sound-generating subsystem that produces up to three tones simultaneously.

The 8253 control port is found at I/O address 43H. The operand you send to this port contains information regarding the mode of operation. Timer three's input port is located at I/O address 42H. Here you put the actual two-byte counter value.

The hex value 0B6 should be sent to 43H, preparing the timer for the first note. Next, send the low byte of the pitch operand (the counter value) to 42H, immediately followed by the high byte of the pitch operand. Now everything is set, but where is the sound?

One final step remains: turning the speaker on. To do this, send the hex value 4F to the I/O port 61H. (Be careful with this value. If you err, you could inadvertently turn off keyboard input and have to power off to reboot).

Suddenly a square-wave tone comes from the Personal Computer. After a few seconds you will realize that the first note is still playing, and the computer isn't running out of breath. When will it stop? Whenever you send hex value 4D to I/O port 61H.

Putting It All Together

The counter values generated previously now can be used in an assembly language program that produces a musical melody.

Rhythms

Keep in mind that you will need a way to pause for variable periods of time to provide various durations of notes. You might also consider a method of implementing rests. (A rest is the absence of sound.) You can turn the speaker off and pause for a period of time, or you can send the counter a very small or very large value which makes a sound above or below the audible range. I recommend a small value (such as 1), because it avoids clicks that may be audible with a very large number. Of course, if you want special effects...

You might wish to experiment with playing triplets—where three notes are played in a time interval meant for two notes—or even more complex rhythms.

Unfortunately, chords and timbre changes are not easily implemented on the IBM Personal Computer. These require specialized hardware such as D/A converters. You may succeed somewhat if you have an extensive mathematical background and some experience in Fourier analysis. Otherwise, just be content with the melody, not the whole symphony.

Music on the PCjr

The IBM PC*jr* architecture includes an advanced sound-generating subsystem that produces up to three tones simultaneously. Using PC*jr*'s Cartridge BASIC, you can play more complex music than a PC, PC XT, Portable PC, or Personal Computer AT allows.

Before you begin making music, you will need a PC*jr* with Cartridge BASIC, and a cassette or diskette on which to store your latest compositions. You also need a monitor (which must have a speaker) or a television attached to the PC*jr*.

In PC*jr* BASIC, you must issue a SOUND ON command to prepare the sound subsystem for multiple-voice data. This also disables the speaker located inside the PC*jr* itself. Now you need only issue PLAY statements in a slightly altered format.

If you wish to activate all three voices at once, issue the following PLAY statement:

```
PLAY note1$, note2$, note3$
```

Each of the string variables in this statement contains music data that is nearly identical to data accepted by PLAY statements in standard IBM Personal Computer BASIC 2.00 and above. The major difference is that the PC*jr* PLAY statement will accept three different strings (separate by commas) as data to be sent *simultaneously* to the three tone generators.

Figure 2 shows a PC*jr* Cartridge BASIC program that plays music data from file BACHINC.MUS. This music is an excerpt from J. S. Bach's *Little Prelude and Fugue in C Major*. I have put measure numbers in the data file to identify sections of the music easily. Each line of data within a measure is played concurrently with the others.

```
 10  OPEN "BACHINC.MUS"
       FOR INPUT AS #1
 20  SOUND ON
 30  PLAY "T100", "T100", "T100"
 40  WHILE NOT EOF(1)
 50  INPUT #1,X$
     'Read measure number
      from file
 60  INPUT #1,A$
      'Music data read into
       A$,B$,C$
 70  INPUT #1,B$
 80  INPUT #1,C$
 90  PLAY A$,B$,C$
100  WEND
110  CLOSE #1
120  END
```

Figure 2. PCjr BASIC: Excerpts from Bach

Figure 3 shows the BACHINC.MUS file. In this file, the commented lines that indicate the measure number are read in as X$, then discarded. The next three lines after the comment are A$, B$ and C$ respectively.

Editor's note: We have placed an assembly language program on the IBM Electronic Bulletin Board System into the file FUGUE.ASM and the fully assembled program into the file FUGUE.EXE. We have also placed the listing shown in Figure 3 into the file BACHINC.MUS. All three of these files are available for downloading from the <F>iles section of the bulletin board which you can access by calling (305) 998-EBBS.

```
* measure 1
o1mlc1
o2p2p16l16c<b>c<g>c<b>c
o3l16p16c<b>c<g>c<b>c<e4p4
* measure 2
o1c4mnp8e8f4p4
o1e4p8g8a4p4
o3p16c<b>c<g>c<b>c<afefcfef
* measure 3
o1f+4p8f+8g4p4
o1a4p8a8b4p4
o2l16d>dcd<a>dcd<bgf+gdgf+g
* measure 4
o1g+4p8g+8a4p4
o1b4p8b8>c4p4
o2l16e>ede<b>edec<ag+aeag+a
* measure 5
o1l8ddddgggg
o1l8aaaabbbb
o3l16c<f+ef+>c<f+ef+bgf+gbgf+g
* measure 6
o1ccccf+f+f+fo1ggggaaaa
o2l16bedebedeaf+ef+af+ef* measure 7
o0bbbb>eeee
o1f+f+f+f+gggg
o2l16adcdadcdgedegede
* measure 8
o0aaaa>dddd
o1eeeef+f+f+fo2gc+<b>c+gc+<b>c
  f+dc+df+dc+d
* measure 9
o1mld1
o1b4f+4g4f+4
o2l16gf+gdagadbabd>c<b>c<d
* measure 10
o1d4p8o2g8g2
o1l16g>def+g8l8<b>c<bag
o3l16d<f+gab8>d8egdgcg<b>g
* measure 11
o2g8mnf+16e16f+8o1l8gf+gd4
o2d4p8d8d8d8d8<f+8
o2a4p8b8>cd<b>c<b16a32b32a48
  b48a48g16
* measure 12
o1g4p2.
o1l16bgf+gdgf+gg4
o2g4p2.
```

Figure 3. A Listing of the BACHINC.MUS File

AT Utilities

Richard Serbin
Lawrence Livermore National Laboratory

Turbo Keyboard for the IBM Personal Computer AT

If you have an IBM Personal Computer AT and would like to speed up the keyboard response, this program is for you. It can increase response time by 300 percent, making word processing or text editing cursor movement fly on the screen.

The program takes advantage of the way IBM designed the new, more flexible, AT keyboard interface. Part of the flexibility is that the user can speed up or slow down the keyboard.

When you start the AT, the default keyboard configuration is set to .5 seconds of delay and a typematic rate of 10 characters per second.

The delay rate determines how long a key must be held down before it starts to repeat. The typematic rate sets the repeat frequency once the delay requirements have been met. On the AT, the delay is capable of a range from .25 to 1.25 seconds, and the typematic rate can range from 2 to 30 characters per second.

To use the turbo keyboard program, all you have to do is include the program execution statement in your AUTOEXEC.BAT file. Or, at any time, just execute the turbokey program with the desired rates you want to use.

The parameters for changing the typematic rate range from A to Z, where A is the fastest at 30 characters per second and Z is the slowest at 2 characters per second. The parameters for altering the delay rate range from 1 to 4, where 1 is the fastest at .25 seconds delay and 4 is the slowest at 1.25 seconds delay.

For example, if you want the fastest combination possible, type the following at the DOS prompt:

```
TURBOKEY A 1
```

This will set the keyboard to 30 characters per second and .25 seconds delay.

If you like a very slow response, then do the following:

```
TURBOKEY Z 4
```

This will set the keyboard rate to 2 characters per second, and a start delay of 1.25 seconds.

By adjusting the A thru Z and the 1 thru 4 parameters, you can set the keyboard response to whatever speed you want at any time.

Resetting the AT's Clock and Date the Simple Way

While working with the DOS 3.00 manual, I noticed that the TIME and DATE commands had no effect on the Personal Computer AT's C-MOS real-time clock. The manual states: "If you are using a IBM Personal Computer AT, typing a new time [or date] does not change the system clock." In its *Guide to Operations* manual, IBM suggests you change the time and date by running the SETUP program on the diagnostic diskette and then rebooting the system. I have written an assembly language program that allows easy time and date changes on the system clock.

The program consists of a BATch file SETCLOCK.BAT and a program file, ATCLOCK.COM. To use the program, type:

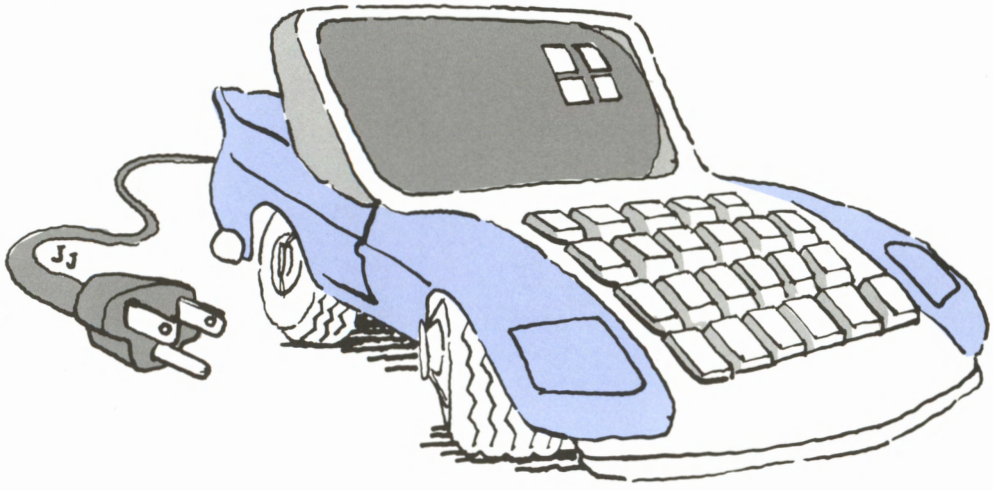
```
setclock  hh:mm:ss
          mm-dd-yy
```

where

hh:mm is the hours, minutes, seconds

mm-dd-yy is the month, day, and year.

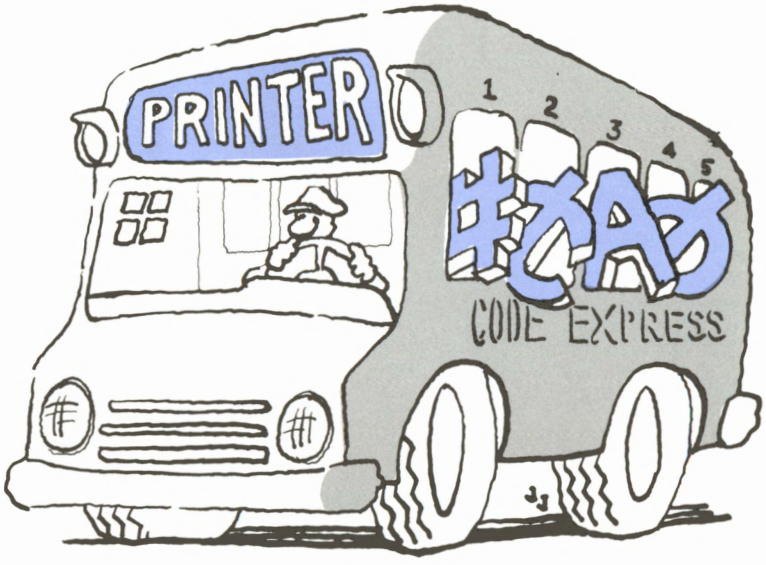
Editor's note: Mr. Serbin's program files, TURBOKEY.COM and ATCLOCK.COM, as well as his BATch file SETCLOCK.BAT, are available for downloading from the <F>iles section of the IBM Electronic Bulletin Board System, (305) 998-EBBS.



All the Codes Fit to Print

John Warnock
IBM Corporation

IBM makes many different printers for the Personal Computer family. Each of these printers has different features and abilities such as quality print or graphics. To use some of these features, you need to send the printer special control codes to activate the features. This article discusses some of the ways to send those codes. Complementing this article is a pull-out chart in the center of this issue that lists the features available on some of the IBM Personal Computer printers, and the codes that activate the features.



Printer Control Codes

Many printer control codes contain the same characters you type every day. What makes these characters recognizable codes is that they include some special characters in a specific sequence. The trick is getting these special characters to the printer. All the characters in the IBM Personal Computer have numbers, called ASCII codes, assigned to them. There are 255 characters available, not counting the Null character, which is zero. For example, the letter "A" is ASCII code 65.

Normally, you wouldn't care what ASCII code makes up the letter "A"; you just press the key on the keyboard and the letter shows up on the screen. You may think that there is no way 255 characters will fit on your keyboard. Ah, but there is! In the bottom row of keys on your keyboard is the Alt key. The Alt key works like a special shift key. Hold it down, and you can enter the special ASCII character codes from the numeric keypad (the Num Lock key can be on or off).

For example, hold down the Alt key and type 65 on the numeric keypad. Release the Alt key, and the letter "A" appears on your screen. The letter "A", by itself, does nothing to control your printer.

But place the ASCII code 27 (ESCape) in front of it, and a few more ASCII values behind it, and you can reset the line spacing on your printer.

Getting There from DOS

Many programs will send the right codes to the printer for you. There are times, however, when the programs don't send the codes you want to your printer, or when you want to talk to the printer from DOS. The DOS MODE command lets you set the line width and line spacing on the printer, but not some of the other features. A printer feature does you no good if you can't tell the printer to use the feature.

One way of sending control codes to the printer is to copy them from the console under DOS. Just as you can copy information from one file to another, you can copy information from one device to another. In this instance, you are copying from the console (CON:) to the printer (LPT1:). The following example will cause your printer to eject a page. Type:

```
COPY CON:  LPT1:
```

and press Enter. Then, hold down the Alt key and type 12 on the numeric keypad. Press the F6 key and the Enter key to copy the code to the printer. The ASCII 12 character is the page eject code, and the F6 key produces an end-of-file character that tells DOS there is nothing more to copy.

This method works for simple print codes, but does not support codes that DOS itself uses. The ESCape character (Alt+27) is one example. Many printer commands start with that character, but DOS uses that same character to erase the current line and start a new one. Many programs you use also trap that character. Therefore, you must use some other method to send the ESCape control code from DOS.

Getting Through DOS

The first 32 ASCII characters aren't used often. DOS uses many of those characters itself and will trap them before they can get to the printer. The trick is to get these characters through DOS. Luckily, most IBM printers support two character sets. In the first character set, ASCII values 128 to 159 duplicate the functions of the first 32 characters. (In the second character set, they represent foreign characters.) While the ESCape character, ASCII code 27, causes DOS to start a new command line, ASCII value 155 (128 added to 27) produces the cent symbol on the screen, but the printer reads it as an ESCape. To set your printer for double-width from DOS, type:

```
COPY CON: LPT1:
```

and press Enter. Then hold down the Alt key and type 155 (the ESCape code) on the numeric keypad. Type a capital W, then hold down the Alt key and type 129 (128 added to 1) on the numeric keypad. The 129 turns the feature on. Typing 128 (128 added to 0) turns the feature off. Press the F6 key and press Enter.

Using Files

While this method works, it can be rather tedious if you use the same codes over and over again. It isn't easy to remember some of these codes either. So far, you have copied the codes from the console to the printer. Another way to send codes to the printer is to store the codes in ASCII files and print or copy those files to the printer when you want them.

You can use text editors like Edlin or Personal Editor to create these files. Some editors have problems handling special characters like the ESCape character, so be careful which editor you use. Or use the character value plus 128. You should name the files so they describe the codes they contain, like EMPHASIZ, CONDENSE, REGULAR and so on.

If you don't have a text editor, or feel that using one is cumbersome, you can copy the codes to a file as easily as you copied them to the printer.

```
COPY CON: CONDENSE
```

Hold down the Alt key and type 15 on the numeric keypad. Press the F6 key and the Enter key to finish creating the file. You can then send the CONDENSE file to the printer by typing:

Sending Escape Codes to Your Printer

Milt Hull
Sacramento PC Users Group

Here's a quick way to send an ESCape code to your printer without using a program or a utility and without wasting paper.

First, change the DOS prompt to the ESCape character (ASCII 27) by entering

```
prompt=$e
```

Second, send the ESCape character to the printer by pressing Ctrl and PrtSc to put the printer in echo mode, and then press the Enter key once to send the prompt (which is now the ESCape character) to the printer.

Third, complete your ESCape code by sending the appropriate character(s) to the printer. For example, type an upper case E for emphasized mode or upper case W1 for double width. However, do not press the Enter key after you type the character, because the character you type goes directly to your printer as code, and pressing the Enter key will only cause the code (E, W1, etc.) to print on the paper, wasting the sheet.

Fourth, press Ctrl and PrtSc once again to turn off the printer echo.

Finally, press the backspace to delete the character (Ctrl + PrtSc) you sent to the printer, and enter the PROMPT command, or a BATch file containing the PROMPT command (and any parameters), to return the prompt to normal.

For a list of ESCape codes for IBM printers, see the pull-out chart in the center of this issue.

```
COPY CONDENSE LPT1:
```

If you have DOS 2.00 or later, you can use I/O redirection and type:

```
TYPE CONDENSE >PRN
```

You can build more complex printer commands this way, and combine many different codes, too. This method is also useful for creating simple BATch files.

Within a Program

Programs can send these special codes to the printer as well. Since their output isn't filtered through DOS, they simply need to print or send the right values to the printer. For example, to tell a printer to use emphasized print with double-strike, a BASIC program might have the line:

```
20 LPRINT CHR$(27)+CHR$(69)
  +CHR$(27)+CHR$(71);
```

You can also substitute characters for the ASCII values, too. While the chart shows only the ASCII values, the same BASIC example might look like this:

```
20 LPRINT
  CHR$(27)+"E"+CHR$(27)+"G";
```

Editors and Word Processors

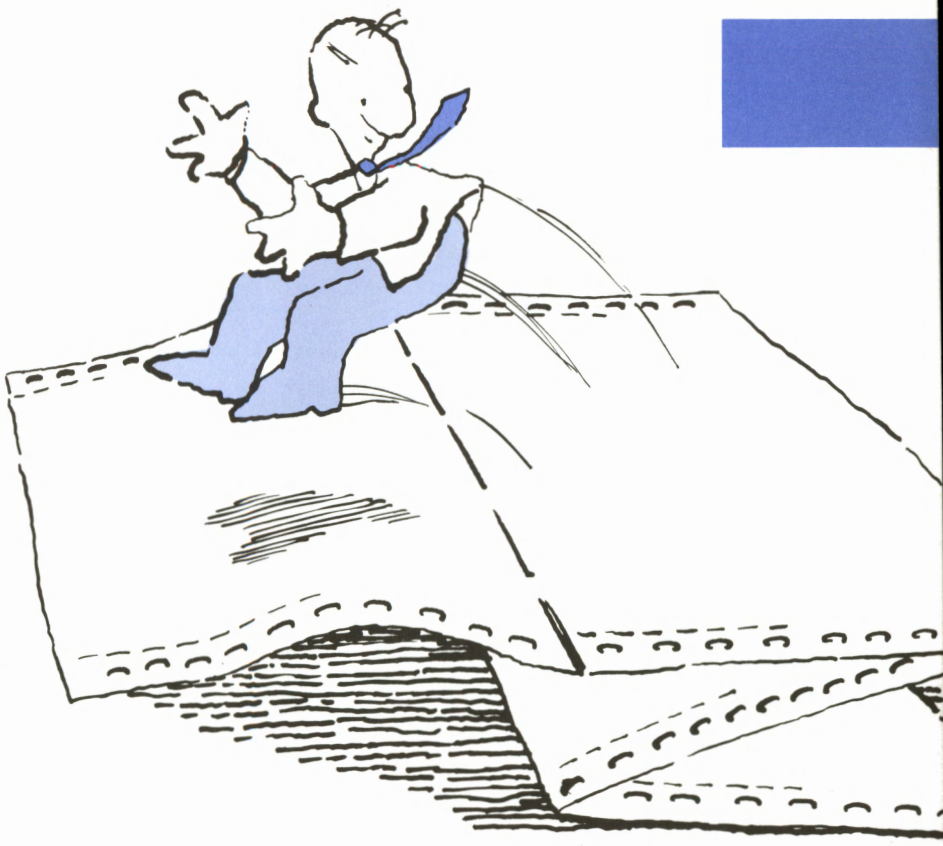
Some editors and word processors lend themselves to print codes more easily than others. The IBM Personal Editor lets you imbed special print codes within the text. You can even assign the print code to one of the keys on the keyboard. This makes it easier to remember and use the code.

Some word processors such as PCWrite store predefined printer control codes in a file. When they want to use emphasized printing they read the code from the file and send it to the printer. Changing these values can be more difficult—you need to know which file holds the codes, and where each code starts in the file.

The chart in the center of this issue is a compilation of all the printer codes for the IBM Personal Computer. The chart may be removed for future reference. For consistency, all values in the chart are expressed as ASCII values rather than character expressions.

Skipping Perforations

Bill Weil
San Francisco PC Users Group



One annoyance with printing ASCII files that are more than one page long is that they print right over fanfold paper perforations. You can eliminate this inconvenience by telling your printer to skip over the perforations.

First, look up the codes in your printer manual that turn "skip over perforations" (SOP) on and off. For IBM printers:

```
<Esc>N#
```

turns SOP on for a specified number (#) of lines before the perforations, and:

```
<Esc>O
```

turns SOP off.

If you don't have a text editor that lets you insert special characters, such as ESCape (ASCII 27), directly into a text file, you can create a DOS file that contains the necessary codes. DOS lets you enter special codes into text files by using the Alt key in combination with the numeric keypad to enter ASCII codes.

The ASCII code for the ESCape character is 27. Unfortunately, you cannot enter ASCII 27 by holding down the Alt key while you type 27 on the keypad, because when ESCape is used in DOS, the computer clears the command line. However, to all IBM printers that use character set 1, an ASCII 27 and an ASCII 155 (27+128) mean the same thing. The binary equivalents of ASCII 27 and 155 are identical except for the eighth bit. Since most printers look only at the first seven bits, the binary code for ASCII 27 and 155 appear identical to the printer. (If you have trouble sending any other lower 128 ASCII character to your printer, try adding 128 to it.)

To avoid having to enter these codes directly in each text file, you can create two files—one containing the codes to turn SOP on and one that has the codes to turn it off. You then create a BATch file that will send the two code files and your text file to the printer.

To build a file that tells the printer to skip six lines at each page perforation, type the following at the DOS prompt:

```
COPY CON: SOP.PRT  <Enter>
<Alt+155>N<Alt+6>  <Enter>
<F6>               <Enter>
```

To type the Alt+155 above, hold down the Alt key and type 155 on the numeric keypad (not the numbers at the top of the keyboard), then lift the Alt key, type N, and finally hold down the Alt key again and type 6.

To build a file to tell the printer to cancel SOP, type the following at the DOS prompt:

```
COPY CON: NOSOP.PRT <Enter>
<Alt+155>O          <Enter>
<F6>                <Enter>
```

The final step is to build a BATch file to send these codes to the printer.

Type the following at the DOS prompt:

```
COPY CON: COPY2PR.BAT                <Enter>
COPY SOP.PRT + %1 + NOSOP.PRT PRN    <Enter>
<F6>                                 <Enter>
```

To print any text file, type:

```
COPY2PR filename <Enter>
```

where

"filename" is the name of the file you want to print.

Note: If you would like the six lines split evenly above and below the perforation, set the paper in the printer three lines below the actual top of the page. If you want to print more than one text file at a time, add a series of BATch variables to the COPY2PR.BAT file. For example, type:

```
COPY SOP.PRT + %1 + %2 + %3 + %4
  + NOSOP.PRT PRN  <Enter>
```

Then, at the DOS prompt, enter:

```
COPY2PR filename1 filename2
  filename3 filename4 <Enter>
```

When you print files this way, they print continuously without a page eject between the files. To instruct the printer that you would like a page eject between the files, you can either add the page eject (ASCII 12) at the end of each file, or you can create a file that contains the page eject code:

```
COPY CON: PAGEOUT  <Enter>
<Alt+12>           <Enter>
<F6>               <Enter>
```

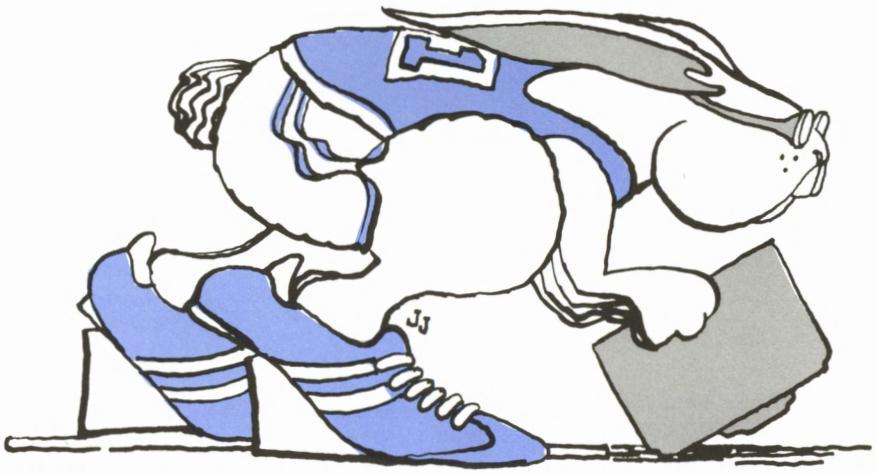
Once you have this file, you can enter it as part of the command string you enter at the DOS prompt:

```
COPY2PR filename1
  pageout filename2 pageout <Enter>
```

You can also use this technique to build other BATch files that turn various printer features on and off.

The Virtual Disk Drive: Fleet and Silent

Chuck Gaston
Poughkeepsie IBM Club Microcomputer Club



If you do not use a virtual disk drive, you are not taking advantage of one of the most valuable utility programs available for your computer. If you do use one already, this article may show you some new applications or stimulate a few ideas of your own.

What Is It?

What *is* a virtual disk drive? First, consider that a real disk drive provides a place to store information that will not fit within a program's work area (64KB is BASIC's absolute limit) or that must exist after the program disappears from memory (a game's high scores, or the program itself).

A virtual disk drive serves the same general purpose, with four distinct differences:

1. It is totally silent
2. It is tremendously fast
3. It may be any size—from 1KB to the limits of your random access memory (RAM) above 64KB (the system reserves 64KB for active memory)
4. If your computer loses power or locks up, everything on the virtual disk is lost!

The concept has been called a ramdisk, disk emulator, electronic disk or disk simulator, but whatever the name, it means the same thing: a portion of memory is set aside to serve as fast-access storage that behaves just as if it were a physical disk drive. Files can be saved, loaded, copied, erased and generally treated just like files on a diskette or fixed disk.

What Good Is It?

Since a virtual disk will be completely erased anytime the power fails, it is *not* the place to save the latest version of that new program you're developing or the latest ten pages of your Great American Novel. So what good is it? In general, it's best used for smaller files that are accessed often, easily replaced, temporary, or needed quickly to avoid sluggish performance.

I constantly use a virtual disk with IBM Personal Editor (PE). My standard AUTOEXEC.BAT file copies PE, the PE profile PE.PRO, and several macro files to my virtual disk. If those macros were on a regular floppy disk, there would be a noticeable noise and delay each time they were used. With the virtual disk as the default drive, access is silent and nearly instantaneous.

Other key definitions depend on the speed of the virtual disk when they create, save and execute small macros on-the-spot, or save and recall files in order to strip trailing blanks or get the current time.

One of my neighbors puts BASICA.COM, a number of BASIC utilities, and many different BATch files on his virtual disk. Within a BATch file he is able to call up BASICA, execute a small program and return to DOS (using SYSTEM at the end of his BASIC program) with no disk grinding and very little elapsed time. This approach gives him the effect of extending DOS with any routine he can program in BASIC.

If you have ever used the SORT function in DOS 2.00 and above, you'll know that the associated disk grinding can make you fear for the safety of your machine. Virtual disk to the rescue! Having a virtual disk as the default drive when doing a SORT transfers all that thrashing into memory—where it belongs.

Did you ever try to copy a large number of small files from one disk to another on a single-drive machine? A standard "COPY A:*.* B:" would require twice as many disk swaps as you have files! With a virtual disk you can copy many files from diskette to virtual disk, then copy all of them back to the other (or to more than one diskette) without doing two disk swaps for each file copied.

Do you need to make many copies of a diskette? If you have enough memory to hold all the diskette's data in a virtual disk, you can set up a BATch file to copy alternately to the A and B drives about as fast as you can swap disks.

In a similar vein, I created a BATch file to speed up formatting a box of diskettes. Running the BATch file on the virtual disk allows me to use both drives alternately. While one drive is formatting, I change diskettes in the other and affix blank labels. The job goes twice as fast.

If you use many files spread over several diskettes (such as consolidating files and discarding duplicates), you might like to have all directories available simultaneously. No problem with a virtual disk. A separate file can be created on the VDISK for each floppy:

```
C>dir a: | sort > articles
```

With the virtual disk as the default drive, access is silent and nearly instantaneous.

In this example, I assumed that the VDISK is drive C (the default drive) and that SORT.EXE already has been copied to it. The directory of the diskette in drive A is sorted alphabetically and saved as C:ARTICLES (the name is your choice, of course). Once each directory of interest has been saved in this way, they can be TYPEd or used as input to your favorite word processor, without having to remember or print everything, or do any further disk shuffling.

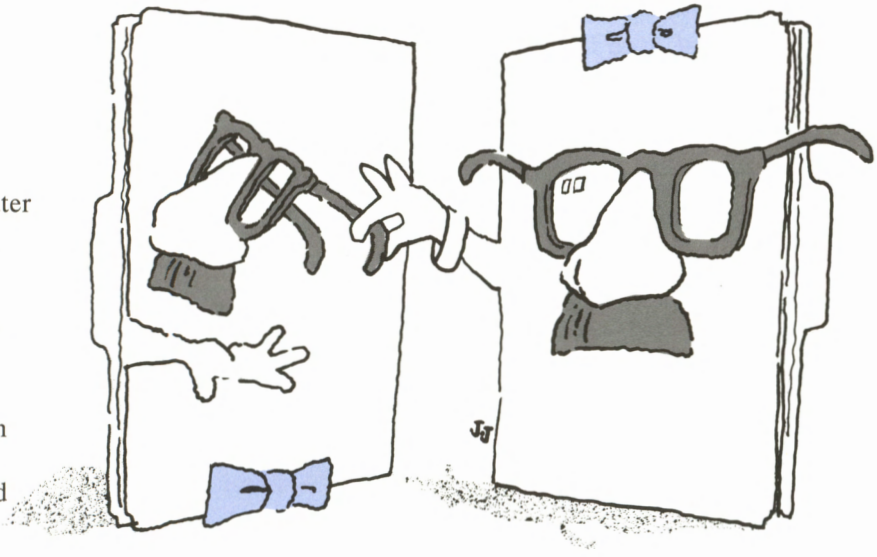
Almost any program that requires frequent disk access for help screens, display images, code overlays and the like can benefit from using a virtual disk. TopView is one of these.

Where Can You Get One?

You may already have one! If you have DOS 3.00 or 3.10, it includes a file named VDISK.SYS, and the reference manual has simple instructions on how to install it. Even DOS 2.00 includes the complete assembly language source code for a virtual disk, if you are up to that sort of thing. If you don't have one of these already, I recommend getting DOS 3.00 or 3.10. They have a number of goodies, and VDISK is only one.

Tips and Techniques

Karen Porterfield
IBM Corporation



Across the country, different IBM Personal Computer users use different configurations of hardware and software. During the course of their usage, small businesses, corporate employees, teachers, housewives, and students all discover special ways to use applications that are helpful or save time.

Some of the tips and techniques listed here are from IBM employees; others come from user group members. Let us know if these tips helped you, and send us your own gems that you've polished and hidden away!

Writing Assistant, Filing Assistant and Personal Editor

If you create a file using Writing Assistant and want to give it to Mary—who uses IBM Personal Editor—to edit it, there will be a problem because the files are not compatible (although you can create a file in Personal Editor and edit it using Writing Assistant). What can you do? Instead of choosing option 4 (Get/Save/Remove) from the Writing Assistant menu, choose option 3 (Print). Then, where the Print Menu shows

```
Print to: PRINTER
```

type the drive and file name over the word PRINTER. For example, you type

```
Print to: a:filename
```

Now, give the diskette to Mary. She will be able to edit the file using Personal Editor.

The same rule applies to Filing Assistant. Suppose you create a mailing list using Filing Assistant and want to give a copy of it to John so he can alter it for his own use. To create a file that he can load with his text editor, choose option 5 (Print) on the IBM Filing Assistant Main Menu. Go through the Print Menus just as you would to print the file on the printer, but when the Print Options Menu appears, type the drive and file name where PRINTER appears. For example:

```
Print to: a:filename
```

Finish filling in all other appropriate fields. When IBM Filing Assistant prints the file, it will print to the A disk. You can give the diskette to John, and he can load the file with his text editor.

Assembly Language Program and BASIC Compiler

When you return to BASIC from an assembly language subroutine, the stack must be restored to its original state. To do this, you type:

```
RET n
```

where n is two times the number of parameters specified in the BASIC Call statement. Here is a convenient way to determine the value of n. First include the following data structure in your assembly language subroutine:

```
PARAMLIST   STRUC
SAVE_BP     DW      ?
RET_OFF     DW      ?
RET_SEG     DW      ?
PARAM4      DW      ?
PARAM3      DW      ?
PARAM2      DW      ?
PARAM1      DW      ?
PARAMLIST   ENDS

Paramsize EQU
  offset Param1 - offset RET_SEG
```

The RET statement would then be specified as:

```
RET PARAMSIZE
```

Note: Be sure that the PROC setting of your subroutine is set to PROC FAR.

DOS Abort, Retry, Ignore Messages

If you get this message from DOS when attempting to copy files from one drive to another, *do not* switch disks in the target drive and hit R for a Retry. You will end up with a hopelessly scrambled mess on the disk.

Such a situation can occur if you tried to copy onto a disk with the write-protect notch covered. The temptation would be to just insert a different disk and Retry. However, when DOS prepares to copy files, it first reads the directory from the target disk. If you switch disks, it will write the in-memory copy of the directory onto the new disk that you inserted.

There are several other ways that a *wrong* response to Abort, Retry, Ignore will produce scrambled contents on a disk. The safest method (if copying files is involved) is to Abort all such operations, correct the problem, and then key in the necessary commands again. Retry is safe for simple operations like DIR that do not involve moving data from one disk to another.

—Bill Weil, San Francisco PC Users Group

Selective Screen Blanking

Wayne Taylor
IBM Corporation

Working on a computer with both an IBM Monochrome and IBM Color Display offers a tremendous amount of freedom. You enjoy the speed and resolution of the Monochrome display, while still being able to call up color capabilities with a single command. But using two monitors simultaneously can be distracting, especially when the cursor on the unused monitor blinks in your peripheral vision all day long.

To remedy the problem, I wrote two similar BASIC programs that, when used in conjunction with the DOS MODE command, "blank" the monitor not in use. Instead of just typing "MODE mono" to switch from the color monitor to the monochrome, type the MODE command and then execute the COLZAP.BAS program listed below to blank the color display.

```
10 DEF SEG=0
20 POKE &H410,
   ((PEEK(&H410) AND &HCF) OR &H20)
30 SCREEN 0:LOCATE ,,0
40 SYSTEM
```

To return to the color display, type "MODE co80" and then run the BWZAP.BAS program listed below to blank the monochrome display.

```
10 DEF SEG = 0
20 POKE &H410, (PEEK(&H410)OR &H30)
30 SCREEN 0 :LOCATE ,,0
40 SYSTEM
```

Because you never know where in memory the BASIC interpreter will be loaded, you must define the current segment of memory to zero (DEF SEG = 0) before you can POKE and PEEK around in low memory.

The memory location at hexadecimal 410 (&H410), or decimal 1040, is the equipment flag byte, and you POKE the values that result from the ANDs and ORs into the the bits that activate the monitors. (The Boolean operators are used to isolate the specific bits within the equipment flag byte that descibe the current monitor being used, while leaving the rest of the byte unchanged.)

The SCREEN 0 command puts the monitor into text mode, and the LOCATE command turns the cursor off. Then, the SYSTEM command returns control to DOS, which should restore (with the help of ANSI.SYS) whatever display configuration you have specified as default.

To speed things up a bit, you can switch monitors by calling both the MODE command and the BASIC program from a BATch file. I named my BATch files COL.BAT and BW.BAT. When I want to use my color monitor, I execute COL.BAT which contains the following commands:

```
ECHO OFF
MODE co80
BASICA bwzap
```

Likewise, to return to my monochome monitor, I execute BW.BAT:

```
ECHO OFF
MODE mono
BASICA colzap
```

To make the transition between monitors as fast as possible, redefine two keys with ANSI.SYS to replace the BATch files. (For more about using ANSI.SYS see Steve Mahlum's article "ANSI.SYS ESCape Codes" in the September issue of *Exchange*.) My monochrome monitor sits to my left and the color sits to my right, so I have defined the F9 key to switch to my monochrome display and the F10 key to switch to my color display. To redefine these keys, make sure you have ANSI.SYS installed as a device in your CONFIG.SYS file and that you include the following two lines in your AUTOEXEC.BAT file:

```
ECHO <Esc>[[0;67;"MODE mono";13;"BASICA
      colzap";13p
ECHO <Esc>[[0;68;"MODE co80";13;"BASICA
      bwzap";13p
```

where <ESC> is ASCII Code 27

There are times, however, when you will not want to blank the screen when you switch monitors. In these instances simply use the MODE command. But most of the time I use only one monitor, and I enjoy having the unused monitor blank.

New Products

Hardware

Personal Computer AT Model 239

The IBM Personal Computer AT grows with the model 239. This new model maintains compatibility with the previous members of the AT family, but includes a 30MB fixed disk drive. This new model is ideal for the high capacity, high speed storage required by data processing, multitasking processing (TopView and Xenix) and IBM PC Network server environments.

The Personal Computer AT model 239 includes eight expansion slots, 512KB of memory on the system board, an Intel 80286 processor, BASIC language interpreter, a clock/calendar with battery backup, a single 1.2MB High Capacity Diskette Drive, one 30MB fixed disk drive, a Fixed Disk and Diskette Drive Adapter and a Serial/Parallel Adapter. A second diskette drive or fixed disk drive is optional. The AT model 239 requires a display and display adapter and DOS 3.00 or higher.

Each system unit comes with a *BASIC Reference* manual and *IBM Personal Computer AT Guide to Operations*. The *Technical Reference, Technical Reference Options and Adapters* and *Hardware Maintenance and Service* manuals may be purchased separately.

30MB Fixed Disk Drive Option

A 30MB Fixed Disk Drive Option is available for all members of the Personal Computer AT family. Two 30MB drives may be installed in the model 068, 099 or 239 to bring the fixed storage capacity to 60MB. No more than two fixed disk drives can be installed in any system unit; if two diskette drives are installed, each model is limited to one fixed disk. Owners of the Personal Computer AT models 068 and 099 should purchase the Upgrade Kit listed below to obtain the new BIOS needed to run the 30MB fixed disk.

30MB Fixed Disk Drive Upgrade Kit

The 30MB Fixed Disk Drive Upgrade kit comes with both the 30MB fixed disk and a new ROM BIOS module to support the disk. This kit allows owners of the Personal Computer AT models 068 and 099 to upgrade their systems to take advantage of the new 30MB fixed disk drives.

Personal Telephone Manager Adapter

The IBM Personal Telephone Manager Adapter is a full-sized feature card that integrates the telephone with the IBM Personal Computer. When used with the IBM Personal Telephone Manager Program (see Software section), this product can improve telephone communications and office productivity.

The adapter:

- Generates both rotary and touch-tone dialing signals
- Contains a call progress processor and programmable interval timers
- Detects telephone handset on or off cradle
- Has one attachment for a single phone line and another attachment for a phone handset (not provided)
- Allows the phone to be used when the PC is powered off
- Includes a built-in speaker for call monitoring and incoming ringing

The phone line complies with FCC Part 68 Rules, and the adapter meets FCC Part 15 Rules. You may purchase the *Personal Telephone Manager Adapter Technical Reference* manual if you want to develop your own telephone applications.

The Personal Telephone Manager Adapter requires a full-length expansion slot in an IBM Personal Computer, IBM Portable Personal Computer, IBM Personal Computer XT, or IBM Personal Computer AT; an analog telephone (pulse or tone) and cable with modular plug; an analog telephone USOC RJ11 C/W wall connector; and DOS 2.00 or later.

Voice Communications Adapter

The IBM Voice Communications Adapter (VCA) is a full-sized adapter card that adds specialized processing ability to the IBM Personal Computer. The adapter and IBM Voice Communications Operating Subsystem form the IBM Voice Communications Option and provide the capability to emulate a modem, record and playback audio (voice), recognize voice commands, convert text to speech, and issue telephony signals over phone lines. Information can come from IBM Personal Computer memory, an attached microphone or telephone, or two telephone lines (asynchronous). The output can be to memory, telephone, telephone lines (asynchronous), or an external speaker.

As a modem, the VCA conforms to Bell 103A and 212A standards, supporting speeds of 110, 300, or 1200 bits per second in full duplex mode. The VCA accepts voice input through telephone or microphone for commands or digitization, compression,

storage, and later playback through a telephone, telephone line or speaker. The VCA can convert ASCII datastreams to synthetic English speech to read text and data over phone lines, a telephone or speaker. As a telephony device, the VCA can generate touch-tone or pulse dialing signals to initiate calls; detect touch-tone pulses for remote phone input; and detect call progress signals for busy, ringing, voice, silence, etc. In conjunction with the Voice Communications Option, IBM announced several applications to use the features (see the software section, including the Voice Communications Operating Subsystem).

The Voice Communication Adapter can be installed in an IBM Personal Computer, IBM Personal Computer XT, IBM Personal Computer AT, or IBM Portable Personal Computer, and requires DOS 2.10 or later, along with the Voice Communications Operating Subsystem. Memory requirements vary by DOS version, application requirements, and serial or concurrent use of functions, and can range from 22KB to 256KB in addition to the DOS requirements. The Exploring Voice communications demonstration program requires 320KB.

Any applications written for the option must be written to the Voice Communications Applications Program Interface or the Voice Activated Keyboard Utility (see the software section), and accessed from an assembly language routine. A *Voice Communications Application Program Interface Reference* manual and Tool Kit diskette are available separately for experienced assembly language programmers.

The Voice Communication Adapter also requires one or two connected telephone lines for asynchronous communications, telephone management, and voice conversations; a connected telephone or microphone for voice recording, or voice recognition; and a connected speaker, telephone, or telephone line for audio output from record/playback or text-to-speech.

Software

IBM Personal Computer C Compiler

IBM Personal Computer C Compiler version 1.00 is a powerful, general-purpose, high-level programming language. The compiler uses C Language ASCII files to generate highly optimized machine language for fast execution in a small storage space. Suited for a wide range of programming application areas, the compiler includes:

- Support for the IBM Personal Computer Network
- Support for linked programs larger than 64KB
- Support for individual data objects larger than 64KB
- User-specified compile time options
- Selectable code optimization (size, speed, etc.)
- Selectable model size (small, medium, large)
- Symbolic debug
- Multiple warning levels for syntax, data conversion, type mismatch, and non-standard constructions
- User-determined floating point support options
- Runtime library support
- Interactive symbolic debug
- Implemented register support
- Program maintenance utilities
- Optional assembly language instruction generation

If used without the IBM PC Network, the C Compiler requires an IBM Personal Computer, IBM Personal Computer XT or IBM Portable Personal Computer with DOS 2.10 or higher; or an IBM Personal Computer AT with DOS 3.00 or higher. It requires 256KB of memory (384KB or higher recommended). At least 320KB is needed for symbolic debug or MAKE library options.

When used with the IBM Personal Computer Network, the C Compiler uses an IBM Personal Computer, IBM Personal Computer XT, IBM Portable Personal Computer or IBM Personal Computer AT with DOS 3.10 and at least 512KB of memory (640KB is recommended to use all the features of the C Compiler and IBM PC Network).

In addition, the IBM PC C Compiler requires an IBM Color Display, IBM Monochrome Display, IBM Enhanced Color Display or IBM Professional Graphics Display with the appropriate adapter; and two double-sided diskette drives (either 360KB or 1.2MB) or one diskette and one fixed disk drive. The C Compiler can also use the IBM Proprinter, IBM Compact Printer, IBM Graphics Printer, IBM Color Printer or IBM Quietwriter Printer.

DisplayWrite 3 Version 1.10

DisplayWrite 3 version 1.10 is an advanced text processing program for the IBM Personal Computer family. It lets you create, revise, footnote, spell check, paginate, and print documents. In addition to the functions of version 1.00, DisplayWrite 3 version 1.10 adds:

- Automatic conversion of DisplayWrite 3 text to and from revisable form text (RFTDCA) and to final form text (FFTDCA)
- Support for the IBM Wheelprinter, Color Jetprinter and IBM Quietwriter Printers
- International Keyboard Character Set to support languages other than English

DisplayWrite 3 includes a spelling dictionary of over 100,000 words to assist you in checking your spelling and hyphenating words. Medical and legal dictionaries are optional. You can also create your own supplemental dictionary of up to 4,500 characters.

Footnotes and outlines can be handled automatically, and frequently used keystrokes can be stored and recalled by function keys. DOS ASCII files can be merged into documents, and the document conversion facility allows DisplayWrite documents to be exchanged with other IBM text processing programs and products.

Personal Computer Family Requirements

DisplayWrite 3 version 1.10 runs on the IBM Personal Computer, Personal Computer XT, Portable Personal Computer or Personal Computer AT. With DOS 2.10, it requires 256KB of memory; with DOS 3.00 or 3.10 it requires 320KB of memory. Document conversion, background printing and other features use additional memory. In addition, DisplayWrite 3 version 1.10 requires two double-sided diskettes (320KB or 1.2MB) or one double-sided diskette and one fixed disk drive. It also requires the IBM Monochrome Display, IBM Color Display or IBM Enhanced Color Display and appropriate adapter.

DisplayWrite 3 version 1.10 supports the IBM Matrix Printer (except for the Personal Computer AT), the IBM Graphics Printer, the IBM Color Printer (no color function), the IBM Color Jetprinter (no color function), the IBM Proprinter, the IBM Wheelprinter, the IBM Quietwriter Printer, the IBM 5218 A03 or A04 Printwheel Printer when supported by the IBM 5218 Printer Driver Program version 1.10 and the requisite hardware (except for the Personal Computer AT), and the NEC 3550 printer. It supports additional printers with the appropriate user-defined printer tables.

3270 Personal Computer Family Requirements

DisplayWrite 3 version 1.10 supports members of the IBM 3270 Personal Computer family including the 3270-PC, 3270-PC/G, 3270-PC/GX (alphanumeric mode only), 3270-Personal Computer AT, 3270-Personal Computer AT/G (alphanumeric mode only) and 3270-Personal Computer AT/GX (alphanumeric mode only). Each system unit requires two double-sided diskette drives or one double-sided diskette drive and one fixed disk drive. The IBM 3270-PC and 3270-Personal Computer AT require the IBM Monochrome Display, the IBM 5272 Color Display or IBM 3295 Plasma Monitor (3270-PC models 24 or 26). The IBM 3270-PC/G, 3270-PC/GX, 3270-Personal Computer AT, 3270-Personal Computer AT/G and 3270-Personal Computer AT/GX require the IBM 5279 Color Display and IBM 5278 Display Attachment Unit, IBM 5379 Model C01 Color Display and IBM 5378 Model C01 Color Display Attachment Unit, or IBM 5379 Model M01 Monochrome Display and IBM 5378 Model M01 Monochrome Display Attachment Unit.

DisplayWrite 3 version 1.10 supports the IBM Graphics Printer, the IBM Color Printer (no color function), the IBM Color Jetprinter (no color function), the IBM Proprinter, the IBM Wheelprinter, the IBM Quietwriter Printer and the IBM 5218 A03 or A04 Printwheel Printer when supported by the IBM 5218 Printer Driver Program version 1.10 and the requisite hardware (3270-Personal Computer only).

DisplayWrite 3 Version 1.00 Upgrade to 1.10

DisplayWrite 3 version 1.00 owners can purchase an upgrade kit for version 1.10. Authorized IBM Personal Computer Dealers, Authorized IBM Personal Computer Software Dealers and IBM Marketing Representatives have a Product Upgrade Order form available. To order the upgrade, complete the order form, enclose a check for the amount listed on the form along with your state sales tax and enclose the blue-colored inside front cover of the first volume of the DisplayWrite 3 User's Guide [6322800]. Mail the completed order to the address on the form or return it to your IBM Branch Office. Orders must be postmarked no later than January 31, 1986.

IBM Personal Decision Series Enhancements

The IBM Personal Decision series is an integrated set of productivity tools and training aids for use by business and professional users of the IBM Personal Computer. This series (DATA, REPORTS+, GRAPHS, PLANS/PLANS+, WORDS, ATTACHMENT/36, ATTACHMENT/370, plus many training editions) contains facilities that allow you to easily access data from many sources covering a broad range of software programs and computers, including some IBM host systems. A wide variety of routines let you manage, analyze, and display your information. Enhanced usability and functions and support for selected IBM office systems are available as program updates for the DATA and the WORDS Editions.

The enhancements include:

- The WORDS Edition is modified to provide a facility for converting WORDS documents to revisable-form-text documents. These documents may then become usable by DisplayWrite 1, 2, or 3.
- The WORDS Edition also contains an additional interface that allows you to more easily check spelling in a WORDS document by using Word Proof, version I or II.
- The DATA edition is modified to provide a procedure for the installation of IBM DisplayWrite 1, 2, or 3 on the IBM Personal Decision series main menu.

Step-by-step instructions for installing and using the program updates are included with the program update. The program update for the IBM Personal Decision series DATA Edition provides a procedure to install IBM DisplayWrite 1, 2, or 3 on the Personal Decision series main menu.

Note: The program updates for the DATA and REPORTS+ Editions are interdependent. Neither should be installed without the other.

The DATA or the WORDS Edition require an IBM Personal Computer, IBM Personal Computer XT, IBM Personal Computer AT, IBM Personal Computer XT/370, IBM Personal Computer AT/370, IBM 3270 Personal Computer, or IBM Portable Personal Computer (some combinations of foreground and background colors

cause characters to be unreadable. The "Set PDS" utility in DATA allows you to change such combinations for all products, except GRAPHS, WORDS, DATA Training, and the tutorial in DATA.); two double-sided diskette drives, or a double-sided diskette drive and a fixed-disk drive (a double-sided diskette drive, and a fixed disk is required if DisplayWrite 1, 2, or 3 is installed with the Personal Decision Series); an IBM Monochrome Display, IBM Color Display, or IBM Enhanced Color Display and appropriate adapter; an IBM Proprinter, IBM Graphics Printer, IBM Personal Computer Color Printer (if color printing is desired), IBM Wheelprinter, or IBM Quietwriter Printer and appropriate adapter.

In addition, DATA Edition's asynchronous communication facilities require an RS232 cable, an asynchronous communications adapter card, and a compatible, full-duplex modem. The DATA Edition is a prerequisite for the other productivity programs. The PLANS/PLANS+, GRAPHS, and Attachment Editions may require additional hardware and software.

Memory requirements (assuming appropriate IBM Personal Computer DOS configuration and not including the IBM 3270-PC Control Program) are:

	DOS 2.00, 2.10	DOS 3.00, 3.10
Data, Reports+,	256KB	320KB
Graphs, Words, Data Training		
Plans, Plans+	320KB	384KB

Additional main storage may be required if the IBM DisplayWrite 1, 2, or 3 is installed with the Personal Decision Series.

For other WORDS or DATA Editions, licensed users may request program updates prior to the service termination date. There will be a charge for each update, as follows:

Extended-support subscribers	$20.00
Non-subscribers	$45.00

Information on the availability of program updates for subscribers and non-subscribers is available through the IBM Program Support Center, toll free, at 1-800-426-2266.

IBM Business Management Series Local Area Network Access Edition

The IBM Business Management Series Local Area Network Access Edition is designed to allow IBM Business Management Series Application Editions to operate with multiple users in the IBM PC Network under the IBM Personal Computer Disk Operating System (DOS) 3.1. When used with other IBM Business Management Series Editions, this product helps provide a growth opportunity for the IBM Business Management Series user by allowing multiple IBM Personal Computers to run any of the six Application Editions with a common set of programs and files. This common set of programs and files can be installed on an IBM Personal Computer XT or an IBM Personal Computer AT which becomes the server machine. Up to four additional workstation systems (nodes) can be attached and supported by the server machine, providing the capability for up to five concurrent users.

The following IBM Business Management Series Application Editions are supported by the IBM Business Management Series Local Area Network Access Edition:

- General Ledger Edition
- Accounts Payable Edition
- Accounts Receivable Edition
- Payroll Edition
- Order Entry and Invoicing Edition
- Inventory Accounting Edition

The IBM Business Management Series Local Area Network Access Edition adds to the IBM Business Management Series:

- Support for up to five concurrent users (one server and four remote)
- Shared IBM Business Management Series programs and data files
- Integrated asset protection feature logs and active users tracking by license/serial number to help prohibit unauthorized use of a license.
- An application job monitor that tracks
  - Which IBM Personal Computers are running IBM Business Management Series application jobs
  - Which of these jobs have reserved files open
  - Which application files are locked
- IBM Business Management Series job conflict manager, which intercepts all IBM Business Management Series application job requests, evaluates them in the context of currently running jobs and displays a message if a conflict situation exists.
- Documentation shipped on diskette in machine-readable form that will provide instructions for integrating IBM Business Management Series into the IBM PC Network.

The IBM Business Management Series Local Area Network Access Edition provides a resource manager for running multiple IBM Business Management Series applications in the IBM PC Network. Multiple users may run the same job, regardless of its function, against files for different companies. Within a company, the IBM Business Management Series Local Area Network Access Edition allows concurrent execution of tasks in the applications except those where files are updated. All IBM Business Management Series programs, data

files, and a central printer are shared using IBM PC Network hardware and software.

The IBM Business Management Series Local Area Network Access Edition requires:

- One Personal Computer to act as the server Personal Computer. This can be an IBM Personal Computer XT, IBM Personal Computer AT, or equivalent with 640KB of memory; one double-sided, double-density diskette and one fixed-disk drive; one monochrome or color monitor that can display at least 24 lines of 80 characters each; one IBM PC Network Adapter; one translator unit for IBM PC Network; one IBM Personal Computer printer capable of printing 132 characters-per-line in compressed or normal density; and DOS 3.10, the IBM PC Network Program, and an IBM Business Management Series Application Edition.

- One additional Personal Computer which can be an IBM Personal Computer, IBM Portable Personal Computer, IBM Personal Computer XT, IBM Personal Computer AT, or equivalent with 512KB of memory; One double-sided, double-density diskette drive; One PC Network Adapter; and DOS 3.10, the IBM PC Network Program, and IBM Business Management Series Local Area Network Access Edition.

IBM Extended Support is available from IBM as a separately priced offering. Licensed users may subscribe to IBM Extended Support via an annual subscription. One annual subscription charge provides support for any or all registered copies of IBM licensed programs within the Business Management Series. This product will be a part of the extended support offering for the Business Management Series. Basic support includes program service extension and extended support journals.

For optional fees, subscribers may obtain:

- Telephone assistance for the supported programs
- Program updates
- Additional extended support journals

This support is available to registered licensed users of the supported programs who desire program-related service or in-depth technical support beyond that which is provided with the program. This support is intended to complement the existing customer assistance provided by IBM Product Centers, IBM representatives and IBM Authorized Dealers.

These fee services are transacted by the IBM Application Program Support Department directly with the customer. There is no requirement for branch or dealer involvement with contract processing or fee collection.

Prices are structured as follows:

	Sub-scriber	Non-Sub-scriber
Annual subscription	$225	N/A
Telephone assistance -per incident	$40	N/A
-call pack (five incidents)	$180	N/A
Program updates	$30	N/A
Additional journals subscription	$30	N/A

IBM PC Network Analysis Program

The IBM PC Network Analysis Program is a network tool that helps the user to monitor and log the activities of an IBM PC Network and identify problem conditions in the network. The IBM PC Network Analysis Program may be run from a single IBM Personal Computer on the network as needed to monitor the network or to continuously monitor and log network activity.

The IBM PC Network Analysis Program provides the capability for the user to:

- Monitor network activity from one IBM Personal Computer on the IBM PC Network.
- Sample, analyze, and present network operational information at user-selected intervals.
- Monitor the IBM PC Network hardware operational conditions at an individual adapter level.
- Log information collected from the monitor function to disk or diskette for later review.
- Review various reports on network utilization and traffic.
- Monitor detailed, continuous traffic for a specific time period.
- Use a function that provides adapter information and acts as an operational check and assists in installing an adapter.
- Maintain a program-created directory of powered-on IBM PC Network adapters.
- Optionally convert log files to ASCII format for use with other user-provided analysis programs.
- Demonstrate most program functions with sample files without being connected to the IBM PC Network.
- Develop other user applications using a sample BASIC program for printing a log file that acts as a guide.
- Obtain on-line help information.

The IBM PC Network Analysis Program provides the functions necessary to determine the operational status and availability of other IBM PC Network adapters on the IBM PC Network and acts as a traffic monitor to investigate the origin, destination, and amount of network activity. The information is gathered at specific, user-selected intervals and may be reviewed from a single IBM Personal Computer on the IBM PC Network.

The IBM PC Network Analysis Program requires an IBM Personal Computer, IBM Personal Computer XT, IBM Personal Computer AT, or an IBM Portable Personal Computer, with at least 256KB of memory; an IBM PC Network Adapter; an IBM

Color Display with the IBM Color/Graphics Display Adapter, the IBM Monochrome Display with the IBM Monochrome Display and Printer Adapter, or the IBM Enhanced Color Display (in compatibility-mode only) with the IBM Enhanced Graphics Adapter, or its equivalent.

Other minimum hardware requirements are based on the number of adapters on the IBM PC Network, as follows:

Adapters	Minimum Requirements
2 to 60	Two double-sided diskette drives, or one double-sided diskette drive and either a fixed disk or one high-capacity diskette drive (IBM Personal Computer AT only).
61 to 200	One double-sided diskette drive and either a fixed disk or one high-capacity diskette drive (IBM Personal Computer AT only).
Over 200	One double-sided diskette drive and a fixed disk. For every 24 adapters over 200 in the network, an additional 2KB of memory is required.

A fixed disk is recommended when logging messages and notices.

Note: When IBM PC Network Analysis Program is installed with TopView, a fixed disk and 512KB of memory are required.

Software Requirements:

- IBM Personal Computer DOS 3.1
- IBM EZ-VU Runtime Facility (6316969)

While monitoring, IBM PC Network Analysis Program adds less than five percent of the traffic on the IBM PC Network. The actual traffic is determined by the number of IBM PC Network adapters monitored and the time interval between sample selection.

Note: The *Technical Reference PC Network* (6322505) publication is necessary for learning and using the IBM PC Network Analysis Program. All of the other programs in the same IBM Personal Computer, including those running under TopView control, must be suspended before IBM PC Network Analysis Program is run.

Voice Communications Subsystem

The IBM Personal Computer Voice Communications Operating Subsystem lets the IBM Personal Computer Voice Communications Adapter emulate an asynchronous modem, record and playback voice, recognize voice commands, convert text to speech, aid in using some PBX/CBX functions, recognize touch tone signals, generate telephony signals, and monitor line status. Together with the IBM Personal Computer Voice Communications Adapter, it forms the IBM Personal Computer Voice Communications Option.

The Voice Communications Option provides six discrete function areas that can be called by other applications:

- *Asynchronous Data Communications* and modem functions that provide full duplex Bell 103 (110 or 300 BPS), and Bell 212A (1200 BPS) compatible modem capabilities
- *Voice Command Recognition* on a speaker-dependent, discrete-utterance (words and/or phrases) recognition basis
- *Text-to-Speech (Synthetic Voice)* that transforms serial ASCII datastreams (including acronyms such as Dr.) into speech
- *Voice (Audio) Record/Playback* that digitizes speech at one of three compression options, storing it with silence compression, and restoring for playback
- *Line Monitoring* receives touch-tone signals from remote telephones for commands or data entry.
- *Telephony* lets applications start telephone calls (autodial), support other PBX functions (three-way calling), and automatically receive calls

The Voice Communications Operating Subsystem requires an IBM Personal Computer, IBM Personal Computer XT, or IBM Personal Computer AT; an IBM Personal Computer Voice Communications Adapter; and one double-sided diskette drive. It also requires DOS 2.10 or later. Memory requirements vary by DOS version, application requirements, and serial or concurrent use of functions, and can range from 22KB to 256KB in addition to the DOS requirements. The Exploring Voice Communications demonstration program (provided) requires an IBM Personal Computer with 320KB of memory.

Any applications written for the option must be written to the Voice Communications Applications Program Interface or the Voice Activated Keyboard Utility (see the software section), and accessed from an assembly language routine. A *Voice Communications Application Program Interface Reference* manual and Tool Kit diskette are available separately for experienced assembly language programmers. Or you may use one of the programs described below.

Augmented Phone Services

The IBM Personal Computer Augmented Phone Services application allows people with speech or hearing impairments who have an IBM Personal Computer with the Voice Communications Option to read touch-tone input over phone lines from people who are not speech- or hearing-impaired. The application uses keyboard input to generate synthetic speech over phone lines, and translates the touch-tone signals it recieves into messages that appear on the IBM Personal Computer display.

In addition, Augmented Phone Services can:

- Automatically answer and store received messages in text
- Store announcements, emergency messages, and "help" messages for later playback
- Maintain a phone directory and dial from entries in the directory, or from the keyboard
- Redial the last number called
- Store conversations on disk and print copies of conversations
- Add personal acronyms, abbreviations, proper names, etc. to the 8500 words in the base word list.

The Augmented Phone Services application requires an IBM Personal Computer, IBM Personal Computer XT, or IBM Personal Computer AT with at least 448KB of memory (additional memory is recommended); an IBM Color Display, IBM Monochrome Display, or IBM Enhanced Color Display, and the appropriate adapter; the IBM Personal Computer Voice Communications Option; two double-sided diskette drives or one double-sided diskette drive and one fixed disk; and DOS 2.10 or later.

Voice/Phone Assistant

The IBM Personal Computer Voice/Phone Assistant is the newest member of the IBM Assistant Series. It provides stored voice (audio) functions for the IBM Personal Computer.

Telephone answering machine capabilities let you record voice messages from the phone, phone line, or microphone and play messages back over the phone, phone lines, or speaker. Voice announcement messages can be recorded and selected. The Voice/Phone Assistant supports multiple users through multiple voice mailboxes with unique passwords. Recorded voice messages can be skipped, erased, or appended. All messages are date and time stamped and can be listed. Remote access permits authorized users to change the announcement message, access voice messages through passwords, record private messages, and transfer to other user-provided applications. Synthetic voice-generated help messages are also available to the remote user. The Voice/Phone Assistant can also provide remote access to other suitably programmed applications.

The Voice/Phone Assistant requires an IBM Personal Computer, IBM Personal Computer XT, or IBM Personal Computer AT with at least 256KB of memory (additional memory is recommended); an IBM Color Display, IBM Monochrome Display, or IBM Enhanced Color Display, and the appropriate adapter; the IBM Personal Computer Voice Communications Option; two double-sided diskette drives, or one double-sided diskette and a fixed disk drive; and DOS 2.10 or later.

Voice-Activated Keyboard Utility

The IBM Personal Computer Voice-Activated Keyboard Utility lets you substitute voice commands for keystrokes in existing applications without modification to the application. The utility provides vocabulary "overlays" for some existing applications. User-defined "overlays" can be created for other applications, and sample overlays for many popular applications are included.

The utility listens to commands spoken into a microphone connected to the IBM Personal Computer Voice Communications Adapter, identifies the spoken command in the "overlay", and sends the appropriate keystroke sequence to the application program. In addition, you can view the voice commands available to a particular application and the keystroke sequence. A menu-oriented interface is available to train the voice recognition component to your voice commands.

The Voice-Activated Keyboard Utility requires an IBM Personal Computer, IBM Personal Computer XT, or IBM Personal Computer AT with at least 320KB of memory (additional memory is recommended); an IBM Color Display, IBM Monochrome Display, or IBM Enhanced Color Display, and the appropriate adapter; the IBM Personal Computer Voice Communications Option; one double-sided diskette and a fixed disk drive; and DOS 2.10 or later. Not compatible with DOS system extensions: TopView, PC Network, and 3270 program.

Personal Telephone Manager Program

The IBM Personal Telephone Manager Program is available in two versions to support the IBM Personal Computer Voice Communications Option or IBM Personal Telephone Manager Adapter. It provides advanced telephone directory, dialing, and reminder functions. On-line tutorials, guidance screens, single key command access, and windowing increase the ease of use factors. The Personal Telephone Manager Program operates as either a stand-alone DOS application, or as a background DOS application.

The Personal Telephone Manager Program lets you store multiple telephone directories, and do on-hook dialing from the directories, keyboard, or telephone keypad. The program can also scan the screen of the IBM Personal Computer for a phone number to dial. The screen may originate from a personal computer application program, or from a host application using the IBM 3278/3279 Emulation Card and the IBM 3278/3279 Emulation program, or from a IBM System/36 or IBM System/38 using the IBM Enhanced Display Station Emulation Adapter and the IBM Enhanced 5250 Emulation Program.

Phone directories can be stored and accessed on the IBM System/36 with the PC Support/36 Virtual Disk Functions, or on the IBM System/38 with the PC Support/38 Virtual Disk Functions, or on the IBM Local Area Network by the IBM Local Area Network PC Adapter and the IBM PC

Network Program, or on the IBM PC Network using the IBM PC Network Adapter and the IBM PC Network Program. In addition, the IBM Personal Telephone Manager Program stores personal reminders, prints telephone directories, and stores up to eight of the last numbers dialed as a redial list.

The Personal Telephone Manager Program requires an IBM Personal Computer, IBM Personal Computer XT, or IBM Personal Computer AT; an IBM Color Display, IBM Monochrome Display, or IBM Enhanced Color Display, and the appropriate adapter; the IBM Personal Computer Voice Communications Option or IBM Personal Telephone Manager Adapter; one double-sided diskette; and DOS 2.00 or later. The program itself uses 108KB of memory as a stand-alone application and 128KB of memory as a background application.

Voice Communications Application Program Interface Reference

The *IBM Voice Communications Application Program Interface Reference* provides experienced assembly language programmers with the information needed to write voice, telephony and communications applications for the IBM Voice Communications Option. The reference includes a Tool Kit diskette of sample programs that assist in writing voice command driven applications and give examples of voice record/playback, telephone dialing, and text-to-speech function programs.

This two-volume book provides information on the IBM Voice Communications Adapter, its base commands, and functions for telephony, line monitoring, asynchronous communications, audio recording and playback, text-to-speech synthesis, and speech recognition. The exercises on the Tool Kit diskette provide hands-on experience with each application area.

The IBM Voice Communications Application Program Interface Reference requires an IBM Personal Computer, IBM Personal Computer XT, or IBM Personal Computer AT with at least 256KB of memory (additional memory is recommended); an IBM Color Display, IBM Monochrome Display, or IBM Enhanced Color Display, and the appropriate adapter; the IBM Personal Computer Voice Communications Option; two double-sided diskette drives, or one double-sided diskette and a fixed disk drive; and DOS 2.10 or later. Other functions may require an FCC approved telephone, a speaker, a microphone, and one or two telephone lines.

Copyrights, Trademarks, and Service Marks

ColorPaint by Marek and Rafal Krepec Incorporated.

ColorPlus is a trademark of Plantronics Corporation.

CompuServe is a trademark of CompuServe, Incorporated.

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

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

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

DIF is a trademark of Software Arts, Incorporated.

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

EasyWriter is a trademark of Information Unlimited Software, Incorporated.

Electric Poet is a registered trademark of Control Color Corporation.

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

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

IBM is a registered trademark of International Business Machines Corp.

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

Jumpman is a trademark of EPYX, Incorporated.

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

Logo is a trademark of Logo Computer Systems Incorporated.

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

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

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

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

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

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

PCjr is a trademark of International Business Machines Corp.

PC Mouse is a trademark of Metagraphics/Mouse Systems.

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

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

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

pfs: is a registered trademark of Software Publishing Corporation.

PlannerCalc is a trademark of Comshare.

REALCOLOR is a trademark of Micro Developed Systems, Inc.

SHAMUS is a trademark of SynSoft(TM).

SMARTMODEM is a trademark of Hayes MicroComputer Products, Inc.

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

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

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

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

TopView is a trademark of International Business Machines Corp.

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

UNIX is a trademark of AT&T Bell Laboratories.

VisiCalc is a trademark of VisiCorp.

Visi On is a trademark of VisiCorp.

WD212-X is a trademark of Wolfdata, Inc

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

WordStar is a trademark of MicroPro International Corporation.

XENIX is a trademark of Microsoft Corporation.

Z-80 is a registered trademark of Zilog.

"When you write a program that you intend to debug with RDT, it is advisable to place an INT3 instruction at the beginning of your program. (page 10)

"Professional Debug Facility has many specific commands that give you much greater control of the debug process. (page 15)

"The IBM PC*jr* architecture includes an advanced sound-generating subsystem that produces up to three tones simultaneously. (page 22)

"With the virtual disk as the default drive, access is silent and nearly instantaneous. (page 30)

G320-0846-00